



Solaris Naming Administration Guide

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

Part No: 802-5752
August 1997

Copyright 1997 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 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. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, SunSoft, SunDocs, SunExpress, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was 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 GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

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

Copyright 1997 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite 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 y en a. 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.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunSoft, SunDocs, SunExpress, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée 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 Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Contents

Preface xxix

Part I Introduction to Solaris Naming

1. Introduction to Name Services 3

What Is a Name Service? 3

Solaris Name Services 8

DNS 8

/etc Files 9

NIS 9

NIS+ 9

FNS 10

2. The Name Service Switch 11

About the Name Service Switch 11

Format of the `nsswitch.conf` File 12

Comments in `nsswitch.conf` Files 16

Keyserver and `publickey` Entry in the Switch File 17

The `nsswitch.conf` Template Files 17

The Default Switch Template Files 17

Default `nsswitch.conf` File 20

DNS and Internet Access 20

	DNS Forwarding for NIS+ Clients	20
	DNS Forwarding for NIS Clients	20
	Adding Compatibility With +/- Syntax	21
	The Switch File and Password Information	22
	FNS and the Name Service Switch	22
	Maintaining Consistency Between FNS and the Switch File	22
	Namespace Updates	23
	Part II NIS+ Introduction and Overview	
3.	Introduction to NIS+	27
	About NIS+	27
	What NIS+ Can Do for You	28
	How NIS+ Differs From NIS	29
	NIS+ Security	32
	NIS+ and the Name Service Switch	33
	Solaris 1.x Releases and NIS-Compatibility Mode	33
	NIS+ Administration Commands	34
	NIS+ API	36
4.	The NIS+ Namespace	39
	NIS+ Files and Directories	39
	Structure of the NIS+ Namespace	40
	Directories	42
	Domains	43
	Servers	44
	How Servers Propagate Changes	45
	NIS+ Clients and Principals	47
	Principal	47
	Client	47
	The Cold-Start File and Directory Cache	48

An NIS+ Server Is Also a Client	51
Naming Conventions	52
NIS+ Domain Names	53
Directory Object Names	54
Tables and Group Names	55
Table Entry Names	55
Host Names	56
NIS+ Principal Names	56
Accepted Name Symbols	57
NIS+ Name Expansion	57
NIS_PATH Environment Variable	57
5. NIS+ Tables and Information	59
NIS+ Table Structure	59
Columns and Entries	61
Search Paths	62
Ways to Set Up Tables	64
How Tables Are Updated	65
6. Security Overview	67
Solaris Security—Overview	67
NIS+ Security—Overview	69
NIS+ Principals	70
NIS+ Security Levels	71
Security Levels and Password Commands	71
NIS+ Authentication and Credentials—Introduction	72
User and Machine Credentials	72
DES versus LOCAL Credentials	72
User Types and Credential Types	74
NIS+ Authorization and Access—Introduction	74

	Authorization Classes	75
	NIS+ Access Rights	78
	The NIS+ Administrator	78
	NIS+ Password, Credential, and Key Commands	79
	Part III Administering NIS+	
7.	Administering NIS+ Credentials	83
	NIS+ Credentials	84
	How Credentials Work	84
	Credential versus Credential Information	84
	Authentication Components	85
	How Principals are Authenticated	85
	The DES Credential in Detail	89
	DES Credential Secure RPC Netname	89
	DES Credential Verification Field	90
	How the DES Credential Is Generated	90
	Secure RPC Password versus Login Password Problem	91
	Cached Public Keys Problems	92
	Where Credential-Related Information Is Stored	92
	The <code>cred</code> Table in Detail	94
	Creating Credential Information	95
	The <code>nisaddcred</code> Command	96
	Related Commands	97
	How <code>nisaddcred</code> Creates Credential Information	97
	The Secure RPC Netname and NIS+ Principal Name	98
	Creating Credential Information for the Administrator	99
	Creating Credential Information for NIS+ Principals	99
	Administering NIS+ Credential Information	103
	Updating Your Own Credential Information	103

	Removing Credential Information	104
8.	Administering NIS+ Keys	105
	NIS+ Keys	105
	Keylogin	106
	Changing Keys for an NIS+ Principal	107
	Changing the Keys	108
	Changing Root Keys From Root	108
	Changing Root Keys From Another Machine	110
	Changing the Keys of a Root Replica from the Replica	110
	Changing the Keys of a Nonroot Server	111
	Updating Public Keys	112
	The <code>nisupdkeys</code> Command	112
	Updating Public Keys Arguments and Examples	112
	Updating IP Addresses	114
	Updating Client Key Information	114
	Globally Updating Client Key Information	114
9.	Administering NIS+ Access Rights	117
	NIS+ Access Rights	118
	Introduction to Authorization and Access Rights	118
	Authorization Classes—Review	118
	Access Rights—Review	119
	Concatenation of Access Rights	119
	How Access Rights Are Assigned and Changed	120
	Table, Column, and Entry Security	120
	Where Access Rights Are Stored	125
	Viewing an NIS+ Object's Access Rights	125
	Default Access Rights	126
	How a Server Grants Access Rights to Tables	127

Specifying Access Rights in Commands	127
Syntax for Access Rights	128
Displaying NIS+ Defaults—The <code>nisdefaults</code> Command	131
Setting Default Security Values	133
Displaying the Value of <code>NIS_DEFAULTS</code>	133
Changing Defaults	134
Resetting the Value of <code>NIS_DEFAULTS</code>	134
Specifying Nondefault Security Values at Creation Time	135
Changing Object and Entry Access Rights	135
Using <code>nischmod</code> to Add Rights	136
Using <code>nischmod</code> to Remove Rights	136
Specifying Column Access Rights	137
Setting Column Rights When Creating a Table	137
Adding Rights to an Existing Table Column	138
Removing Rights to a Table Column	139
Changing Ownership of Objects and Entries	139
Changing Object Owner With <code>nischown</code>	139
Changing Table Entry Owner With <code>nischown</code>	139
Changing an Object or Entry's Group	140
Changing an Object's Group With <code>nischgrp</code>	140
Changing a Table Entry's Group With <code>nischgrp</code>	141
10. Administering Passwords	143
Using Passwords	144
Logging In	144
Changing Your Password	146
Choosing a Password	147
Administering Passwords	149
<code>nsswitch.conf</code> File Requirements	149

	The nispasswd Command	149
	The yppasswd Command	150
	The passwd Command	150
	The nistbladm Command	153
	Related Commands	157
	Displaying Password Information	157
	Changing Passwords	159
	Locking a Password	160
	Managing Password Aging	161
	Specifying Password Criteria and Defaults	168
11.	Administering NIS+ Groups	173
	Solaris Groups	173
	NIS+ Groups	174
	Related Commands	174
	NIS+ Group Member Types	175
	Member Types	175
	Nonmember Types	176
	Group Syntax	176
	Using niscat With NIS+ Groups	177
	Listing the Object Properties of a Group	177
	The nisgrpadm Command	178
	Creating an NIS+ Group	179
	Deleting an NIS+ Group	180
	Adding Members to an NIS+ Group	180
	Listing the Members of an NIS+ Group	181
	Removing Members From an NIS+ Group	182
	Testing for Membership in an NIS+ Group	182
12.	Administering NIS+ Directories	185

NIS+ Directories	186
Using the <code>niscat</code> Command With Directories	186
Listing the Object Properties of a Directory	186
The <code>nislsls</code> Command and <code>niscat</code> Command With Directories	187
Listing the Contents of a Directory—Terse	188
Listing the Contents of a Directory—Verbose	188
The <code>nismkdir</code> Command	189
Creating a Directory	189
Adding a Replica to an Existing Directory	191
The <code>nisrmdir</code> Command	192
Removing a Directory	193
Disassociating a Replica From a Directory	193
The <code>nisrm</code> Command	194
Removing Nondirectory Objects	195
The <code>rpc.nisd</code> Command	195
Starting a NIS-Compatible Daemon	196
Starting a DNS-Forwarding NIS-Compatible Daemon	196
Stopping the NIS+ Daemon	197
The <code>nisinit</code> Command	197
Initializing a Client	197
Initializing the Root Master Server	198
The <code>nis_cachemgr</code> Command	199
Starting and Stopping the Cache Manager	199
The <code>nisshowcache</code> Command	199
Displaying the Contents of the NIS+ Cache	200
Pinging and Checkpointing	200
The <code>nisping</code> Command	201
Displaying When Replicas Were Last Updated	201

	Forcing a Ping	201
	Checkpointing a Directory	202
	The <code>nislog</code> Command	203
	Displaying the Contents of the Transaction Log	204
	The <code>nischttl</code> Command	205
	Changing the Time-to-Live of an Object	207
	Changing the Time-to-Live of a Table Entry	207
13.	Administering NIS+ Tables	209
	NIS+ Tables	210
	The <code>nistbladm</code> Command	210
	<code>nistbladm</code> Syntax Summary	210
	<code>nistbladm</code> and Column Values	212
	<code>nistbladm</code> , Searchable Columns, and Keys	<code>nistbladm</code> and Column Values 213
	<code>nistbladm</code> and Indexed Names	215
	<code>nistbladm</code> and Groups	215
	Creating a New Table	216
	Specifying Table Columns	216
	Deleting a Table	218
	Adding Entries to a Table	218
	Adding a Table Entry With the <code>-a</code> Option	219
	Adding a Table Entry With the <code>-A</code> Option	221
	Modifying Table Entries	222
	Editing a Table Entry With the <code>-e</code> Option	222
	Editing a Table Entry With the <code>-E</code> Option	224
	Removing Table Entries	224
	Removing Single Table Entries	225
	Removing Multiple Entries From a Table	225

The <code>niscat</code> Command	226
Syntax	226
Displaying the Contents of a Table	227
Displaying the Object Properties of a Table or Entry	228
The <code>nismatch</code> and <code>nisgrep</code> Commands	229
About Regular Expressions	230
Syntax	231
Searching the First Column	232
Searching a Particular Column	232
Searching Multiple Columns	233
The <code>nislN</code> Command	233
Syntax	234
Creating a Link	234
The <code>nissetup</code> Command	234
Expanding a Directory Into an NIS+ Domain	235
Expanding a Directory Into an NIS-Compatible Domain	235
The <code>nisaddent</code> Command	236
Syntax	236
Loading Information From a File	237
Loading Data From an NIS Map	238
Dumping the Contents of an NIS+ Table to a File	240
14. Server-Use Customization	241
NIS+ Servers and Clients	241
Default Client Search Behavior	242
Designating Preferred Servers	242
NIS+ Over Wide Area Networks	242
Optimizing Server-Use—Overview	243
<code>nis_cachemgr</code> is Required	243

Global Table or Local File	243
Preference Rank Numbers	244
Preferred Only Servers Versus All Servers	245
Viewing Preferences	246
Server and Client Names	246
Server Preferences	246
When Server Preferences Take Effect	246
Using the <code>nisprefadm</code> Command	247
Viewing Current Server Preferences	249
How to View Preferences for a Machine	249
How to View Global Preferences for Single Machine	249
How to View Global Preferences for a Subnet	250
How to Specify Preference Rank Numbers	250
Specifying Global Server Preferences	250
How to Set Global Preferences for a Subnet	251
How to Set Global Preferences for an Individual Machine	251
How to Set Global Preferences for a Remote Domain	252
Specifying Local Server Preferences	253
How to Set Preferences on a Local Machine	253
Modifying Server Preferences	254
How to Change a Server's Preference Number	254
How to Replace One Server With Another in a Preference List	254
How to Remove Servers From Preference Lists	255
How to Replace an Entire Preferred Server List	256
Specifying Preferred-Only Servers	256
How to Specify Preferred-Only Servers	257
How to Revert to Using Non-Preferred Servers	257
Ending Use of Server Preferences	258

	How to Eliminate Global Server Preferences	258
	How to Eliminate Local Server Preferences	258
	Putting Server Preferences Into Immediate Effect	260
	How to Immediately Implement Preference Changes	260
15.	NIS+ Backup and Restore	261
	Backing Up Your Namespace With <code>nisbackup</code>	261
	<code>nisbackup</code> Syntax	262
	What <code>nisbackup</code> Backs Up	263
	The Backup Target Directory	264
	Maintaining a Chronological Sequence of NIS+ Backups	264
	Backing Up Specific NIS Directories	265
	Backing Up an Entire NIS+ Namespace	265
	Backup Directory Structure	265
	Backup Files	266
	Restoring Your NIS+ Namespace With <code>nisrestore</code>	267
	Prerequisites to Running <code>nisrestore</code>	267
	<code>nisrestore</code> Syntax	268
	Using <code>nisrestore</code>	268
	Using Backup/Restore to Set Up Replicas	269
	Replacing Server Machines	270
	Machine Replacement Requirements	270
	How to Replace Server Machines	270
16.	Removing NIS+	273
	Removing NIS+ From a Client Machine	273
	Removing NIS+ That Was Installed Using <code>nisclient</code>	273
	Removing NIS+ That Was Installed Using NIS+ Commands	274
	Removing NIS+ From a Server	274
	Removing the NIS+ Namespace	276

Part IV Administering NIS

17. Network Information Service (NIS) 281

NIS Introduction 281

 NIS Architecture 282

 NIS and NIS+ 283

 NIS and FNS 284

NIS Machine Types 284

 NIS Servers 284

 NIS Clients 285

NIS Elements 285

 The NIS Domain 285

 NIS Daemons 285

 NIS Utilities 286

 NIS Maps 287

 Summary of NIS-Related Commands 291

NIS Binding 292

 Server-List Mode 293

 Broadcast Mode 293

Differences Between This and Earlier NIS Versions 294

 NSKit Discontinued 294

 The `ypupdated` Daemon 294

`/var/yp/securenets` 294

 Multihomed Machine Support 295

 Sun Operating System 4.X Compatibility Mode 295

 Using the Name Service Switch 296

18. Administering NIS 299

 Password Files and Namespace Security 300

 Administering NIS Users 300

Adding a New User to an NIS Domain	300
User Passwords	302
Netgroups	303
Working With NIS Maps	304
Obtaining Map Information	304
Changing a Map's Master Server	305
Modifying Configuration Files	307
Modifying and Using the <code>Makefile</code>	307
Updating Existing Maps	310
Adding a New Slave Server	316
Using NIS with C2 Security	318
Changing a Machine's NIS Domain	318
Using NIS in Conjunction With DNS	318
Problems in Mixed NIS Domains	320
Turning Off NIS Services	320
NIS Problem Solving and Error Messages	321
Part V Administering FNS	
19. FNS Quickstart	325
Federated Naming Service (FNS)	326
X/Open Federated Naming (XFN)	326
Why FNS?	326
Composite Names and Contexts	326
Composite Names	326
Contexts	327
Attributes	327
Enterprise Naming Services	328
NIS+	328
NIS	328

Files-Based	329
Global Naming Services	329
FNS Naming Policies	329
Organization Names	330
Site Names	331
User Names	331
Host Names	332
Service Names	332
File Names	332
Getting Started	333
Designating a Non-Default Naming Service	333
Creating the FNS Namespace	333
NIS+ Considerations	334
NIS Considerations	335
Files Considerations	335
Browsing the FNS Namespace	335
Listing Context Contents	335
Displaying the Bindings of a Composite Name	336
Showing the Attributes of a Composite Name	337
Searching for FNS Information	337
Updating the Namespace	338
FNS Administration Privileges	338
Binding a Reference to a Composite Name	339
Removing Bindings	341
Creating New Contexts	341
Creating File Contexts	342
Creating Printer Contexts	343
Destroying Contexts	345

	Working With Attributes	345
	Federating a Global Namespace	346
	Copying and Converting FNS Contexts	347
	Namespace Browser Programming Examples	348
	Listing Names Bound in a Context	349
	Creating a Binding	350
	Listing and Working With Object Attributes	351
	Searching for Objects in a Context	355
20.	Federated Naming Overview	357
	XFN and FNS	357
	The XFN Model	359
	XFN Architectural Model	359
	User's View	362
	File System View	362
	Application View	363
	API Usage Model	364
	Federated Naming Service	364
	FNS and Application Development	364
	FNS and Composite Names	365
	FNS Policy Principles	365
	FNS in the Solaris Environment	366
	Solaris Enterprise-Level Naming Services	366
	FNS and NIS+ Naming	367
	FNS and NIS Naming	367
	FNS and Files-Based Naming	368
	Global Naming Services	368
	FNS and DNS	369
	FNS and X.500	369

FNS and Applications	370
FNS File Naming	370
FNS Printer Naming	370
FNS Application Support	370
Administering FNS	371
Troubleshooting and Error Messages	372
21. FNS Policies	373
Introduction to FNS and XFN Policies	374
What FNS Policies Specify	374
What FNS Policies Do Not Specify	374
Policies for the Enterprise Namespace	375
Default FNS Enterprise Namespaces	375
Enterprise Namespace Identifiers	376
Default FNS Namespaces	377
Significance of Trailing Slash	381
FNS Reserved Names	381
Composite Name Examples	382
Structure of the Enterprise Namespace	383
Enterprise Root	386
Using Three Dots to Identify the Enterprise Root	386
Using <code>org//</code> to Identify the Enterprise Root	386
Enterprise Root Subordinate Contexts	387
Initial Context Bindings for Naming Within the Enterprise	391
FNS and Enterprise Level Naming	396
How FNS Policies Relate to NIS+	397
How FNS Policies Relate to NIS	399
How FNS Policies Relate to Files-Based Naming	400
Target Client Applications of FNS Policies	400

FNS File System Namespace	403
NFS File Servers	403
The Automounter	404
The FNS Printer Namespace	405
Policies for the Global Namespace	405
Initial Context Bindings for Global Naming	406
Federating DNS	406
Federating X.500/LDAP	407
22. FNS and Enterprise Name Services	409
FNS and Enterprise-Level Naming Services	409
Choosing an Enterprise-Level Name Service	410
FNS and Naming Service Consistency	410
FNS and Solstice AdminSuite	410
Checking Naming Inconsistencies	411
Selecting a Naming Service	412
Default Naming Service	412
When NIS+ and NIS Coexist	413
Advanced FNS and NIS+ Issues	413
Migrating to NIS+ From NIS or Files-Based Naming	413
Mapping FNS Contexts to NIS+ Objects	413
Browsing FNS Structures Using NIS+ Commands	413
Checking Access Control	414
Advanced FNS and NIS Issues	415
NIS and FNS Maps and Makefiles	416
Large FNS Contexts	416
Printer Backward Compatibility	417
Migrating From NIS to NIS+	417
Advanced FNS and File-Based Naming Issues	418

	FNS Files	418
	Migrating From Files-Based Naming to NIS or NIS+	419
	Printer Backward Compatibility	419
23.	Enterprise Level Contexts	421
	Creating Enterprise Level Contexts	422
	Creating an Organization Context	423
	All Hosts Context	424
	Single Host Context	425
	Host Aliases	425
	All-Users Context	425
	Single User Context	426
	Service Context	426
	Printer Context	427
	Generic Context	427
	Site Context	428
	File Context	429
	Namespace Identifier Context	429
	Administering Enterprise Level Contexts	430
	Displaying the Binding	430
	Listing the Context	431
	Binding a Composite Name to a Reference	433
	Removing a Composite Name	436
	Renaming an Existing Binding	436
	Destroying a Context	437
24.	Administering File Contexts	439
	File Contexts Administration	439
	Creating a File Context With <code>fncreate_fs</code>	440
	Creating File Contexts With an Input File	441

	Creating File Contexts With Command-line Input	442
	Advanced Input Formats	443
	Multiple Mount Locations	443
	Variable Substitution	443
	Backward Compatibility Input Format	444
25.	FNS and Global Naming Systems	445
	FNS and Global Naming Systems	445
	Obtaining the Root Reference	446
	NIS+ Root Reference	446
	NIS Root Reference	447
	Federating Under DNS	448
	Federating Under X.500/LDAP	449
	Specifying an X.500 Root Reference	449
	Specifying an X.500 Client API	451
26.	Administering FNS Attributes	453
	Attributes Overview	453
	Examining Attributes	453
	Searching for Objects Associated With an Attribute	455
	Customizing Attribute Searches	455
	Updating Attributes	456
	Adding an Attribute	457
	Deleting an Attribute	458
	Listing an Attribute	458
	Modifying an Attribute	459
	Other Options	459
	Part VI Administering DNS	
27.	Introduction to DNS	463
	DNS Basics	464

Name-to-Address Resolution	464
DNS Administrative Domains	466
in.named and DNS Name Servers	467
DNS Clients and the Resolver	467
Introducing the DNS Namespace	468
DNS Namespace Hierarchy	468
DNS Hierarchy in a Local Domain	469
DNS Hierarchy and the Internet	469
Zones	473
Reverse Mapping	473
DNS Servers	474
Master Servers	475
Caching and Cache-only Servers	475
Root Domain Name Server	476
How DNS Affects Mail Delivery	477
DNS Boot and Data Files	478
Names of DNS Data Files	478
The named.boot File	479
The named.ca File	479
The hosts File	480
The hosts.rev File	480
The named.local File	480
\$INCLUDE Files	480
Data File Resource Record Format	481
Standard Resource Record Format	481
Special Resource Record Characters	482
Control Entries	483
Resource Record Types	484

Solaris DNS BIND Implementation	491
28. Administering DNS	493
Trailing Dots in Domain Names	493
Modifying DNS Data Files	494
Changing the SOA Serial Number	494
Forcing in.named to Reload DNS Data	494
Adding and Deleting Machines	495
Adding a Machine	495
Removing a Machine	496
Adding Additional DNS Servers	496
Creating DNS Subdomains	497
Planning Your Subdomains	498
Setting Up a Subdomain	499
DNS Error Messages and Problem Solving	501
Part VII Appendices	
A. Problems and Solutions	505
Troubleshooting NIS+	506
NIS+ De-Bugging Options	506
NIS+ Administration Problems	507
NIS+ Database Problems	511
NIS+ and NIS Compatibility Problems	512
NIS+ Object Not Found Problems	514
NIS+ Ownership and Permission Problems	517
NIS+ Security Problems	519
NIS+ Performance and System Hang Problems	528
NIS+ System Resource Problems	532
NIS+ User Problems	533
Other NIS+ Problems	535

NIS Problems and Solutions	537
Symptoms:	537
NIS Problems Affecting One Client	537
NIS Problems Affecting Many Clients	541
DNS Problems and Solutions	544
Clients Can Find Machine by Name but Server Cannot	544
Changes Do Not Take Effect or Are Erratic	545
DNS Client Cannot Lookup “Short” Names	546
Reverse Domain Data Not Correctly Transferred to Secondary	546
Server Failed and Zone Expired Problems	547
rlogin, rsh, and ftp Problems	548
Other DNS Syntax Errors	548
FNS Problems and Solutions	549
Cannot Obtain Initial Context	549
Nothing in Initial Context	549
“No Permission” Messages (FNS)	550
fnlist Does not List Suborganizations	550
Cannot Create Host- or User-related Contexts	551
Cannot Remove a Context You Created	551
Name in Use with fnunbind	552
Name in Use with fnbind/fncreate -s	552
fndestroy/fnunbind Does Not Return Operation Failed	553
B. Error Messages	555
About Error Messages	555
Error Message Context	555
Context-Sensitive Meanings	556
How Error Messages Are Alphabetized	556
Numbers in Error Messages	557

	FNS Error Messages	557
	Common Namespace Error Messages	558
C.	Information in NIS+ Tables	603
	NIS+ Tables	604
	NIS+ Tables and Other Name Services	604
	NIS+ Table Input File Format	604
	auto_home Table	605
	auto_master Table	605
	bootparams Table	606
	client_info Table	608
	cred Table	608
	ethers Table	609
	group Table	610
	hosts Table	610
	mail_aliases Table	611
	netgroup Table	612
	netmasks Table	613
	networks Table	614
	passwd Table	614
	protocols Table	616
	rpc Table	617
	services Table	617
	timezone Table	618
D.	FNS Reference Formats and Syntax	619
	DNS Text Record Format for XFN References	619
	X.500 Attribute Syntax for XFN References	621
	Object Classes	622
	Glossary	625

Preface

Solaris Naming Administration Guide describes how to customize and administer the four name services: NIS+, NIS, FNS, and DNS once they have been initially set up and configured. This manual is part of the Solaris 2.6 Release System and Network Administration manual set.

Who Should Use This Book

This book is written primarily for system and network administrators. It assumes you are an experienced system administrator.

Although this book introduces networking concepts relevant to Solaris name services, it makes no attempt to explain networking fundamentals or 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.

(*Solaris Naming Setup and Configuration Guide* explains how to initially set up and configure the four Solaris naming services.)

How This Book Is Organized

This book has seven parts:

Part I, Introduction to Solaris Naming

This part provides an introduction and overview of namespaces and Solaris naming services, and using the `nsswitch.conf` file to coordinate naming service usage.

- Chapter 1 provides an overview describing what *namespaces* and *naming services* are and what they do, then briefly describes the four Solaris naming services: DNS, NIS, FNS, and NIS+.
- Chapter 2. You use the name service switch to coordinate the use of different naming services. This chapter describes the name service switch, what it does, and how clients use it to obtain naming information from one or more sources.

Part II, NIS+ Introduction and Overview

This part describes NIS+:

- Chapter 3 provides an overview of the *Network Information Service Plus* (NIS+).
- Chapter 4 describes the structure of the NIS+ namespace, the servers that support it, and the clients that use it.
- Chapter 5 describes the structure of NIS+ tables and provides a brief overview of how they can be set up.
- Chapter 6 describes the NIS+ security system and how it affects the entire NIS+ namespace.

Part III, Administering NIS+

This part describes how to administer a functioning NIS+ namespace.

- Chapter 7 describes NIS+ credentials and how to administer them.
- Chapter 8 describes NIS+ keys and how to administer them.
- Chapter 9 describes NIS+ access rights and how to administer them.
- Chapter 10 describes how to use the `passwd` command from the point of view of an ordinary user (NIS+ principal) and how an NIS+ administrator manages the password system.
- Chapter 11 describes NIS+ groups and how to administer them.
- Chapter 12 describes NIS+ directory objects and how to administer them.
- Chapter 13 describes NIS+ tables and how to administer them. (See Appendix C, for detailed descriptions of the default NIS+ tables.)
- Chapter 14 describes how to customize and control which servers NIS+ clients use.
- Chapter 15 describes how to backup and restore an NIS+ namespace.

- Chapter 16 describes how to use the NIS+ directory administration commands to remove NIS+ from clients, servers, and the namespace as a whole.

Part IV, Administering NIS

- This part describes the *Network Information Service* (NIS) and how to administer it.
- Chapter 17 describes NIS.
- Chapter 18 describes how to administer NIS.

Part V, Administering FNS

This part describes the Federated Naming Service (FNS) and how to administer it.

- Chapter 19 is for experienced administrators. It provides a brief overview of FNS, basic set up and configuration steps, and a programming example.
- Chapter 20 describes the Federated Naming Service (FNS) which is Sun's implementation of the X/Open XFN federated naming standard.
- Chapter 21 describes FNS policies.
- Chapter 22 describes the relationship between FNS and enterprise level naming services.
- Chapter 23 describes how to individually create, and administer existing enterprise-level contexts.
- Chapter 24 describes how to administer application specific contexts.
- Chapter 25 describes two global naming systems (DNS and X.500/LDAP) and how to federate them under FNS.
- Chapter 26 describes FNS attributes and how to administer them.

Part VI, Administering DNS

This part describes the Domain Name System and how to administer it.

- Chapter 27 describes the Domain Name System.
- Chapter 28 describes how to administer the Domain Name System.

Part VII, Appendices

This part provides reference material and a glossary.

- Appendix A describes some of the problems you may encounter while administering Solaris namespaces and how to correct them.
- Appendix B provides an alphabetic listing of some commonly encountered error messages.
- Appendix C summarizes the information stored in the default NIS+ tables. (See Chapter 13 for general information regarding NIS+ tables and the commands used to administer them.)
- Appendix D contains supplemental information about the use of DNS text (TXT) records and the use of X.500 attributes in XFN references.
- *Glossary* defines namespace terms.

Related Books

- *Solaris Naming 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 documentation set:

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

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of SunExpressTM on The Internet at <http://www.sun.com/sunexpress>.

What Typographic Changes 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. <code>machine_name%</code> You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	<code>machine_name%su</code> Password:
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type <code>rm <i>filename</i></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 options</i> . 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	<code>machine_name%</code>
C shell superuser prompt	<code>machine_name#</code>
Bourne shell and Korn shell prompt	<code>\$</code>
Bourne shell and Korn shell superuser prompt	<code>#</code>

Introduction to Solaris Naming

This part provides an introduction and overview of namespaces and Solaris naming services, and using the `nsswitch.conf` file to coordinate naming service usage.

- Chapter 1
- Chapter 2

Introduction to Name Services

This chapter provides an overview describing what *namespaces* and *naming services* are and what they do. (Other names for name services are *network information services* and *directory services*.) This chapter then briefly describes the four Solaris naming services: DNS, NIS, FNS, and NIS+.

- “What Is a Name Service?” on page 3
- “Solaris Name Services” on page 8
- “DNS” on page 8
- “NIS” on page 9
- “NIS+” on page 9
- “FNS” on page 10

Directions for setting up NIS+, NIS, DNS, and FNS namespaces are contained in *Solaris Naming 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 in a central place that users, workstations, and applications must have to communicate across the network such as:

- Machine (host) names and addresses
- User names
- Passwords
- Access permissions

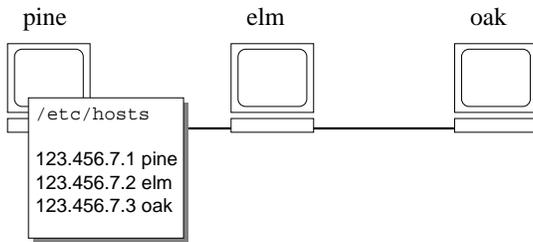
Without a central name service, each workstation would have to maintain its own copy of this information. Name service information may be stored in files, maps, or database tables. Centrally locating this data makes it easier to administer large networks.

Name services are fundamental to any computing network. Among other features, a name service provides functionality that:

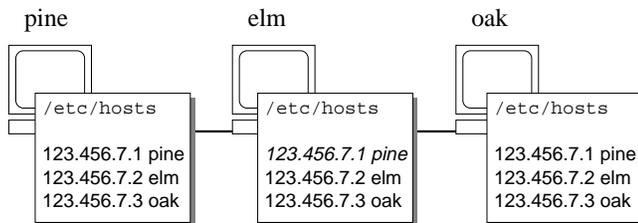
- Associates (*binds*) names with objects
- Resolves names to objects
- Removes bindings
- Lists names
- Renames

A network information service enables workstations to be identified by common names instead of numerical addresses. This makes communication simpler because users don't have to remember and try to enter cumbersome numerical addresses like "129.44.3.1."

For example, take a simple network of three workstations named, `pine`, `elm`, and `oak`. Before `pine` can send a message to either `elm` or `oak`, it must know their numerical 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.

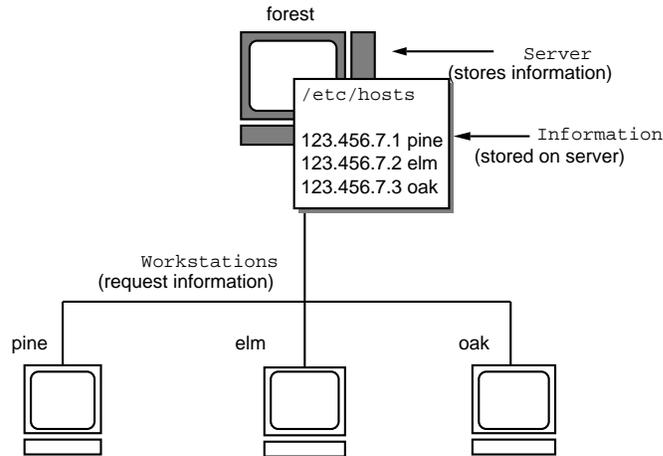


Addresses are not the only network information that workstations need to store. They also need to store security information, mail data, information about their Ethernet interfaces, network services, groups of users allowed to use the network, services offered on the network, and so on. As networks offer 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.

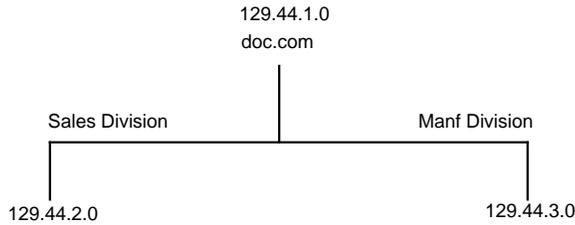
A network information service solves this problem. It stores network information on servers and provides it to any workstation that asks for it:



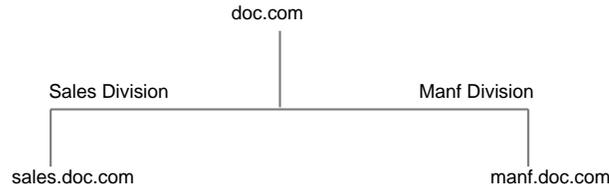
The workstations are known as *clients* of the server. Whenever information about the network changes, instead of updating each client's local file, an administrator updates only the information stored by the network information service. This reduces errors, inconsistencies between clients, and the sheer 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. For example, assume your company has set up a network and connected it to the Internet. The Internet has assigned your network the network number of 129.44.0.0 and the domain name `doc.com`. Your company has two divisions, Sales and Manufacturing (Manf), so its network is divided into a main net and two subnets, one for each division. Each net has its own address:



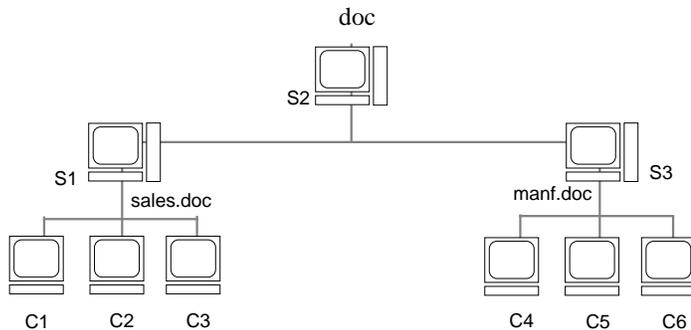
Each division could be identified by its network address, as shown above, but descriptive names made possible by name services would be preferable:



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

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 doc.com network is supported by three servers, S1, S2, and S3, and that two of those servers, S1 and S3, support clients:

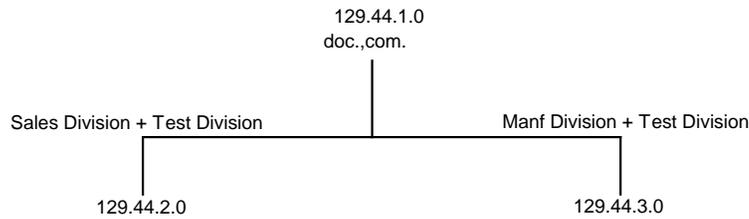


Clients C1, C2, and C3 would obtain their network information from server S1. Clients C4, C5, and C6 would obtain it from server S3. The resulting network is summarized in Table 1-1. (The table is a generalized representation of that network but does not resemble an actual network information map.)

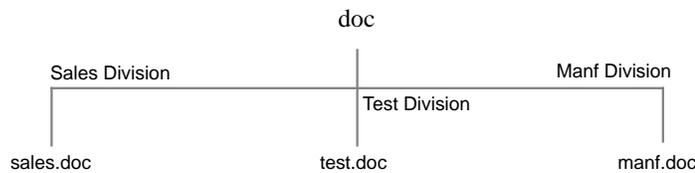
TABLE 1-1 Representation of doc.com Network

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

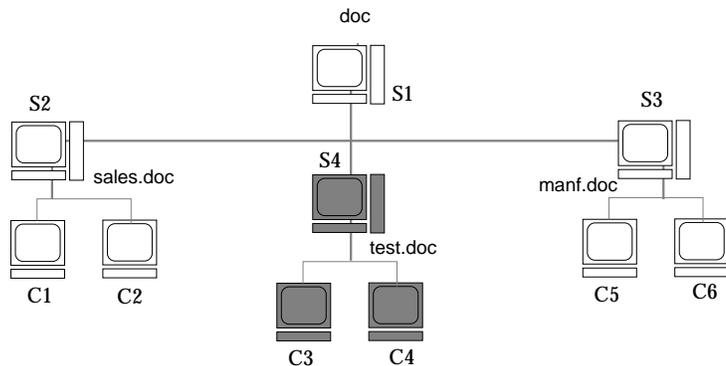
Now assume that you create a third division, Testing, which borrowed some resources from the other two divisions, but did not create a third subnet. The physical network would then no longer parallel the corporate structure:



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 your organization would continue to be accommodated by changes to the “soft” network information structure without reorganizing the “hard” network structure.

Solaris Name Services

The Solaris 2.6 release provides the following name services:

- DNS, the *Domain Name System* (see “DNS” on page 8).
- `/etc` files, the original UNIX naming system (see “`/etc` Files” on page 9).
- NIS, the *Network Information Service* (see “NIS” on page 9).
- NIS+, the *Network Information Service Plus* (see “NIS+” on page 9).
- FNS, the *Federated Naming Service*, supports the use of different autonomous naming systems in a single Solaris environment (see “FNS” on page 10).

Most modern networks use two or more of these services in combination. When more than one service is used, they are coordinated by the `nsswitch.conf` file which is discussed in Chapter 2.

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.

/etc Files

The original host-based UNIX naming system was developed for stand-alone UNIX machines and then adapted for network use. Many old UNIX operating systems and machines still use this system, but it is not well suited for large complex networks.

NIS

The *Network Information Service* (NIS) was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using workstation names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about workstation names and addresses, users, the network itself, and network services. This collection of network *information* is referred to as the *NIS namespace*.

NIS namespace information is stored in NIS 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 (see “NIS Maps” on page 287).

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 *slave* servers. Both master and slave servers use the NIS information retrieval software and both store NIS maps. For more information on NIS Architecture, see “NIS Architecture” on page 282.

See *Administering NIS* for more information about NIS and how to administer it.

NIS+

The *Network Information Service Plus* (NIS+) is similar to NIS but with many more features. NIS+ is not an extension of NIS. It is a new software program.

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. Unlike NIS, the NIS+ name space is dynamic because updates can occur and be put into effect at any time by any authorized user.

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.

See for a more detailed description of NIS+, and for information on using NIS+.

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. FNS conforms to the X/Open federated naming (XFN) specification.

FNS is not a replacement for NIS+, NIS, DNS, or */etc* files. Rather, FNS is implemented on top of these services and allows you to use a set of common names with desktop applications.

See *Administering FNS* for more information about FNS and how to administer it.

The Name Service Switch

This chapter describes the name service switch, what it does, and how clients use it to obtain naming information from one or more sources. You use the name service switch to coordinate usage of different naming services.

- “About the Name Service Switch” on page 11
- “The `nsswitch.conf` Template Files” on page 17
- “DNS and Internet Access” on page 20
- “Adding Compatibility With +/- Syntax” on page 21
- “The Switch File and Password Information” on page 22
- “FNS and the Name Service Switch” on page 22

About the Name Service Switch

The name service switch is a file named `nsswitch.conf`. It controls how a client workstation or application obtains network information. It is used by client applications that call any of the `getXbyY()` interfaces such as:

- `gethostbyname()`
- `getpwuid()`
- `getpwnam()`

The name service switch is often referred to as simply the *switch* or the *switch file*. Each workstation has a switch file in its `/etc` directory. Each line of that file identifies a particular type of network information, such as host, password, and group, followed by one or more sources where the client is to look for that information.

A client can obtain naming information from one or more of the switch's sources. For example, an NIS+ client could obtain its hosts information from an NIS+ table 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 13).

The Solaris 2.6 release software automatically loads an `nsswitch.conf` file into every workstation's `/etc` directory as part of the installation process. Three alternate (template) versions of the switch file are also loaded into `/etc`:

- `/etc/nsswitch.files`
- `/etc/nsswitch.nis`
- `/etc/nsswitch.nisplus`

These three files are alternate default switch files. Each one is designed for a different primary naming service: `/etc` files, NIS, or NIS+. When Solaris 2.6 release software is first installed on a workstation, the installer selects the workstation's default name service: NIS+, NIS, or local files. During installation, the corresponding template file is copied to `nsswitch.conf`. For example, for a workstation client using NIS+, the installation process copies `nsswitch.nisplus` to `nsswitch.conf`. Unless you have an unusual namespace, the default template file as copied to `nsswitch.conf` should be sufficient for normal operation.

No default file is provided for DNS, but you can edit any of these files to use DNS (see "DNS and Internet Access" on page 20).

If you later change a workstation's primary name service, you simply copy the appropriate alternate switch file to `nsswitch.conf`. (See "The `nsswitch.conf` Template Files" on page 17.) You can also change the sources of particular types of network information used by the client by editing the appropriate lines of the `/etc/nsswitch.conf` file. The syntax for doing this is described below, and additional instructions are provided in *Solaris Naming Setup and Configuration Guide*.

Format of the `nsswitch.conf` File

The `nsswitch.conf` file is essentially a list of 15 types of information and the sources that `getXXbyYY()` routines search for that information. The 15 types of information, not necessarily in this order, are:

- `aliases`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `netgroup`

- netmasks
- networks
- passwd (includes shadow information)
- protocols
- publickey
- rpc
- services
- automount
- sendmailvars

Table 2-1 provides a description of the kind of sources that can be listed in the switch file for the information types above.

TABLE 2-1 Switch File Information Sources

Information Sources	Description
files	A file stored in the client's /etc directory. For example, /etc/passwd
nisplus	An NIS+ table. For example, the hosts table.
nis	A NIS map. For example, the hosts map.
compat	Compat can be used for password and group information to support old-style + or - syntax in /etc/passwd, /etc/shadow, and /etc/group files.
dns	Can be used to specify that host information be obtain from DNS.

Search Criteria

Single Source. If an information type has only one source, such as `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.

Multiple Sources. If a table has more than one source for a given information type, the switch directs the routine to start searching for the information in the first source that is listed. If it finds the information, it returns a `success` status message. If it does not find the information in the first source, it tries the next source. The routine will

search through all of the sources until it has found the information it needs, or it is halted by encountering a `return` specification. If all of the listed sources are searched without find the information, the routine stops searching and returns a `non-success` status message.

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. Possible status messages are listed in Table 2-2.

TABLE 2-2 Switch Search Status Messages

Status Message	Meaning of Message
<code>SUCCESS</code>	The requested entry was found in the specified source.
<code>UNAVAIL</code>	The source is not responding or is unavailable. That is, the NIS+ table, or NIS map, or <code>/etc</code> file could not be found or accessed.
<code>NOTFOUND</code>	The source responded with "No such entry." In other words, the table, map, or file was accessed but it did not contain the needed information.
<code>TRYAGAIN</code>	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 2-3.

TABLE 2-3 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*.

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 (or only) source, return with a `NOTFOUND` status.
- `NOTFOUND=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status.
- `TRYAGAIN=continue`. Go to the next `nsswitch.conf` file source and continue searching. If this is the last (or only) source, return with a `NOTFOUND` status.

Because these are the default search criteria, they are assumed. That is, you do not have to explicitly specify them in the switch file. You can change these default search criteria by explicitly specifying some other criteria using the `STATUS=action` syntax show above. For example, the default action for a `NOTFOUND` condition is to continue the search to the next source. To specify that for a particular type of information, such as `networks`, the search is to halt on a `NOTFOUND` condition, you would edit the `networks` line of the switch file to read:

```
networks: nis [NOTFOUND=return] files
```

The `networks: nis [NOTFOUND=return] files` line specifies a nondefault criterion for the `NOTFOUND` status. (Nondefault criteria are delimited by square brackets.)

In this example, the search routine behaves as follows:

- If the `networks` map is available and contains the needed information, the routine returns with a `SUCCESS` status message.
- If the `networksmap` is not available, the routine returns with an `UNAVAIL` status message and by default continues on to search the appropriate `/etc` file.
- If the `networks` map is available and found, but it does not contain the needed information, the routine returns with a `NOTFOUND` message. But instead of continuing on to search the appropriate `/etc` file (the default behavior), the routine stops searching.
- If the `networks` map is busy, the routine returns with an `TRYAGAIN` status message and by default continues on to search the appropriate `/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 switch file's defaults.

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 default values.

Auto_home and Auto_master

The switch search criteria for the `auto_home` and `auto_master` tables and maps is combined into one category called `automount`.

Timezone and the Switch File

The `timezone` table does not use the switch, so it is not included in the switch file's list.

Comments in `nsswitch.conf` Files

Any `nsswitch.conf` file line beginning with a hash character (`#`) is interpreted as a comment line and is 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 not acted upon.

TABLE 2-4 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] file</code>
Partially interpreted line (the <code>files</code> element not interpreted)	<code>hosts: nisplus [NOTFOUND=return] # files</code>

Keyserver and `publickey` Entry in the Switch File

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.

The `nsswitch.conf` Template Files

Three `nsswitch.conf` template files are provided with the Solaris 2.6 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.
- *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.
- *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 that entry in the switch file.

Copy the template file that most closely meets your requirements to `nsswitch.conf` configuration file is `nsswitch.conf` and then modify `nsswitch.conf` as needed. (See the switch chapter of *Solaris Naming 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
```

The Default Switch Template Files

Here are the three switch files supplied with Solaris 2.6 release:

CODE EXAMPLE 2-1 NIS+ Switch File Template (nsswitch.nisplus)

```
#
# /etc/nsswitch.nisplus:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS+ (NIS Version 3) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.

# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nisplus
group: files nisplus
# consult /etc "files" only if nisplus is down.
hosts: nisplus [NOTFOUND=return] files
# Uncomment the following line, and comment out the above, to use
# both DNS and NIS+. You must also set up the /etc/resolv.conf
# file for DNS name server lookup. See resolv.conf(4).
# hosts: nisplus dns [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
sendmailvars: files nisplus
```

CODE EXAMPLE 2-2 NIS Switch File Template

```
#
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
#
# the following two lines obviate the "+" entry in /etc/passwd
# and /etc/group.
passwd: files nis
group: files nis
# consult /etc "files" only if nis is down.
```

(continued)

```

hosts: 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
# for efficient getservbyname() avoid nis
services: files nis
sendmailvars: files

```

CODE EXAMPLE 2-3 Files Switch File Template

```

#
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet"
# transports.
passwd: files
group: files
hosts: files
networks: files
protocols: files
rpc: files
ethers: files
netmasks: files
bootparams: files
publickey: files
# At present there isn't a 'files' backend for netgroup;
# the system will figure it out pretty quickly, and won't use
# netgroups at all.
netgroup: files
automount: files
aliases: files
services: files
sendmailvars: files

```

Default nsswitch.conf File

The default `nsswitch.conf` file that is installed when you install the Solaris operating system for the first time is determined by which name service you select during the Solaris software installation process. When you chose a name service, the switch template file for that service is copied to create the new `nsswitch.conf` file. For example, if you choose NIS+, the `nsswitch.nisplus` file is copied to create a new `nsswitch.conf` file

DNS and Internet Access

The `nsswitch.conf` file also controls DNS forwarding for clients as described in the following subsections. DNS forwarding grants Internet access to clients.

Note - The NIS+ client must have a properly configured `/etc/resolv.conf` file (as described in “DNS Clients and the Resolver” on page 467).

See the switch file chapter of *Solaris Naming 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

DNS forwarding is inherent in the NIS name service. The proper format for the `hosts` line in a NIS-primary switch file to enable DNS forwarding is:

```
hosts: nis [NOTFOUND=return] files
```



Caution - If an NIS client is using the DNS forwarding capability of a NIS-compatible NIS+ server, its `nsswitch.conf` file should *not* have `hosts: nis dns` files as the syntax for the hosts file. This is because DNS forwarding automatically forwards host requests to DNS and this syntax would cause the NIS+ server to forward unsuccessful requests to the DNS servers twice, which would reduce performance. To take best advantage of DNS forwarding, use the default syntax for the `nsswitch.nis` file.

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 provide the same syntax as in the Sun Operating System 4.x release, change the `passwd` and `groups` sources to `compat`.

```
passwd: compat
group: compat
```

This specifies that `/etc` files and NIS maps as indicated by the +/- entries in the files.

Note - Users working on a client machine being served by an NIS+ server running in NIS 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 *Solaris Naming Setup and Configuration Guide* for step by step instructions on adding +/- semantics to an `nsswitch.conf` file.

The Switch File and Password Information

For `passwd` information, `files` should always be the first source searched.

For example, in an NIS+ environment, the `passwd` line of the `nsswitch.conf` file should look like this:

```
passwd: files nisplus
```

In a NIS environment, the `passwd` line of the `nsswitch.conf` file should look like this:

```
passwd: files nis
```



Caution - `files` should be the first source in the `nsswitch.conf` file for `passwd` information. If `files` is not the first source, network security could be weakened and users could encounter log in difficulty.

FNS and the Name Service Switch

See , for more about the Federated Naming Service.

FNS, the Solaris implementation of the XFN API, can also be used to specify which name service a client is to query for naming information. The XFN API is more general in both the X and Y dimensions than the update `getXbyY()` interfaces that use the switch file. For example, it can be used to lookup information on both hosts and users, from both NIS+ and NIS. An application can be a client of either `getXbyY()`, or XFN, or both.

Maintaining Consistency Between FNS and the Switch File

In order to ensure that changes made to namespace data through FNS are always available to clients obtaining namespace information through the switch file, always configure both the switch and FNS to use the same name service.

Namespace Updates

The support for data updates provided by the XFN API is superior to that of the `getXbyY()` interfaces. Most namespaces are composed of data from multiple sources. A groups namespace, for example, might contain information from both the `/etc/group` file and the NIS+ `group.org_dir` object. But the switch file does not provide enough information for an application update routine to identify the source of some particular piece of group data or the source to update.

Because each FNS subordinate namespace comes entirely from a single name service, updates are simple and straightforward because there is no confusion over which name service the update applies to.

This part describes NIS+.

- Chapter 3
- Chapter 4
- Chapter 5
- Chapter 6

Introduction to NIS+

This chapter provides an overview of the *Network Information Service Plus* (NIS+)

- “About NIS+” on page 27
- “What NIS+ Can Do for You” on page 28
- “How NIS+ Differs From NIS” on page 29
- “NIS+ Security” on page 32
- “NIS+ and the Name Service Switch” on page 33
- “Solaris 1.x Releases and NIS-Compatibility Mode” on page 33
- “NIS+ Administration Commands” on page 34
- “NIS+ API” on page 36

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

About 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.

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.

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 32, 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

The *Network Information Service Plus* (NIS+) differs from the *Network Information Service* (NIS) in several ways. NIS+ has many new features, and the terminology it uses for concepts similar to NIS is different. Look in the *Glossary* if you see a term you don't recognize. Table 3-1 gives an overview of the major differences between NIS and NIS+.

TABLE 3-1 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 multi-column tables
Uses no authentication	Uses DES authentication

TABLE 3-1 Differences Between NIS and NIS+ (continued)

NIS	NIS+
Single choice of network information source	Name service switch—lets client choose information source: NIS, NIS+, DNS, or local /etc 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 multi-vendor 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 client-server networks 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.

For example, Figure 3-1, shows a sample company with a parent domain named `doc`, and two subdomains named `sales` and `manf`.

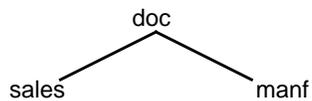


Figure 3-1 Example of Hierarchical Domains

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.

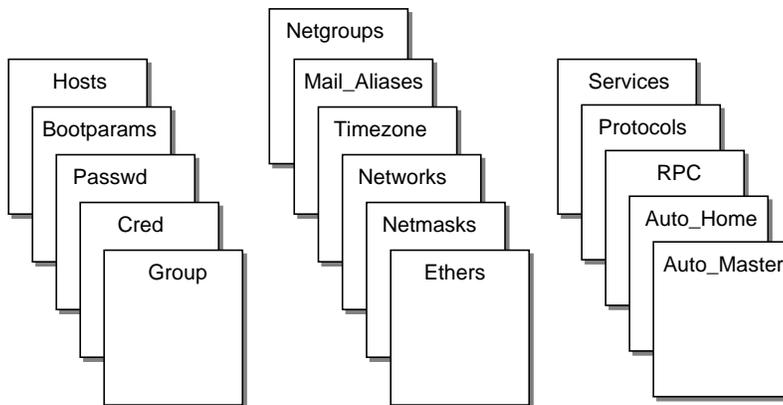
However, NIS+ uses an updating model that is completely different from the one used by NIS. Since at the time NIS was developed, the type of information it would store changed infrequently, NIS was developed with an update model that focused on stability. Its updates are handled manually and, in large 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 4.

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 33). 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, you can search an NIS+ table 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, you can access and manipulate the information in NIS+ tables 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 5.

You can use NIS in conjunction with NIS+ under the following principles and conditions:

- Servers within a domain. While you can have both NIS and NIS+ servers operating in the same domain, doing so is not recommended for long periods. As a general rule, using both services in the same domain should be limited to a relatively short transition period from NIS to NIS+.
- Subdomains. If the master server of your root domain is running NIS+, you can set up subdomains whose servers are all running NIS. (If your root domain master server is running NIS, you cannot have subdomains.)
- Workstations within a domain.
 - If a domain’s servers are running NIS+, individual workstations within that domain can be set up to use either NIS+, NIS, or `/etc` files for their name service information. In order for an NIS+ server to supply the needs of an NIS client, the NIS+ server must be running in NIS-Compatibility mode.
 - If a domain’s servers are running NIS, individual workstations within that domain can be set up to use either NIS or `/etc` files for name services (they cannot use NIS+).

The service a workstation uses for various name services is controlled by the workstation’s `nsswitch.conf` file. This file is called the *switch* file. See Chapter 2 for further 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*. An NIS+ principal can be a process, machine, root, or a user. Valid NIS+ principals possess an 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 6, and

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 2.6 releasebased 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 2.

Solaris 1.x Releases and NIS-Compatibility Mode

NIS+ can be used by workstations running NIS with Solaris 1x or 2x Release software. In other words, machines within an NIS+ domain can have their `nsswitch.conf` files set to `nis` rather than `nisplus`. To access NIS+ service on machines running NIS, you must run the NIS+ servers in *NIS-compatibility mode*.

NIS-compatibility mode enables an NIS+ server running Solaris 2.6 release 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 that an 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*.

Two more differences need to be pointed out. First, 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 *Solaris Naming Setup and Configuration Guide*. Second, 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 6.

NIS+ Administration Commands

NIS+ provides a full set of commands for administering a namespace. Table 3-2, below, summarizes them.

TABLE 3-2 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>nisbackup</code>	Backs up NIS directories.
<code>nis_cachemgr</code>	Starts the NIS+ cache manager on an NIS+ client.
<code>niscat</code>	Displays the contents of NIS+ tables.
<code>nis_checkpoint</code>	Forces service to checkpoint data that has been entered in the log but not checkpointed to disk.
<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>nisclient</code>	Initializes NIS+ principals.

TABLE 3-2 NIS+ Namespace Administration Commands *(continued)*

Command	Description
<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+ tables.
<code>nislog</code>	Displays the contents of NIS+ transaction log.
<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>nis_ping</code>	Forces a replica to update its data from the master server.
<code>nispopulate</code>	Populates the NIS+ tables in a new NIS+ domain.
<code>nisprefadm</code>	Specifies the order in which clients are to seek NIS+ information from NIS+ servers.
<code>nisrestore</code>	Restores previously backed up NIS+ directories and can also be used to quickly bring online new NIS+ replica servers.
<code>nism</code>	Removes NIS+ objects (except directories) from the namespace.
<code>nismdir</code>	Removes NIS+ directories and replicas from the namespace.

TABLE 3-2 NIS+ Namespace Administration Commands *(continued)*

Command	Description
<code>nisserver</code>	Shell script used to set up a new NIS+ server.
<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.
<code>nisstat</code>	Reports statistics and other information about an NIS+ server.
<code>nistbladm</code>	Creates or deletes NIS+ tables, and adds, modifies or deletes entries in an NIS+ table.
<code>nistest</code>	Reports the current state of the NIS+ namespace.
<code>nisupdkeys</code>	Updates the public keys stored in an NIS+ object.
<code>passwd</code>	Changes password information stored in the NIS+ <code>Passwd</code> 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()`)

The NIS+ Namespace

This chapter describes the structure of the NIS+ namespace, the servers that support it, and the clients that use it.

- “NIS+ Files and Directories” on page 39
- “Structure of the NIS+ Namespace” on page 40
- “Directories” on page 42
- “Domains” on page 43
- “Servers” on page 44
- “NIS+ Clients and Principals” on page 47
- “Naming Conventions” on page 52
- “NIS+ Name Expansion” on page 57

NIS+ Files and Directories

Table 4-1 lists the UNIX directories used to store NIS+ files.

TABLE 4-1 Where NIS+ Files are Stored

Directory	Where	Contains
<code>/usr/bin</code>	All machines	NIS+ user commands
<code>/usr/lib/nis</code>	All machines	NIS+ administrator commands

TABLE 4-1 Where NIS+ Files are Stored (continued)

Directory	Where	Contains
/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 Release 2.4 and earlier versions, the `/var/nis` directory contained two files named `hostname.dict` and `hostname.log`. It also contained a subdirectory named `/var/nis/hostname`. Starting with Solaris Release 2.5, the two files were named `trans.log` and `data.dict`, and the subdirectory was named `/var/nis/data`. The *content* of the files was also changed and they are not backward compatible with Solaris Release 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris Release 2.4 patterns, the files will not work with *either* the Solaris 2.4 Release or the current version of `rpc.nisd`. Therefore, you should not rename either the directories or the files.

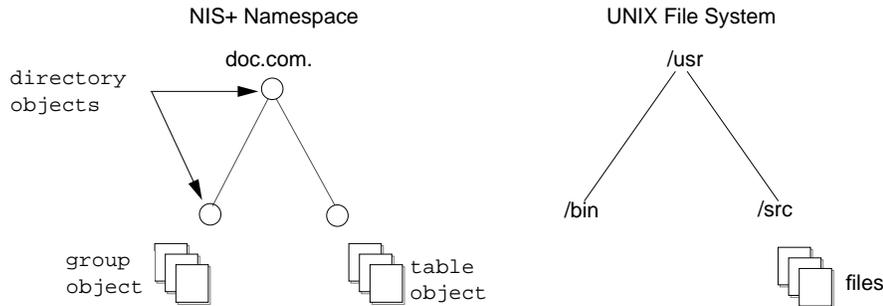
Note - With the Solaris 2.6 release, the NIS+ data dictionary (`/var/nis/data.dict`) is now machine independent. This allows you to easily change the name of an NIS+ server. You can also now use the NIS+ backup and restore capabilities to transfer NIS+ data from one server to another. See .

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:

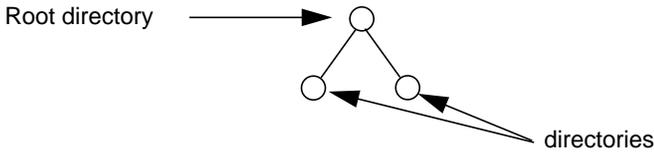


Although an NIS+ namespace resembles a UNIX file system, it has five important differences:

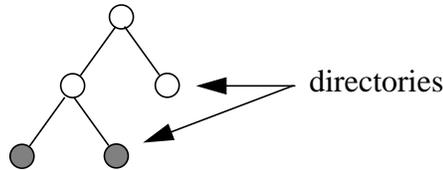
- Although both use directories, the other objects in an NIS+ namespace are tables and groups, not files.
- The NIS+ namespace is administered only through NIS+ administration commands or graphical user interfaces designed for that purpose, such as the Solstice AdminSuite tools; it cannot be administered 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 (`doc.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.doc.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”:



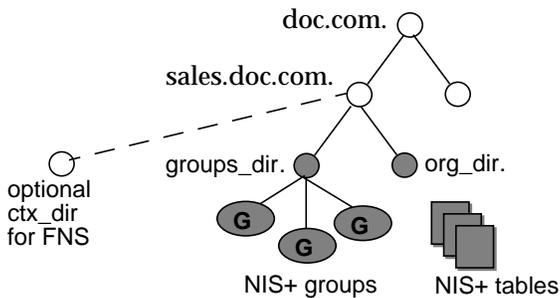
A namespace can have several levels of directories:



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. Each NIS+ domain-level directory contains the following sub-directories:

- `groups_dir`. Stores NIS+ group information.
- `org_dir`. Stores NIS+ system tables.
- `ctx_dir`. This directory is only present if you are using FNS.

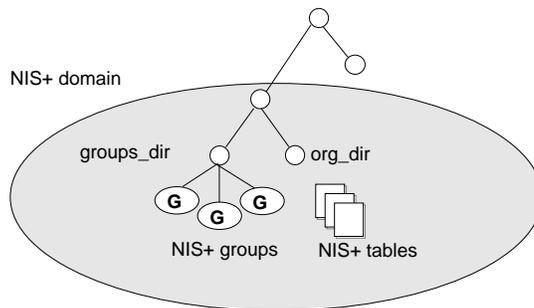


Technically, you can arrange directories, tables, and groups into any structure that you like. However, NIS+ directories, tables, and groups in a namespace are normally

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, suppose the DOC company has Sales and Manufacturing divisions. To support those divisions, its NIS+ namespace would most likely be arranged into three major directory groups, with a structure that looked like this:

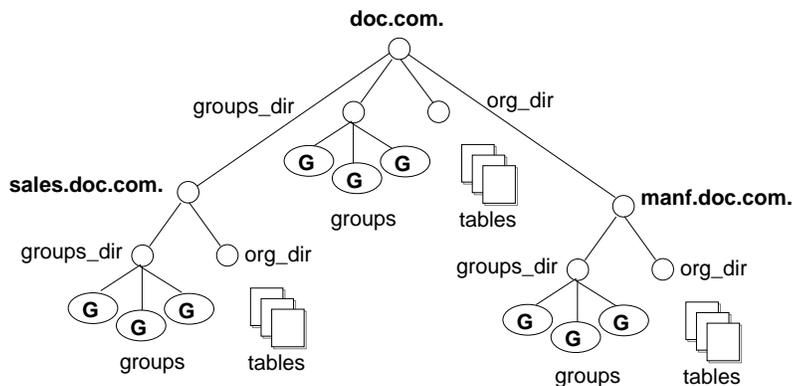


Figure 4-1 Example NIS+ Directory Structure

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

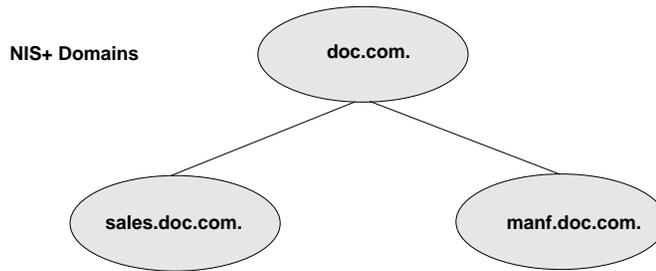
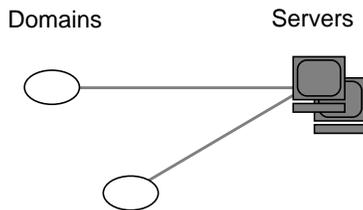


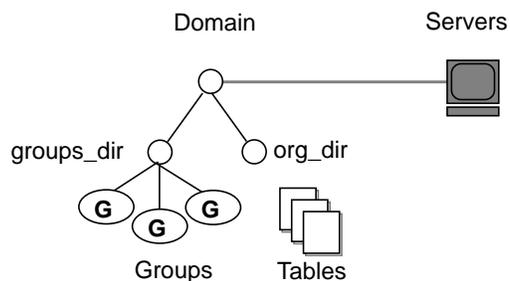
Figure 4-2 Example NIS+ Domains

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 2.6 release 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 Release 2 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.

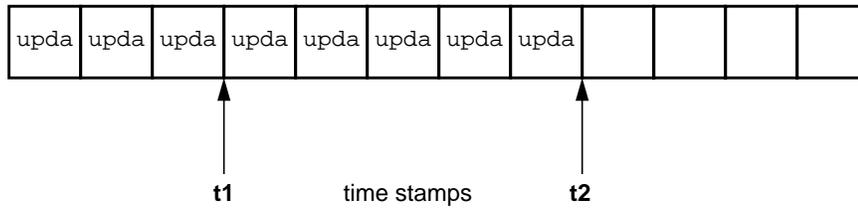
This arrangement has two benefits. First, it avoids conflicts between tables because only one set of master tables exists; the tables stored by the replicas are only copies of the masters. Second, it makes the NIS+ service much more *available*. If either the master or a replica is down, another server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

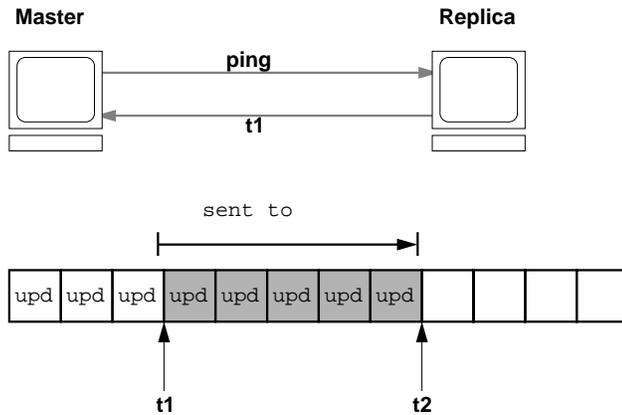
The transaction log is used by a master server to store changes to the namespace until they can be propagated to replicas. A transaction log has two primary components: updates and time stamps.

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.)

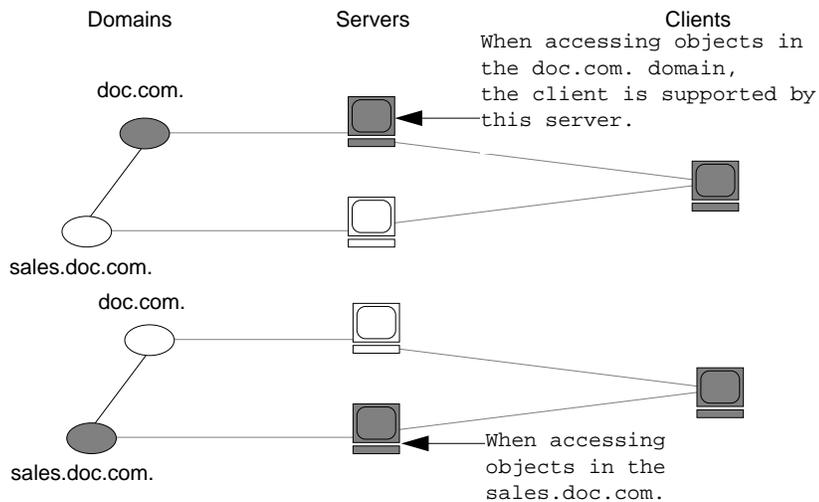
An NIS+ client can access any part of the namespace, subject to security constraints. In other words, if it has been authenticated and has been granted the proper permissions, it can access information or objects in any domain in the namespace.

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:

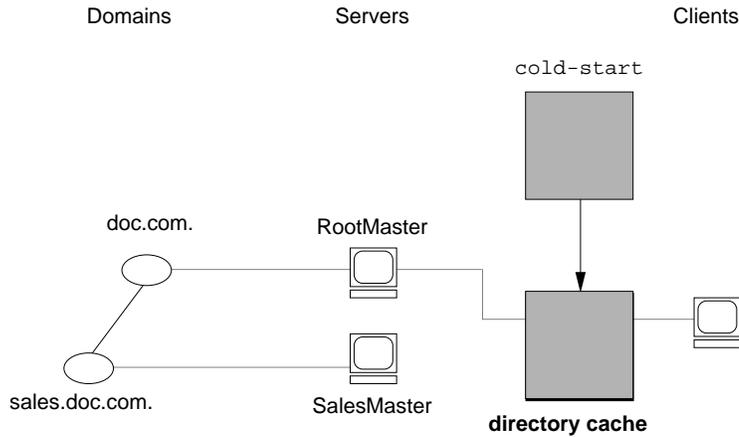


How does the client know which server that is? By trial and error. Beginning with its home server, the client tries first one server, then another, until it finds the right one. When a server cannot answer the client's request, it sends the client information to help locate the right server. Over time, 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.

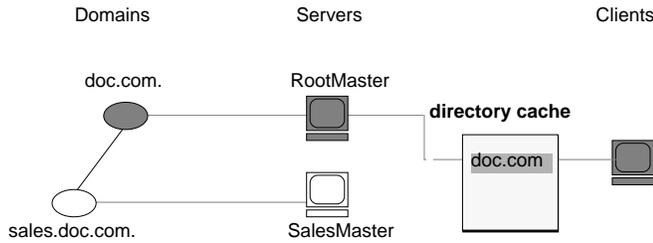


By storing a copy of the namespace's directory objects in its directory cache, a client can know which servers support which domains. (To view the contents of a client's cache, use the `nisshowcache` command, described in .) Here is a simplified example:

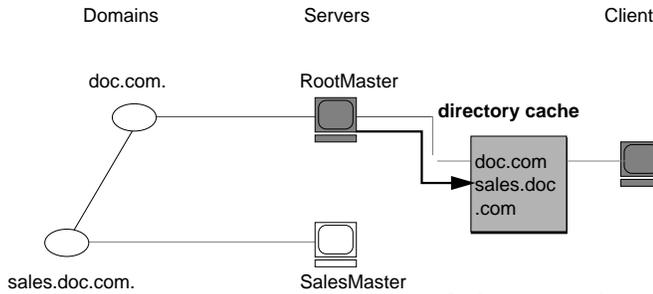
Domain Name and Directory Name are the same	Supporting Server	IP Address
<code>doc.com.</code>	<code>rootmaster</code>	<code>123.45.6.77</code>
<code>sales.doc.com.</code>	<code>salesmaster</code>	<code>123.45.6.66</code>
<code>manf.doc.com.</code>	<code>manfmaster</code>	<code>123.45.6.37</code>
<code>int.sales.doc.com.</code>	<code>Intlsalesmaster</code>	<code>111.22.3.7</code>

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 205. 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.

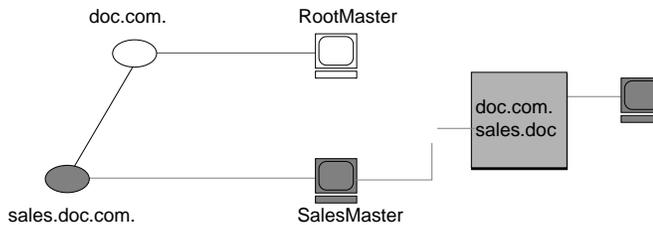
How does the directory cache accumulate these directory objects? As mentioned above, the `cold-start` file provides the first entry in the cache. Therefore, when the client sends its first request, the request goes to the server specified by the `cold-start` file. If the request is for access to the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



If the request is for access to another domain (for example, `sales.doc.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.



If the server does not support the domain the client is trying to access, it sends



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 2

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

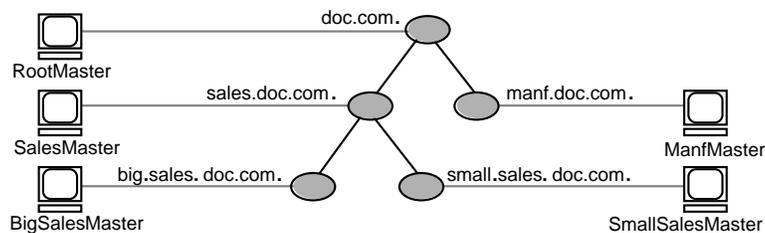
An NIS+ server is also an NIS+ client. In fact, before you can set up a workstation as a server (as described in Part 2), you must initialize it as a client. The only exception is the root master server, which has its own unique setup process.

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:

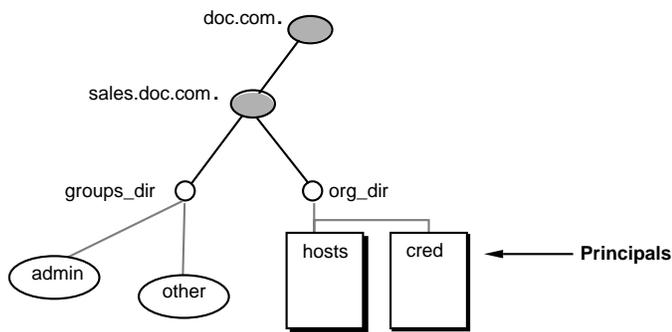
Server	Supports	Belongs to
RootMaster	doc.com.	doc.com.
SalesMaster	sales.doc.com.	doc.com.
IntlSalesMaster	intl.sales.doc.com.	sales.doc.com.
ManfMaster	manf.doc.com.	doc.com.

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 57.

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 4-3.

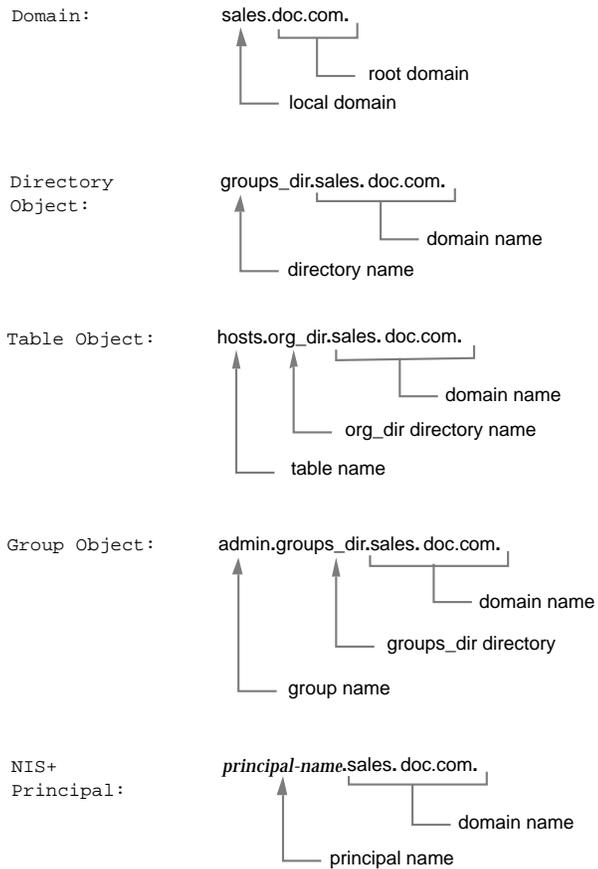


Figure 4-3 Fully qualified Names of Namespace Components

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:

`doc.com.` (root domain)

`sales.doc.com.` (subdomain)

`intl.sales.doc.com.` (a third level subdomain)

The first line above shows the name of the root domain. The root domain must always have at least two elements (labels) and must end in a dot. The last (right most) label may be anything you want, but in order to maintain Internet compatibility, the last element must be either an Internet organizational name (as shown in Table 4-2), or a two or three character geographic identifier such as `.jp.` for Japan.

TABLE 4-2 Internet Organizational Domains

Domain	Purpose
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
mil	Military groups
net	Major network support centers
org	Nonprofit organizations and others
int	International organizations

The second and third lines above 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.doc.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 57).

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:

```
admin.groups_dir.doc.com.  
admin.groups_dir.doc.com.  
admin.groups_dir.sales.doc.com.  
admin.groups_dir.sales.doc.com.  
hosts.org_dir.doc.com.  
hosts.org_dir.doc.com.  
hosts.org_dir.sales.doc.com.  
hosts.org_dir.sales.doc.com.
```

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,...],tablename
```

Column is the name of the table column. *Value* is the actual value of that column. *Tablename* 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.doc.com.  
[addr=129.44.2.2,name=elm],hosts.org_dir.sales.doc.com.  
[addr=129.44.2.3,name=oak],hosts.org_dir.sales.doc.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. Dots are only used as part of a fully qualified host name to identify the domain components. For example, `myco-2.sales.doc.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:

TABLE 4-3 NIS+ Principal Names

<code>olivia.sales.doc.com.</code>	NIS+ principal name
<code>unix.olivia@sales.doc.com</code>	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 56 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.doc.com. domain).

```
mydir —————> mydir.software.big.sales.doc.com.  
                   expands into mydir.big.sales.doc.com.  
                                mydir.sales.doc.com.  
                                mydir.doc.com.
```

```
hosts.org_dir —————> hosts.org_dir Software.big.sales.doc.com.  
                   expands into hosts.org_dir.big.sales.doc.com.  
                                hosts.org_dir.sales.doc.com.  
                                hosts.org_dir.doc.com.
```

NIS_PATH Environment Variable

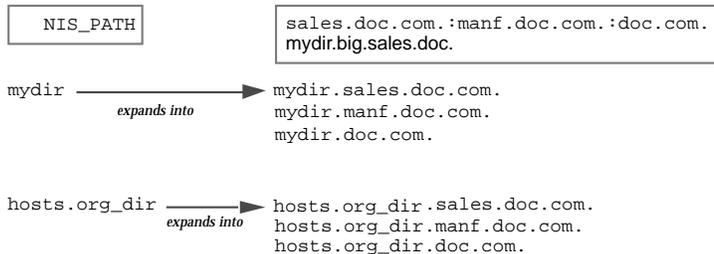
You can change or augment the list of directories NIS+ searches through by changing the value of the environment variable NIS_PATH. NIS_PATH accepts a list of directory names separated by colons:

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

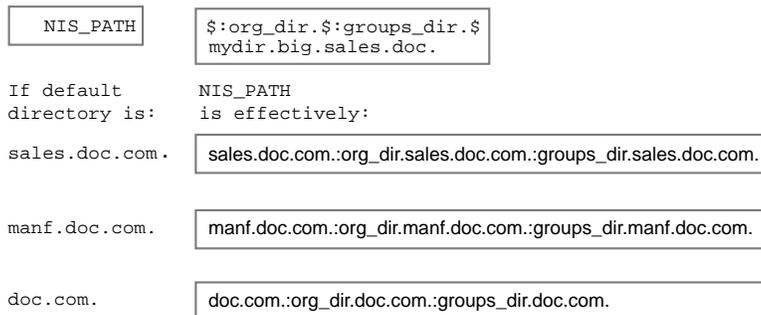
or

```
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

This chapter describes the structure of NIS+ tables and provides a brief overview of how they can be set up.

- “NIS+ Table Structure” on page 59
- “Ways to Set Up Tables” on page 64

NIS+ Table Structure

NIS+ stores a wide variety of network information in tables. 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 5-1 lists the preconfigured NIS+ tables.

TABLE 5-1 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 5-1 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

Because it contains only information related to NIS+ security, the Cred table, is described in Chapter 7.

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.

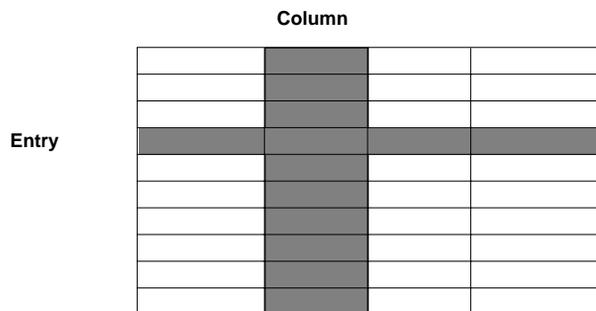
Since this information is generated from such a wide field of operations, much of it is beyond the scope of this manual. However, as a convenience, Appendix C, summarizes the information contained in each column of the tables, providing details only when necessary to keep things from getting confusing,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

Note - 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. When creating additional automount maps to supplement `auto_master` (which is created for you), the column names must be `key` and `value`. For more information about the automounter consult books about the automounter or books that describe the NFS file system.

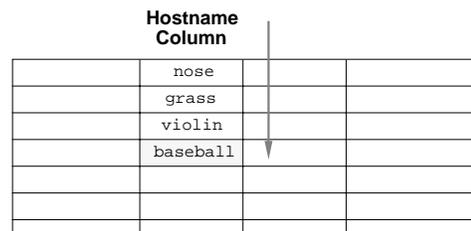
Note - As a naming service, NIS+ tables are designed to store references to objects, not the objects themselves. For this reason, NIS+ does not support tables with large entries. If a table contains excessively large entries, `rpc.nisd` may fail.

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”):



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`.



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 9.

Search Paths

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

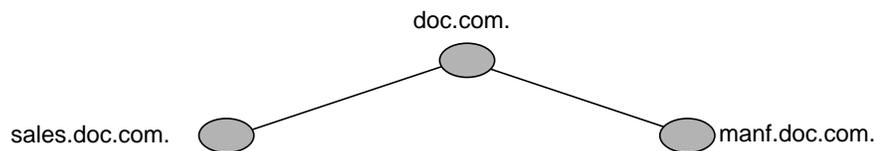
If a client in one domain tries to find information that is stored in another domain, it has to provide a fully qualified name. As described in “NIS+ Name Expansion” on page 57 if the `NIS_PATH` environment variable is set up properly, the NIS+ service will do this automatically.

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 two domains have the following contents:

TABLE 5-2 Example Hosts Table

sales.doc.com.		manf.doc.com.	
127.0.0.1	localhost	127.0.0.1	localhost
111.22.3.22	luna	123.45.6.1	sirius
111.22.3.24	phoebus	123.45.6.112	rigel
111.22.3.25	europa	123.45.6.90	antares
111.22.3.27	ganymede	123.45.6.101	polaris
111.22.3.28	mailhost	123.45.6.79	mailhost

Assume now that a user logged onto the luna machine in the sales.doc.com. domain wants to log in remotely to another client. Without providing a fully qualified name, that user can only remotely log on to five workstations: localhost, phoebus, europa, ganymede, and the mailhost.

Now assume that the search path of the hosts table in the sales.doc.com. domain listed the hosts table from the manf.doc.com. domain:

```
hosts.org_dir.manf.doc.com.
```

Now a user in the sales.doc.com. domain can enter something like rlogin sirius, and the NIS+ server will find it. It will first look for sirius in the local domain, but when it does not find a match, it will look in the manf.doc.com. domain. How does the client know how to find the manf.doc.com. domain? As described in Chapter 4, 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 4.

There is a slight drawback, though, to specifying a search path. If the user were to enter an incorrect name, such as rlogin luba (rather than “luna”), 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.doc.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 210.

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 non-system tables (optional)
4. Populating the tables with information

As described in Chapter 4, 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 `nissserver` script. The `nissserver` script creates the appropriate directories and a full set of system tables. Running the `nissserver` 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 `nissserver` script and the `nissetup` and `nismkdir` utilities are described in *Solaris Naming Setup and Configuration Guide*. Additional information on the `nismkdir` command can be found in "The `nismkdir` Command" on page 189.

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. 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 easier way is to use the `nissserver` script.

To create a non-system table—that is, a table that has not been preconfigured by NIS+—use the `nistbladm` command. (Note that if you are creating additional

automount maps, the first column must be named `key` and the second column named `value`.)

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 describes the NIS+ security system and how it affects the entire NIS+ namespace.

- “Solaris Security—Overview” on page 67
- “NIS+ Security—Overview” on page 69
- “NIS+ Authentication and Credentials—Introduction” on page 72
- “NIS+ Authorization and Access—Introduction ” on page 74
- “The NIS+ Administrator” on page 78
- “NIS+ Password, Credential, and Key Commands ” on page 79

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 6-1 for a schematic representation of this system.)

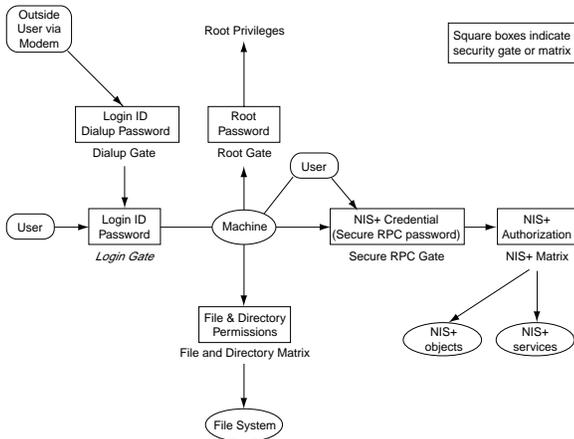


Figure 6-1 Solaris Security Gates and Filters

As you can see in Figure 6-1, 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.
- *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.

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

To enter the Secure RPC gate requires presentation of a Secure RPC password. (In some contexts *Secure RPC passwords* have been referred to as *network passwords*.) 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 91 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 an NIS+ service.

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 7.

- *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 9.

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.
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. This is similar in concept to the standard UNIX file and directory permissions system. (See “Authorization Classes” on page 75 “Authorization Classes” on page 137 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 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 6-2 Figure 6-2 details this process:

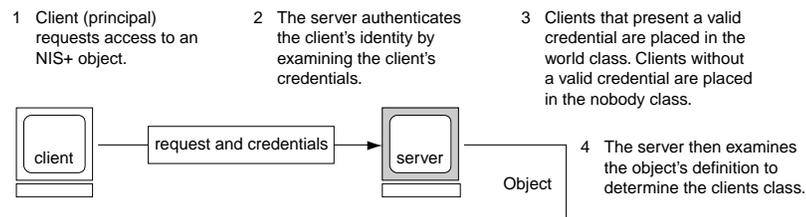


Figure 6-2 Summary of the NIS+ Security Process

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.

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.

NIS+ Security Levels

NIS+ servers operate at one of two security levels. These levels determine the type of credential principals that 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 6-1 Table 6-1 on page 133.

TABLE 6-1 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 `nispaswd` command to change your password. However, `nispaswd` 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.) Starting with Solaris Release 2.5, 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 an NIS+ service or access to an 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 not logged in and then accessing NIS+ objects with the second user's NIS+ access privileges.

Once a server authenticates a principal, the principal is placed in one of four authorization classes. The server then checks the NIS+ object that the principal wants to access to see what activities that class of principal is authorized to perform. (See "NIS+ Authorization and Access—Introduction" on page 74 "NIS+ Authorization and Access—Introduction" on page 136 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

Note - 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 an NIS+ principal, it is the DES credential that NIS+ is validating.

Each time a principal requests an NIS+ service or access to an 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 an NIS+ administrator, as explained in Chapter 7, “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.

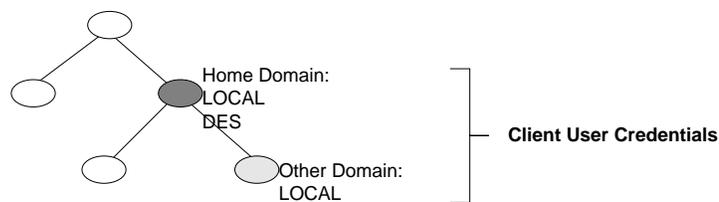


Figure 6-3 Credentials and Domains

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 credentials, 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 6-2 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” on page 75 below for details.)
- *Access rights.* There are four types of access rights (permissions): create, destroy, modify, and read. (See “NIS+ Access Rights” on page 78 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. (In this context, *group* refers to NIS+ groups, not UNIX or net groups.)
- **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.)
- **Nobody.** Everyone belongs to the nobody class even those who are not authenticated.

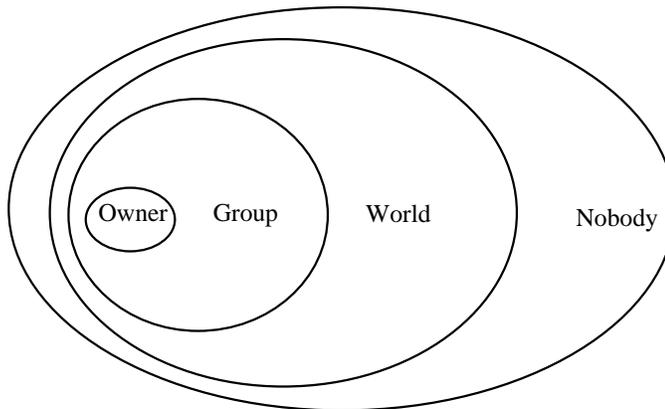


Figure 6-4 Authorization Classes

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 127).
- 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 139).

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

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.

Note - Information about NIS+ groups is not stored in the NIS+ `group` table. The `group` table stores information about UNIX groups. Information about NIS+ groups is stored in the appropriate `groups_dir` directory object.

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

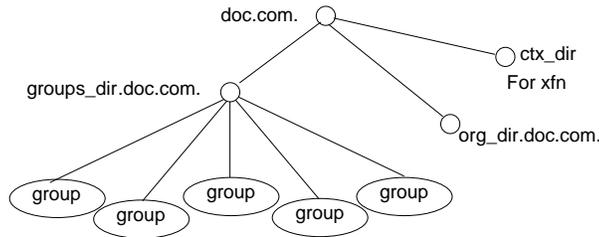


Figure 6-5 NIS+ Directory Structure

Instructions for administering NIS+ groups are provided in Chapter 11.

The World Class

The world class contains all NIS+ principals that are authenticated by NIS+. In other words, the world class includes 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, the nobody class includes everyone 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, an entry (row) 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. Within a table, individual columns have the same owner and group as the table 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 an NIS+ directory object, you can create new tables within that directory. If you have create rights to an 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 78 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 an NIS+ Principal” on page 107 for more information.
- `keylogin`. Decrypts and stores a principal’s secret key with the `keyserv`.
- `keylogout`. Deletes stored secret key from `keyserv`.
- `keyserv`. Enables the server for storing private encryption keys. See “Keylogin” on page 106 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 95 and “Administering NIS+ Credential Information” on page 103 for more information.
- `nisupdkeys`. Updates public keys in directory objects. See “Updating Public Keys” on page 112 for more information.
- `passwd`. Changes and administers principal’s password. See Chapter 10 for more information.

PART III Administering NIS+

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

- Chapter 7
- Chapter 8
- Chapter 9
- Chapter 10
- Chapter 11
- Chapter 12
- Chapter 13
- Chapter 14
- Chapter 15
- Chapter 16

Administering NIS+ Credentials

This chapter describes NIS+ credentials and how to administer them.

- “How Credentials Work” on page 84
- “Credential versus Credential Information” on page 84
- “Authentication Components” on page 85
- “How Principals are Authenticated” on page 85
- “The DES Credential in Detail” on page 89
- “Where Credential-Related Information Is Stored” on page 92
- “The cred Table in Detail” on page 94
- “Creating Credential Information” on page 95
- “The nisaddcred Command” on page 96
- “How nisaddcred Creates Credential Information” on page 97
- “The Secure RPC Netname and NIS+ Principal Name” on page 98
- “Creating Credential Information for the Administrator” on page 99
- “Creating Credential Information for NIS+ Principals” on page 99
- “Updating Your Own Credential Information” on page 103
- “Removing Credential Information” on page 104

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

NIS+ Credentials

NIS+ credentials are used to identify NIS+ users. 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, for this information.)

For a complete description of NIS+ credential-related commands and their syntax and options, see the NIS+ man pages.

How Credentials Work

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

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 6, 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 89 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 89 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 an NIS+ principal makes a request for an NIS+ service or access to an 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 *Solaris Naming Setup and Configuration Guide*. This section describes how to create client information using the NIS+ command set.

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 99.
- Create server public keys. (See "Updating Public Keys" on page 112.)

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 91.

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 an NIS+ principal requests access to an NIS+ object, the NIS+ software performs a multistage 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 it needs.
 - 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.
 - 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.
 - 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.
 - 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.
16. If the time stamp is within the window limit, and greater than the previous request from that principal, the server accepts the request.
17. The server then complies with the request and stores the time stamp from this principal as the most recently received and acted on request.
18. 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.
19. 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" on page 89below).
- A *verification* field (see "DES Credential Verification Field" on page 90, below).

DES Credential Secure RPC Netname

- *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).

Note - Remember that an NIS+ principal name *always* has a trailing dot, and a Secure RPC netname *never* has a trailing dot.

TABLE 7-1 Secure RPC Netname Format

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

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 in "Request Phase—Detailed Description" on page 87
- The encrypted time stamp
- The time window

How the DES Credential Is Generated

See Figure 7-2.

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 91 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.

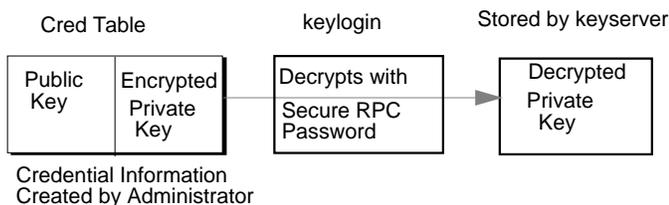


Figure 7-1 `keylogin` Generates a Principal's Private Key

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:

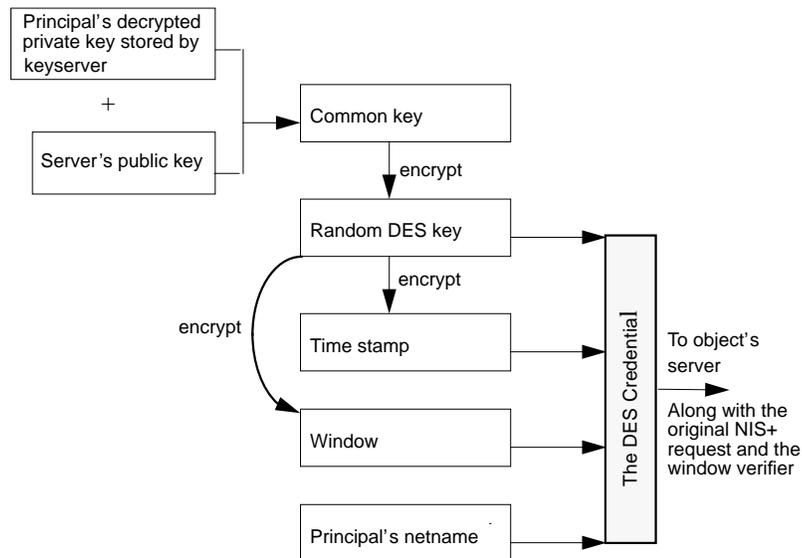


Figure 7-2 Creating the DES Credential

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.

Note - 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 a password. (See "Keylogin" on page 106.)

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 an NIS+ Principal" on page 107.

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 112 for information on using the `nisupdkeys` command and "Stale and Outdated Credential Information" on page 520 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. lists all the objects, tables, and files that store credential-related information and how to reset it.

TABLE 7-2 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_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. On a client workstation, first remove the <code>NIS_COLD_START</code> and <code>NIS_SHARED_DIRCACHE</code> 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 workstation's <code>NIS_COLD_START</code> file.

TABLE 7-2 Where Credential-Related Information Is Stored (continued)

Item	Stores	To Reset or Change
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 must have its own *cred* table. Never 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 13.

The *cred* table as shown in Table 7-3 has five columns:

TABLE 7-3 cred Table Credential Information

	NIS+ Principal Name	Authentication Type	Authentication Name	Public Data	Private Data
Column Name	cname	auth_type	auth_name	public_data	private_data
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 *Solaris Naming 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 *Solaris Naming 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, use:

```
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 7-4 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.	"Listing the Object Properties of a Directory" on page 186
<code>nismatch-</code>	When run on the cred table, displays credential information for <i>principal</i> .	"The <code>nismatch</code> and <code>nisgrep</code> Commands " on page 229

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@doc.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 91.)

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`:

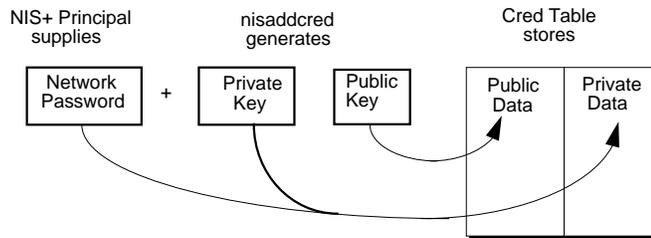


Figure 7-3 How `nisaddcred` Creates a Principal’s Keys

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 99.

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

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 `doc.com.` domain, so this example enters her credential information from a principal machine of that domain:

```
client# nisaddcred -p 11177 -P morena.doc.com. local
client# nisaddcred -p unix.11177@sales.doc.com \
-P morena.doc.com. des
Adding key pair for unix.11177@sales.doc.com
(morena.doc.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 7-5 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, whose login ID is `eiji`, belongs to the root domain, so you would enter his credential information from the root master server which is named `rootmaster`.

TABLE 7-5 Creating Administrator Credentials: Command Summary

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

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

```
rootmaster# nisaddcred -p unix.starshine1@doc.com -
P starshine1.doc.com. des
Adding key pair for unix.starshine1@doc.com
(starshine1.doc.com.).
Enter starshine1.doc.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.doc.com`. The first example removes both types of credential information from her home domain (`doc.com`), the second removes her LOCAL credential information from the `sales.doc.com` domain. Note how they are each entered from the appropriate domain's master servers.

```
rootmaster# nisaddcred -r morena.doc.com.  
salesmaster# nisaddcred -r morena.doc.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 13.

```
rootmaster# nismatch morena.doc.com. cred.org_dir  
salesmaster# nismatch morena.doc.com. cred.org_dir
```

Administering NIS+ Keys

This chapter describes NIS+ keys and how to administer them.

- “Keylogin” on page 106
- “Changing Keys for an NIS+ Principal” on page 107
- “Updating Public Keys” on page 112
- “The `nisupdkeys` Command” on page 112
- “Updating Public Keys Arguments and Examples ” on page 112
- “Updating IP Addresses ” on page 114

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

NIS+ Keys

NIS+ keys are used to encrypt NIS+ related information.

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 6, for this information).

For a complete description of NIS+ key-related commands and their syntax and options, see the NIS+ man pages. (The `nisaddcred` command also performs some key-related operations. See “The `nisaddcred` Command” on page 96 for more information.)

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.

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.)

Note - In this context, *network password* is sometimes used as a synonym for *Secure RPC 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 an NIS+ Principal" on page 107).

Changing Keys for an 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 an NIS+ environment, when you change your login password with any of the current administration tools or the `passwd` (or `nispaswd`) 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 8-1 which illustrates how to perform a `keylogin` and `chkey` for a principal user:

TABLE 8-1 Re-encrypting Your Private Key : Command Summary

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

Changing the Keys

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

Note - Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information " on page 114

Changing Root Keys From Root

Table 8-2 shows how to change the keys for the root master server from the root master (as root):

TABLE 8-2 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>	rootmaster# ps -e grep rpc.nisd
Kill the NIS+ daemon	rootmaster# kill <i>pid</i>
Restart NIS+ daemon with no security	rootmaster# rpc.nisd -S0
Perform a keylogout (previous keylogin is now out of date).	rootmaster# keylogout -f
Update the keys in the directories served by the master	rootmaster# nisupdkeys <i>dirs</i>
Find the Process ID of <code>rpc.nisd</code>	rootmaster# ps -e grep rpc.nisd
Kill the NIS+ daemon	rootmaster# kill <i>pid</i>
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 8-2, `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 8-2 are necessary to successfully update all the objects.

Note - Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 114

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 8-3 Remotely Changing Root Master Keys: Command Summary

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

Where:

- *principal* is the root machine's Secure RPC netname. For example: `unix.rootmaster@doc.com` (no dot at the end).
- *nis-principal* is the root machine's NIS+ principal name. For example, `rootmaster.doc.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.

Note - Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information" on page 114

Changing the Keys of a Root Replica from the Replica

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

```
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.

Note - Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information " on page 114

Changing the Keys of a Nonroot Server

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

```
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.

Note - Whenever you change a server's keys, you must also update the key information of all the clients in that domain as explained in "Updating Client Key Information " on page 114

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 an 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 8-4 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.
-H <i>servername</i>	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.
-s -H <i>servername</i>	Updates the keys of all the directory objects served by the named server.
-C	Clears the keys.

Table 8-5 gives an example of updating a public key:

TABLE 8-5 Updating a Public Key: Command Examples

Tasks	Commands
Update all keys of all servers of the current domain (<i>doc.com</i>).	rootmaster# /usr/lib/nis/nisupdkeys Fetch Public key for server rootmaster.doc.com. netname='unix.rootmaster@doc.com' Updating rootmaster.doc.com.'s public key. Public key: <i>public-key</i>
Update keys of all servers supporting the <i>sales.doc.com</i> domain directory object.	salesmaster# nisupdkeys sales.doc.com (Screen notices not shown)
Update keys for a server named <i>master7</i> in all the directories that store them.	rootmaster# nisupdkeys -H master7
Clear the keys stored by the <i>sales.doc.com</i> directory object.	rootmaster# nisupdkeys -C sales.doc.com
Clear the keys for the current domain directory object for the server named <i>master7</i> .	rootmaster# nisupdkeys -C -H master7

Updating IP Addresses

If you change a server's IP address, or add additional addresses, 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
```

Updating Client Key Information

Whenever you change any server's keys, you must update all of the clients as well. Remember, that all NIS+ servers are also NIS+ clients, so if you update the keys on one server, you must update key information on all other machines in the domain regardless of whether or not they are NIS+ servers or ordinary clients.

There are three ways to update client key information:

- The easiest way to update an individual client's key information is by running the `nisclient` script on the client as described in *Solaris Naming Setup and Configuration Guide*.
- Another way to update an individual client's key information is by running the `nisinit` command on the client as described in "Initializing a Client" on page 197.
- You can globally update client key information for all the machines in a domain by shortening the Time To Live value of the domain's directory object as explained in "Globally Updating Client Key Information" on page 114.

Globally Updating Client Key Information

After changing a server's keys, you can globally update client key information for all the machines in a domain by:

1. Use the `nischt1` command to reduce the Time To Live (TTL) value of the domain's directory object so that the value expires almost immediately.

For example, if you have changed the keys for a server in the `sales.doc.com` domain, to reduce the directory's TTL value to one minute you would enter:

```
client% nischttl 60 sales.doc.com.
```

2. **When the directory's TTL value expires, the cache manager expires the entry and then obtains the new, updated information for clients.**
3. **Once the directory object's TTL value has expired, reset the directory object's TTL to its default value.**

For example, to reset the TTL value to 12 hours for the `sales.doc.com` domain's directory object, you would enter:

```
client% nischttl 12h sales.doc.com.
```

See "The `nischttl` Command" on page 205 for more information on working with TTL values.

Administering NIS+ Access Rights

This chapter describes NIS+ access rights and how to administer them.

- “Introduction to Authorization and Access Rights” on page 118
- “Concatenation of Access Rights” on page 119
- “How Access Rights Are Assigned and Changed” on page 120
- “Table, Column, and Entry Security” on page 120
- “Where Access Rights Are Stored” on page 125
- “Viewing an NIS+ Object’s Access Rights” on page 125
- “Default Access Rights” on page 126
- “How a Server Grants Access Rights to Tables” on page 127
- “Specifying Access Rights in Commands” on page 127
- “Displaying NIS+ Defaults—The `nisdefaults` Command” on page 131
- “Setting Default Security Values” on page 133
- “Specifying Nondefault Security Values at Creation Time” on page 135
- “Changing Object and Entry Access Rights” on page 135
- “Specifying Column Access Rights” on page 137
- “Changing Ownership of Objects and Entries” on page 139
- “Changing an Object or Entry’s Group” on page 140

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

NIS+ Access Rights

NIS+ access rights determine what operations NIS+ users can perform and what information they have access to. 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 6, for this information).

For a complete description of NIS+ access-related commands and their syntax and options, see the NIS+ man pages.

Introduction to Authorization and Access Rights

See “NIS+ Authorization and Access—Introduction ” on page 74 and Chapter 6” 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 75, 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 78, 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 126 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 131 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 135 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
- You can specify access rights to each *entry* (row) by itself.
- You can specify access rights to each table *column* individually,

A *field* is the intersection between a column and an entry (row). All data values are entered in fields.

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:

- *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.

Note - Remember that authorization classes concatenate. A higher class gets the rights assigned to lower classes. See “Concatenation of Access Rights” on page 119

- *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.

- *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. Tables and entries can 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 9-1 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 9-2 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 9-3 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 9-4). 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 9-4). Neither the world nor the group classes could read any cells marked *NP* (for Not Permitted).

TABLE 9-4 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 9-5:

TABLE 9-5 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:

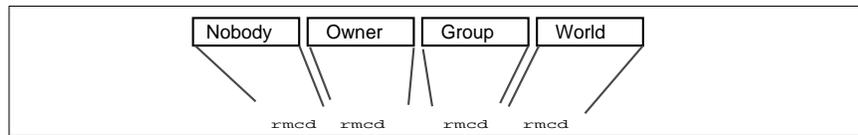


Figure 9-1 Access Rights Display

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: I

TABLE 9-6 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 assume an NIS+ environment running at security level 2 (the default level).

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 9-7 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 9-8 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 9-9 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 9-10 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.

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

For owner

principalname

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 9-11 Object and Table Entry—Examples

Type	Example
Object	hosts.org_dir.sales.doc.com.
Table entry	'[uid=33555],passwd.org_dir.Eng.doc.com.'
Two-value table entry	'[name=sales,gid=2],group.org_dir.doc.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 210 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 9-12 The Seven NIS+ Default Values and `nisdefaults` Options

Default	Option	From	Description
Domain	<code>-d</code>	<code>/etc/defaultdomain</code>	Displays the home domain of the workstation from which the command was entered.
Group	<code>-g</code>	<code>NIS_GROUP</code> environment variable	Displays the group that would be assigned to the next object created from this shell.
Host	<code>-h</code>	<code>uname -n</code>	Displays the workstation’s host name.

TABLE 9-12 The Seven NIS+ Default Values and `nisdefaults` Options (continued)

Default	Option	From	Description
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	NIS_DEFAULTS environment variable	Displays the access rights that will be assigned to the next object or entry created from this shell. Format: <code>----rmcdr---r---</code>
Search path	-s	NIS_PATH 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 NIS_PATH environment variable if it is set.
Time-to-live	-t	NIS_DEFAULTS 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.doc.com.
Domain Name : doc.com.
Host Name : rootmaster.doc.com.
Group Name : salesboss
Access Rights : ----rmcdr---r---
Time to live : 12:00:00:00:00
Search Path : doc.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---
doc.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+rmcd
```

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 131.

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 C-shell or `$NIS_DEFAULTS=`, `export` for Bourne and Korn shells) with the following arguments:

- `access=right`, where *right* are the access rights using the formats described in “Specifying Access Rights in Commands” on page 127.
- `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 9-13 shows some examples:

TABLE 9-13 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>doc.com</code> .	<code>client% setenv NIS_DEFAULTS owner=abe.doc.com.</code>
This command combines the first two examples on one code line.	<code>client% setenv NIS_DEFAULTS access=o+r:owner=abe.doc.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 127.

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.doc.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.doc.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.doc.com` directory object you would type:

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

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

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

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.doc.com` directory object you would type:

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

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

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

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 210. 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
 - 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 127.
- ... 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 `doc.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.doc.com.
```

For more information about the `nistbladm` and the `-c` option, see Chapter 13.

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 127.
- ... 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 210.

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

```
client% nistbladm -u `[name=g+rm,addr=g+rm],hosts.org_dir..doc.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” on page 138 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.doc.com.` table.

```
client% nistbladm -u 'name=g-rm,hosts.org_dir.doc.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 `doc.com.` domain to the user named `lincoln` whose home domain is `doc.com.`:

```
client% nischown lincoln.doc.com. hosts.org_dir.doc.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):

```
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 doc.com. domain to takeda whose home domain is doc.com. The entry is the one whose value in the name column is virginia.

```
client% nischown takeda.doc.com. '[name=virginia],hosts.org_dir.doc.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 doc.com. domain to admins.doc.com.:

```
client% nischgrp admins.doc.com. hosts.org_dir.doc.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 130):

```
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 doc.com. domain to sales.doc.com. The entry is the one whose value in the host name column is `virginia`.

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


Administering Passwords

This chapter describes how to use the `passwd` command from the point of view of an ordinary user (NIS+ principal) and how an NIS+ administrator manages the password system.

- “Logging In” on page 144
- “The Login incorrect Message” on page 144
- “The password expired Message” on page 145
- “The will expire Message” on page 145
- “The Permission denied Message ” on page 146
- “Changing Your Password” on page 146
- “Choosing a Password” on page 147
- “`nsswitch.conf` File Requirements ” on page 149
- “The `nispaswd` Command” on page 149
- “The `yppaswd` Command” on page 150
- “The `passwd` Command ” on page 150
- “The `nistbladm` Command ” on page 153
- “Related Commands” on page 157
- “Displaying Password Information ” on page 157
- “Changing Passwords ” on page 159
- “Locking a Password ” on page 160
- “Managing Password Aging ” on page 161
- “Forcing Users to Change Passwords ” on page 162
- “Setting Minimum Password Life ” on page 163
- “Setting a Password Age Limit ” on page 162

- “Establishing a Warning Period ” on page 164
- “Turning Off Password Aging ” on page 165
- “Password Privilege Expiration” on page 165
- “Specifying Maximum Number of Inactive Days” on page 167
- “Specifying Password Criteria and Defaults” on page 168
- “The `/etc/defaults/passwd` File” on page 169
- “Password Failure Limits” on page 171

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

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 147, 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 146.)

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

To maintain security, you should change your password regularly. (See “Choosing a Password” on page 147 for password requirements and criteria.)

Note - The `passwd` command now performs all functions previously performed by `nispaswd`. For operations specific to an 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 147 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 108 and "Changing Root Keys From Another Machine" on page 110 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#fOOD you can change it to dog#Meat but you cannot change it to daT#FOod.

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

This section describes how to administer passwords in an NIS+ namespace. 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 6, for this information).

Note - The `passwd` command now performs all functions previously performed by `nispasswd`. For operations specific to an 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`.

(The `nispasswd` command 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`.

(The `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 an 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 10-1 Access Rights for `passwd` Command

This Operation	Requires These Rights	To This Object
Displaying information	read	<code>passwd</code> table entry
Changing Information	modify	<code>passwd</code> table entry
Adding New Information	modify	<code>passwd</code> table

The `passwd` Command and Keys

If you use `passwd` in an 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.



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

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 165 and "Specifying Maximum Number of Inactive Days" on page 167.)

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.org_dir
```

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 156 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 163 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 162 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 longing time. See “Establishing a Warning Period ” on page 164 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 167 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 165 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:

`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` and `expire` fields as a number of days since January 1, 1970. For example:

TABLE 10-2 Number of Days Since 1/1/70

Date	Day Number
January 1, 1970	0
January 2, 1970	1
January 2, 1971	365
January 1, 1997	9863

Related Commands

The `passwd` and `nistbladm` commands provide capabilities that are similar to those offered by other commands. Table 10-3 summarizes their differences.

TABLE 10-3 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>nispaswd</code>	Is now linked to the <code>passwd</code> command. Using <code>nispaswd</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 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 expire inactive* where

TABLE 10-4 NIS+ Password Display Format

Field	Description	For Further Information
<i>username</i>	The user's login name.	
<i>status</i>	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 160.
<i>mm/dd/yy</i>	The date, based on Greenwich mean time, that the user's password was last changed.	
<i>min</i>	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 163.
<i>max</i>	The maximum number of days the password can be used without having to change it.	See "Setting a Password Age Limit " on page 162.
<i>warn</i>	The number of days' notice that users are given before their passwords have to be changed.	See "Establishing a Warning Period " on page 164.
<i>expire</i>	A date on which users loose the ability to log in to their accounts.	See "Password Privilege Expiration" on page 165.
<i>inactive</i>	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 167.

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

New passwords must meet the criteria described in “Password Requirements” on page 147.

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’ 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 an NIS+ environment (see “The `passwd` Command and “NIS+ Environment” ” on page 151) 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

When operating in an NIS+ environment (see “The `passwd` Command and “NIS+ Environment” ” on page 151), 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.

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 dismissed 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 enter:

```
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 162 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 162 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 163 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 164 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 167 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 165 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 169.)

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

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. Setting the *min* value greater than the *max* value has two effects:
 - The user is unable to change password. In this case, only someone with administer privileges could change the password. For example, in situations where multiple users share a common group password, setting the *min* value for that password greater than the *max* value would prevent any individual user from changing the group password.
 - The password is only valid for the length of time set by the *max* value, but the user cannot change it because the *min* value is greater than the *max* value. Thus, there is no way for the user to prevent the password from becoming invalid at the expiration of the *max* time period. In effect, this prevents the user from logging in after the *max* time period unless an administrator intervenes.

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 "Setting a Password Age Limit" on page 162.
- *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.

Note - You can also use Solstice AdminSuite to set a warn value for a user's password.

Turning Off Password Aging

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 162 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
```

Note - You can also use Solstice AdminSuite to set this parameter for a user's password.

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 153.

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, 1997, for a user named pete, on January 1, 1998 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 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 provide 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 10-2). *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 `pete` of December 31, 1995 you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:9493:n' [name=pete],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:-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 `sam`. On January 1, `sam` logs in to both machine-A and machine-B, and then logs off both machines. Four days later on January 4, `sam` logs in on machine-B and then logs out. Nine days after that on January 13, `sam` can still log in 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 `sam` must log in at least once every seven days, you would type:

```
station1% nistbladm -m 'shadow=n:n:n:n:7:n:n' [name=sam],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.

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 effect 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

Note - This is no point in setting a WARNWEEKS default unless you also set a MAXWEEKS default.

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.

Note - A `WARNWEEKS` default will not work unless there is also 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
```

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 of 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

This chapter describes NIS+ groups and how to administer them.

- “NIS+ Group Member Types” on page 175
- “Listing the Object Properties of a Group” on page 177
- “Creating an NIS+ Group ” on page 179
- “Deleting an NIS+ Group ” on page 180
- “Adding Members to an NIS+ Group ” on page 180
- “Listing the Members of an NIS+ Group ” on page 181
- “Removing Members From an NIS+ Group ” on page 182
- “Testing for Membership in an NIS+ Group ” on page 182

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

Solaris Groups

In a Solaris-NIS+ environment, there are three kinds of groups: UNIX groups, net groups, and NIS+ groups.

- *UNIX groups.* A UNIX group is simply a collection of users who are given additional UNIX access permissions. In an NIS+ namespace, UNIX group information is stored in the group table located in the `org_dir` directory object (`group.org_dir`). See Chapter 13, for information on how to add, modify, or delete members of a UNIX group and “group Table ” on page 610, for a description of the group table.

- *Net groups.* A net group is a group of workstations and users that have permission to perform remote operations on other workstations. In an NIS+ namespace, net groups information is stored in the netgroup table located in the `org_dir` directory object (`netgroup.org_dir`). See Chapter 13, for information on how to add, modify, or delete members of a net groups and “netgroup Table ” on page 612, for a description of the net group table.
- *NIS+ groups.* An NIS+ group is a set of NIS+ users that are assigned specific access rights to NIS+ objects, usually for the purpose of administering the namespace. NIS+ group information is stored in tables located in the `groups_dir` directory object.

NIS+ Groups

NIS+ groups are used to assign access rights to NIS+ objects to one or more NIS+ principles. These access rights are described in Chapter 6. Information about NIS+ groups is stored in tables located in the NIS+ `groups_dir` directory object. Information about each group is stored in a table of the same name. For example, information about the `admin` group is stored in `admin.groups_dir`.

It is recommended practice to create at least one NIS+ group called `admin`. The `admin` NIS+ group is normally used to designate those users who are to have NIS+ access rights. You can name this group anything you want, but the NIS+ manual set assumes that the group with NIS+ administrator privileges is named `admin`. You can also create multiple NIS+ groups with different sets of users and different sets of rights.

Note - Always use the `nisgrpadm` command to work with NIS+ group membership. You can also use the `nislsl` and `nischgrp` commands on the group table. Do not use the `nistbladm` command on the group table.

For a complete description of NIS+ group-related commands and their syntax and options, see the NIS+ man pages.

Related Commands

The `nisgrpadm` command performs most group administration tasks but several other commands affect groups as well:

TABLE 11-1 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> .	
<code>nisls</code>	Lists the contents of the <code>groups_dir</code> directory; in other words, all the groups in a domain. For each named groups there will be a table of that name in <code>groups_dir</code> .	"The <code>nisls</code> Command <code>niscat</code> Command With Directories" on page 187
<code>nischgrp</code>	Changes or assigns a group to any NIS+ object.	"Changing an Object or Entry's Group" on page 140
<code>niscat</code>	Lists the object properties and membership of an NIS+ group.	"Using <code>niscat</code> With NIS+ Groups" on page 177
<code>nisdefaults</code>	Lists, among other things, the group that will be assigned to any new NIS+ object.	"Displaying NIS+ Defaults—The <code>nisdefaults</code> Command" on page 131

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

Note - Do not use the `nistbladm` command to work with the NIS+ groups table.

NIS+ Group Member Types

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 178.

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.

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.

Nonmember Types

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.

Group Syntax

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 11-2:

TABLE 11-2 Specifying Group Members and Nonmembers

Type of member	Syntax
Explicit member	<i>username.domain</i>
Implicit member	<i>*.domain</i>
Recursive member	<i>@groupname.domain</i>
Explicit nonmember	<i>-username.domain</i>

TABLE 11-2 Specifying Group Members and Nonmembers (continued)

Type of member	Syntax
Implicit nonmember	- * . domain
Recursive nonmember	@groupname.domain

Using niscat With NIS+ 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.doc.com.  
Object Name : sales  
Owner : rootmaster.doc.com.  
Group : sales.doc.com.  
Domain : groups_dir.doc.com.  
Access Rights : ---rmcdr---r---  
Time to Live : 1:0:0  
Object Type : GROUP  
Group Flags :  
Group Members : rootmaster.doc.com.  
                 topadmin.doc.com.  
                 @.admin.doc.com.  
                 *.sales.doc.com.
```

Note - A better list of members is provided by the `nisgrpadm -l` command.

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 11-3 Rights Required for `nisgrpadm` Command

This Operation	Requires This Access Right	To This Object
Create a group	Create	<code>groups_dir</code> directory
Destroy a group	Destroy	<code>groups_dir</code> 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 these forms:

```
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 11-2):

```
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-name*. 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.domainname
```

When you create a group, an NIS+ groups table with the name you have given is created in groups_dir. You can use nislsl to confirm that the new group table now exists in groups_dir, and niscat to list the groups members listed in the table.

A newly created group contains no members. See “Adding Members to an NIS+ Group” on page 180 for information on how to specify who belongs to a group.

The example below creates three groups named admin. The first is in the doc.com. domain, the second in sales.doc.com., and the third in manf.doc.com. All three are created on the master server of their respective domains.

```
rootmaster# nisgrpadm -c admin.doc.com.
Group admin.doc.com. created.
salesmaster# nisgrpadm -c admin.sales.doc.com.
Group admin.sales.doc.com. created.
manfmaster# nisgrpadm -c admin.manf.doc.com.
Group admin.manf.doc.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 9). Used without options, it provides this output:

```
rootmaster# nisdefaults
Principal Name : rootmaster.doc.com.
Domain Name : doc.com.
Host Name : rootmaster.doc.com.
Group Name :
Access Rights : ----rmcdr---r---
Time to live : 12:0:0
```

```
Search Path : doc.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. For example, assuming a C-shell, to set NIS_GROUP to fns_admins.doc.com:

```
rootmaster# setenv NIS_GROUP fns_admins.doc.com
```

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.doc.com.-
c admin.sales.doc.com.
Group admin.sales.doc.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.doc.com. group.

```
salesmaster% nisgrpadm -d test.sales.doc.com.
Group 'test.sales.doc.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 "NIS+ Group Member Types" on page 175, 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.doc.com., and the principal makeba, from the manf.doc.com. domain, to the group top-team.sales.doc.com.

```
client% nisgrpadm -a Ateam panza valjean makeba.manf.doc.com.  
Added panza.sales.doc.com to group Ateam.sales.doc.com  
Added valjean.sales.doc.com to group Ateam.sales.doc.com  
Added makeba.manf.doc.com to group Ateam.sales.doc.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 doc.com. domain to the staff.doc.com. group. It is entered from a client in the doc.com. domain. Note the * symbol and the dot in front of the domain name.

```
client% nisgrpadm -a Staff *.doc.com.  
Added *.doc.com. to group Staff.manf.doc.com.
```

This example adds the NIS+ group admin.doc.com. to the admin.manf.doc.com. group. It is entered from a client of the manf.doc.com. domain. Note the @ symbol in front of the group name.

```
client% nisgrpadm -a admin @admin.doc.com.  
Added @admin.doc.com. to group admin.manf.doc.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.doc.com. group. It is entered from a client in the manf.doc.com. group:

```
client% nisgrpadm -l admin  
Group entry for admin.manf.doc.com. group:  
No explicit members  
No implicit members:  
Recursive members:  
@admin.doc.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.doc.com` from the `Ateam.sales.doc.com` group. It is entered from a client in the `sales.doc.com` domain:

```
client% nisgrpadm -r Ateam allende hugo.manf.doc.com.  
Removed allende.sales.doc.com. from group Ateam.sales.doc.com.  
Removed hugo.manf.doc.com. from group Ateam.sales.doc.com.
```

This example removes the `admin.doc.com` group from the `admin.manf.doc.com` group. It is entered from a client in the `manf.doc.com` domain:

```
client% nisgrpadm -r admin @admin.doc.com.  
Removed @admin.doc.com. from group admin.manf.doc.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.doc.com` group. It is entered from a client in the `doc.com` domain.

```
client% nisgrpadm -t admin topadmin  
topadmin.doc.com. is a member of group admin.doc.com.
```

This example tests whether the NIS+ principal `jo`, from the `sales.doc.com` domain, belongs to the `admin.sales.doc.com` group. It is entered from a client in the `doc.com` domain.

```
client% nisgrpadm -t admin.sales.doc.com. jo.sales.doc.com.  
jo.sales.doc.com. is a member of group admin.sales.doc.com.
```


Administering NIS+ Directories

This chapter describes NIS+ directory objects and how to administer them.

- “Listing the Object Properties of a Directory” on page 186
- “Listing the Contents of a Directory—Terse” on page 188
- “Listing the Contents of a Directory—Verbose” on page 188
- “Creating a Directory ” on page 189
- “Adding a Replica to an Existing Directory ” on page 191
- “Removing a Directory ” on page 193
- “Disassociating a Replica From a Directory ” on page 193
- “Removing Nondirectory Objects ” on page 195
- “Starting a NIS-Compatible Daemon” on page 196
- “Starting a DNS-Forwarding NIS-Compatible Daemon” on page 196
- “Stopping the NIS+ Daemon” on page 197
- “Initializing a Client” on page 197
- “Initializing the Root Master Server ” on page 198
- “Starting and Stopping the Cache Manager” on page 199
- “Displaying the Contents of the NIS+ Cache” on page 200
- “Pinging and Checkpointing” on page 200
- “Displaying When Replicas Were Last Updated ” on page 201
- “Checkpointing a Directory” on page 202
- “Changing the Time-to-Live of an Object” on page 207
- “Changing the Time-to-Live of a Table Entry” on page 207

NIS+ Directories

NIS+ directory objects are used to store information related to an NIS+ domain. For each NIS+ domain, there is a corresponding NIS+ directory structure. See Chapter 4, for more information about NIS+ directories.

For a complete description of NIS+ directory-related 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 doc.com.  
Object Name : doc  
Owner : rootmaster.doc.com.  
Group :  
Domain : Com.  
Access Rights : r---rmdir---r---  
Time to Live : 24:0:0  
Object Type : DIRECTORY  
.  
.
```

The `nisl`s Command

Command With Directories

The `nisl`s 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:

```
nisl [-dgLmMR] directory-name
```

To display in verbose format, use:

```
nisl -l [-gm] [-dLMR] directory-name
```

TABLE 12-1 Options for the `nisl`s 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] directory-name
```

or

```
nisls [-dLMR]
```

For example, this instance of `nisls` is entered from the root master server of the root domain `doc.com.`:

```
rootmaster% nisls doc.com.:  
org_dir  
groups_dir
```

Here is another example entered from the root master server:

```
rootmaster% nisls -R sales.doc.com.  
sales.doc.com.:  
org_dir  
groups_dir  
groups_dir.sales.doc.com.:  
admin  
org_dir.sales.doc.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] directory-name
```

or

```
nisls -l [-gm] [-dLMR]
```

Here is an example, entered from the master server of the root domain `doc.com.`:

```
rootmaster% nisls -l
doc.com.
D r---rmcdr---r--- rootmaster.doc.com. date org_dir
D r---rmcdr---r--- rootmaster.doc.com. date groups_dir
```

The nismkdir Command

Note - This section describes how to add a nonroot server to an existing domain using the `nismkdir` command. An easier way to do this is with the `nisserver` script as described in *Solaris Naming 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 197.) 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 *Solaris Naming 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:

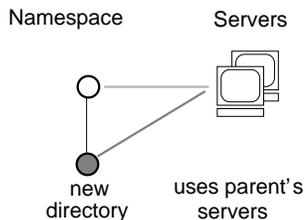
```
nismkdir -m master directory
nismkdir -s replica directory
```



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.doc.com.` directory and specifies its master server, `smaster.doc.com.` and its replica, `repl.doc.com.` It is entered from the root master server.

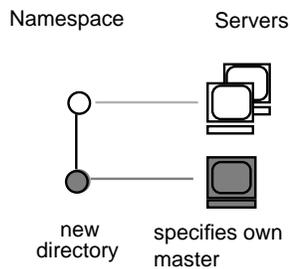
```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -m smaster.doc.com. groups_dir.sales.doc.com.
rootmaster% nismkdir -s repl.doc.com. sales.doc.com.
rootmaster% nismkdir -s repl.doc.com. org_dir.sales.doc.com.
rootmaster% nismkdir -s repl.doc.com. groups_dir.sales.doc.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.doc.com.` directory and associates it with its parent directory's master and replica servers.

```
rootmaster% nismkdir sales.doc.com
```



The second example creates the `sales.doc.com.` directory and specifies its own master server, `smaster.doc.com.`

```
rootmaster% nismkdir -m smaster.doc.com. sales.doc.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.doc.com.` domain already existed, the `nismkdir` command as shown above would have made `salesmaster.doc.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 *Solaris Naming Setup and Configuration Guide*.

Keep in mind the following principles:

- Root domain servers reside in (are part of) the root domain.
- Subdomain servers reside in (are part of) the parent domain immediately above the subdomain in the hierarchy. For example, if a namespace has one root domain named `prime` and a subdomain named `sub1`:
 - The master and replica servers that serve the `prime` domain are themselves part of the `prime` domain because `prime` is the root domain.
 - The master and replica servers that serve the `sub1` subdomain are also part of the `prime` domain because `prime` is the parent of `sub1`.
- While it is possible for a master or replica server to serve more than one domain, doing so is not recommended.

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 `rep1` being the name of the new replica machine:

```
rootmaster% nismkdir -s repl.doc.com. doc.com.
rootmaster% nismkdir -s repl.doc.com. org_dir.doc.com.
rootmaster% nismkdir -s repl.doc.com. groups_dir.doc.com.
```



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 doc.com.
rootmaster# nisping org_dir.doc.com.
rootmaster# nisping group_dir.doc.com.
```

You should see results similar to these:

```
rootmaster# nisping doc.com.
Pinging replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
Last update occurred at Wed Nov 18 19:54:38 1995
Replica server is repl.doc.com.
Last update seen was Wed Nov 18 11:24:32 1995
Pinging ... repl.doc.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 a directory is removed or disassociated from a replica server, that machine no longer functions as an NIS+ replica server for that NIS+ domain.)

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.

If problems occur, see “Removal or Disassociation of NIS+ Directory from Replica Fails” on page 510.

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
nisping domain
```

This example removes the `manf.doc.com.` directory from beneath the `doc.com.` directory:

```
rootmaster% nisrmdir manf.doc.com.
rootmaster% nisping doc.com.
```

Disassociating a Replica From a Directory

To disassociate a replica server from a directory, you must first remove the directory's `org_dir` and `groups_dir` subdirectories. To do this, use the `nisrmdir` command with the `-s` option. After each of the subdirectories are removed, you must run `nisping` on parent domain.

```
nisrmdir -s replicanameorg_dir.domain
nisrmdir -s replicanamegroups_dir.domain
nisrmdir -s replicaname domain
nisping domain
```

This example disassociates the `manfreplical` server from the `manf.doc.com.` directory:

```
rootmaster% nisrmdir -s manfreplica1 org_dir.manf.doc.com.
rootmaster% nisrmdir -s manfreplica1 groups_dir.manf.doc.com.
rootmaster% nisrmdir -s manfreplica1 manf.doc.com.
rootmaster% nisping manf.doc.com.
```

If the replica server you are trying to dissociate is down or out of communication, the `nisrmdir -s` command returns a `Cannot remove replicaname: attempt to remove a non-empty table` error message. In such cases, you can run `nisrmdir -f -s replicaname` on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -s` to dissociate an out-of-communication replica, you *must* run `nisrmdir -f -s` *again* as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s replicaname` when the replica is back in service, the old out-of-date information left on the replica could cause problems.

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 180), and you can empty tables with `nistbladm -r` or `nistbladm -R` (see "Deleting a Table" on page 218).

To remove a nondirectory object, use:

```
nisrm [-if] object-name
```

TABLE 12-2 `nisrm` Syntax Options

Option	Purpose
<code>-i</code>	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.
<code>-f</code>	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.doc.com. groups.org_dir.doc.com.  
Remove admins.doc.com.? y  
Remove groups.org_dir.doc.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 *Solaris Naming 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 12-3 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.

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` 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 *Solaris Naming Setup and Configuration Guide*.

The `nisinit` command initializes a workstation to be an NIS+ client or server. 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 *Solaris Naming Setup and Configuration Guide*.

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 *Solaris Naming 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
```

You will need the following information

- The superuser password of the workstation that will become the root master server.
- The name of the new root domain. The root domain name must have at least two elements (labels) and end in a dot (for example, *something.com.*). The last element must be either an Internet organizational name (as shown in Table 12-4), or a two or three character geographic identifier such as *.jp.* for Japan.

TABLE 12-4 Internet Organizational Domains

Domain	Purpose
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
mil	Military groups
net	Major network support centers
org	Nonprofit organizations and others
int	International organizations

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 and Stopping 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 : doc.com.
Type : NIS
Master Server :
  Name : rootmaster.doc.com.
  Public Key : Diffie-Hellman (192 bits)
  Universal addresses (3)
  .
  .
Replicate:
  Name : rootrepical.doc.com.
  Public Key : Diffie-Hellman (192 bits)
  Universal addresses (3)
  .
  .
Time to live : 12:0:0
Default Access Rights :
```

Pinging and Checkpointing

When a change is made to the NIS+ data set, that change is made in the memory of the master server for the NIS+ domain (or subdomain). A record of the change is also logged in the master server's transaction log (`/var/nis/data/trans.log`).

Normally, the master server transfers a change in the NIS+ data set to the domain's replica servers 120 seconds (2 minutes) after the change was made. This transfer process is called *pinging*. When the master server pings a replica, it updates the replica's data set with the change. The changed NIS+ data now resides in memory of the master and replica servers.

If the automatic ping process fails to update one or more replica servers, you need to manually force a ping as described in "Forcing a Ping" on page 201. If you suspect that a replica has not been correctly updated with the most current NIS+ data, you can check when the replica was last updated as described in "Displaying When Replicas Were Last Updated" on page 201.

Changes to the NIS+ data set stored in server memory and recorded in the transaction log need to be written into the NIS+ tables stored on disk. The process of updating the NIS+ tables is called *checkpointing*.

Checkpointing is not an automatic process. You must issue the checkpoint command as described in “Checkpointing a Directory” on page 202.

The `nisping` Command

The `nisping` command is used to:

- Display when a replica was last pinged as described in
 - Force the master server to ping a replica if the automatic ping cycle has not been successful as described in “Forcing a Ping” on page 201
- Checkpoint servers as described in

Displaying When Replicas Were Last Updated

When used with the `-u` option, the `nisping` command displays the update times for the master and replicas of the local domain.

```
/usr/lib/nis/nisping -u [domain]
```

To display the last updates in some other domain, specify the domain name in the command line. Note that when used with the `-u` option, the `nisping` command does not actually ping any replicas.

For example, to display the most recent replica update times for the local `doc.com.` domain, you would enter:

```
rootmaster# /usr/lib/nisping -u
Last updates for directory doc.com.:
Master server is rootmaster.doc.com.
  Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplica1.doc.com.
  Last update seen was Wed Nov 25 10:53:37 1992
```

Forcing a Ping

If the `nisping -u` command reveals that a replica has not been properly updated, you can use the `nisping` command to force the master server to ping all the replicas in a domain, or one replica in particular.

To ping all the replicas, use the `nisping` command without options:

```
/usr/lib/nis/nisping
```

This forces the master server to ping all the replicas in the domain. Here is an example that pings all the replicas of the local `doc.com.` domain:

```
rootmaster# /usr/lib/nis/nisping
Pinging replicas serving directory doc.com.:
Master server is rootmaster.doc.com.
  Last update occurred at Wed Nov 25 10:53:37 1992
Replica server is rootreplical.doc.com.
  Last update seen was Wed Nov 18 11:24:32 1992
Pinging ... rootreplical.doc.com.
```

To ping all the replicas in a domain other than the local domain, append a domain name:

```
/usr/lib/nis/nisping domainname
```

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

Each domain and subdomain should be checkpointed at least once every 24 hour, or more often if the transaction log grows too large in relationship to swap space or total disk space.

Note - Checkpointing large domains, or any domain with a large transaction log, is a time-consuming process which ties up NIS+ servers and slows NIS+ service. While a server is checkpointing, it will still answer requests for service, but it will be unavailable for updates. If possible, checkpoint operations should be scheduled for times when system use is low. You can use the `cron` file to schedule checkpoint operations.

To perform a checkpoint operation, run `nisping -C` on the domain's master server. It is good practice to first ping all replicas before checkpointing. This ensures that the replicas are checkpointing data that is current and up to date.

- To checkpoint a particular directory, run the `nisping` command with the `-C directoryname` option. For example,

```
rootmaster# /usr/lib/nis/nisping
rootmaster# /usr/lib/nis/nisping -C org_dir
```

- To checkpoint *all* the directories in the local domain, run the `nisping` command with the `-C -a` options. For example,

```
rootmaster# /usr/lib/nis/nisping
rootmaster# /usr/lib/nis/nisping -C -a
```

Once a server has transferred information from the server's transaction log to the appropriate NIS+ tables, the transactions in the log file are erased to conserve disk space.

For example, to checkpoint all of the directories in the `doc.com.` domain, you would enter:

```
rootmaster# /usr/lib/nis/nisping -C -a
Checkpointing replicas serving directory doc.com. :
Master server is rootmaster.doc.com.
  Last update occurred at Wed May 25 10:53:37 1995
Master server is rootmaster.doc.com.
checkpoint has been scheduled with rootmaster.doc.com.
Replica server is rootreplica1.doc.com.
  Last update seen was Wed May 25 10:53:37 1995
Replica server is rootreplica1.doc.com.
checkpoint has been scheduled with rootmaster.doc.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 12-5 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 doc.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 : doc.com.
Entry type : ADD Name
Entry timestamp : Wed Nov 25 10:50:59 1992
Principal : rootmaster.doc.com.
Object name : org_dir.doc.com.
.....Object.....
Object Name : org_dir
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : doc.com.
Access Rights : r---rmcdr---r---
Time to Live : 24:0:0
Object Type : DIRECTORY
Name : 'org_dir.doc.com.'
```

(continued)

```

Type: NIS
Master Server : rootmaster.doc.com.
.
.
.....
@@@@@@@@@@@@@@@@TRANSACTION@@@@@@@@@@@@
#00000, XID : 2

```

The nischt1 Command

The `nischt1` 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 9.

To change the time-to-live value of objects, use:

```
nischt1 time-to-live object-name
```

or

```
nischt1 [-L] time-to-live object-name
```

To change the time-to-live value of entries, use:

```
nischttl time-to-live \  
[column=value,...], \  
table-name
```

OR

```
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 1,234 seconds. A number followed by the letter *s* is also interpreted as a number of seconds. Thus, 987*s* 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, 90*m* 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, 9*h* 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, 7*d* for TTL would be interpreted as 7 days.

These values may be used in combination. For example, a TTL value of 4*d*3*h*2*m*1*s* 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 12-6 `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 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:

```
client% nischttl 86400 sales.doc.com.  
client% nischttl 24h sales.doc.com.  
client% nischttl 2dlh1mls sales.doc.com.
```

The first two commands change the time-to-live of the `sales.doc.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`.

```
nischttl [-ALP] time-to-live \  
[column=value, ...], \  
table-name
```

Note - C-shell users should use quotes to prevent the shell from interpreting the square brackets (`[]`) around the column value as a meta character.

These examples are similar to those above, but they change the value of table entries instead of objects:

```
client% nischttl 86400 '[uid=99],passwd.org_dir.doc.com.'  
client% nischttl 24h '[uid=99],passwd.org_dir.doc.com.'  
client% nischttl 2dlh1m1s '[name=fred],hosts.org_dir.doc.com'
```

Administering NIS+ Tables

This chapter describes NIS+ tables and how to administer them. (See Appendix C, for detailed descriptions of the default NIS+ tables.)

- “NIS+ Tables” on page 210
- “The `nistbladm` Command ” on page 210
- “`nistbladm` and Column Values ” on page 212
- “`nistbladm` and Indexed Names” on page 215
- “`nistbladm` and Groups” on page 215
- “Creating a New Table ” on page 216
- “Deleting a Table ” on page 218
- “Adding Entries to a Table ” on page 218
- “Modifying Table Entries ” on page 222
- “Removing Table Entries ” on page 224
- “The `niscat` Command ” on page 226
- “Displaying the Contents of a Table ” on page 227
- “Displaying the Object Properties of a Table or Entry ” on page 228
- “The `nismatch` and `nisgrep` Commands ” on page 229
- “Searching the First Column ” on page 232
- “Searching a Particular Column ” on page 232
- “Searching Multiple Columns ” on page 233
- “The `nisln` Command” on page 233
- “Creating a Link ” on page 234
- “The `nissetup` Command” on page 234


```

column="value" \
... tablename
nistbladm -a indexedname

```

```

nistbladm options \
 [ columnspec | columnvalue ] \
 [ tablename | indexedname ]

```

Where:

- *Columnspec* is a specification defining a column to be created in a table as described in “Specifying Table Columns ” on page 216.
- *columnvalue* identifies a particular cell value in the table identified by *tablename* as described in “nistbladm and Column Values ” on page 212.
- *Tablename* is the name of the table. For example, `hosts.org_dir.doc.com`.
- *Indexedname* identifies a particular cell value in a certain table as described in “nistbladm and Column Values ” on page 212. In essence *indexedname* is the equivalent of *columnvalue* plus *tablename*.

TABLE 13-1 nistbladm Options

Option	Description
-a -A	Add an entry to an existing NIS+ table. The -a option returns an error if execution of the command would result in overwriting any existing entry. The -A option forces execution of the command even if it results in overwriting an existing entry. (See “Adding Entries to a Table ” on page 218.)
-D <i>defaults</i>	Specify a different set of default properties when creating an object. (See the nistbladm man page for details.)
-d	Destroy a table. (See “Deleting a Table ” on page 218.)
-c	Create a table. (See “Creating a New Table ” on page 216.)

TABLE 13-1 nistbladm Options (continued)

Option	Description
-r -R	Remove one or more entries from an existing NIS+ table. The -r option returns an error if execution of the command would result in removal of more than one entry. The -R option forces execution of the command even if it results in removing multiple entries. (See "Removing Table Entries " on page 224.)
-m	An obsoleted option for modifying table entries that is still supported for backwards compatibility. The -e and -E options are the preferred method for editing entries.
-e -E	Edit an entry in an existing NIS+ table. The -e option returns an error if execution of the command would affect more than one entry. The -E option forces execution of the command even if it results in changing an existing entry in such a way as to overwrite a different entry. (See "Modifying Table Entries " on page 222.)

nistbladm and Column Values

Column values are used to identify individual entries in tables using the format:

```
columnname="value" , \
columnname="value" , ...
```

Where:

- *columnname* is the name of a table column
- *value* is the contents of a particular cell within a column. That value is what identifies a table row. (When using *column=value* to create or modify table data, always enclose the *value* element in quotes.)

For example, suppose you had a `hosts` table that listed machine names and IP addresses:

TABLE 13-2 Example Hosts Table

IP address	name	aliases
129.146.168.4	altair	
129.146.168.119	deneb	mail
129.146.168.120	regulus	dnsmaster
129.146.168.121	regulus	dnsmaster
129.146.168.11	sirius	

In this example, you could identify the `altair` entry (row) in three different ways using the *column=value* of:

- `name=altair`
- `address=129.146.168.4`
- `name=altair,address=129.146.168.4`.

But notice in the table above that the machine `regulus` is multi-homed and has *two* IP addresses. In that case, the *column=value* of `host=regulus` identifies two rows. To identify just the first `regulus` row, you would enter either:

- `address=129.146.168.120` or
- `address=129.146.168.120,name=regulus,dnsmaster`

Note - Some `nistbladm` operations require that you enter a *column=value* pair for every column in the table.

`nistbladm`, Searchable Columns, and Keys `nistbladm` and Column Values

When an NIS+ table is created, one or more columns are designated *searchable* with either the `S` or the `I` flags as described in “Specifying Table Columns ” on page 216. You can use the `niscat -o tablename` command to display a list of a table’s columns and their characteristics.

A table is *keyed* on its searchable columns. This means that each row in the table must have a unique combination of values in the searchable columns. For example, if

a table has one searchable column, each table row must have a unique value in that column, no two rows may contain the same value.

For example, suppose you had a table containing one searchable column named `city` and a non-searchable column named `country`. The following rows would all be permitted:

City	Country
San Francisco	United States
Santa Fe	United States
Santiago	Chile

But you could not have two rows like:

City	Country
London	Canada
London	England

If a table has multiple searchable columns, it is the *combination* of values that must be unique. For example, suppose you had a table containing two searchable columns, `Lastname`, `Firstname` and a non-searchable column named `city`. The following rows would all be permitted:

Lastname	Firstname	City
Kuznetsov	Sergei	Odessa
Kuznetsov	Rima	Odessa
Sergei	Alex	Odessa

But you could not have two rows like this:

Lastname	Firstname	City
Kuznetsov	Rima	Odessa
Kuznetsov	Rima	Chelm

NIS+ commands use the values in the searchable columns to identify specific table rows.

nistbladm and Indexed Names

In the context of table administration, an NIS+ *indexed name* is a name that combines a table name with column value search criteria to identify and select particular entries in a table. Indexed names use the format:

```
[ search_criteria ], tablename . directory
```

Note that `search_criteria` must be enclosed in square brackets []. The `search_criteria` use the format:

```
columnname=value, \  
columnname=value, . . .
```

Where `columnname=value` pairs are column values from the table's searchable columns as described in "nistbladm and Column Values" on page 212.

For example, to identify the `altair` entry in Table 13-2 you could use the indexed name:

```
[addr=129.146.168.4,cname=altair],hosts.org_dir.doc.com.
```

The `nistbladm -R` command allows you to remove all the entries in a table by using the two square brackets with nothing between them [] as a wildcard specifying all table rows.

nistbladm and Groups

In a Solaris-NIS+ environment, there are three types of groups:

- **UNIX groups.** Information about UNIX groups is stored in the `groups.org_dir` table. Use `nistbladm` to administer UNIX group information.
- **Netgroups.** Information about net groups is stored in the `netgroups.org_dir` table. Use `nistbladm` to administer net group information.
- **NIS+ groups.** Information about NIS+ groups is stored in one or more tables in the `groups_dir` directory object. Use `nisgrpadm` to administer NIS+ group information.

Note - Do not use `nistbladm` to administer NIS+ groups.

(See "Solaris Groups" on page 173 for more information on the different types of groups and how to work with them.)

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 tabletype columnspec \  
... tablename
```

Where:

- *Tabletype* is simply a name that identifies a class of tables to which this table belongs. You can use any name you choose.
- A *columnspec* specifies the name and characteristics of each column in a new table. Enter one *columnspec* for each column you want in your new table. Separate the *columnspecs* with spaces:

```
nistbladm -c tabletype columnspec columnspec \  
... columnspec tablename
```

Columnspec formats are described in “Specifying Table Columns ” on page 216, below.

Specifying Table Columns

Each *columnspec* entry has two to four components in the format:

```
name=type,rights:
```

TABLE 13-3 Table Column Components

Component	Description
<i>name</i>	Name of the column
=	An equal sign which is required.

TABLE 13-3 Table Column Components (continued)

Component	Description
<i>type</i>	[Optional] The type of column specified by the letters S, I or C (see Table 13-4). 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 127.

A column can be one of the following types:

TABLE 13-4 Table Column Types

Type	Description
	No column type specified after the = sign. The column is neither searchable nor encrypted.
S	Searchable.
I	Searchable, but case-insensitive. When NIS+ commands search through the column, they will ignore case.
C	Encrypted.

NIS+ commands search through the column and identify individual table rows based on the contents of the searchable columns. Searchable columns are designated with either the S or the I option. In database terminology, a searchable column is a key. The first column in each table must be searchable. The remaining columns do not have to be searchable. Because the table is keyed on the searchable columns, if you have more than one searchable column, they must be the first and subsequent columns and not skip any columns. For example, if only one column in a table is searchable, it has to be the first column. If two columns are searchable, they must be the first two columns. (see "nistbladm, Searchable Columns, and Keysnistbladm and Column Values " on page 213 for more information on searchable columns.)

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.

In the example below, a 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 Name=C \  
divs.mydir.doc.com.
```

For more information about specifying column access rights when creating a table, see “Setting Column Rights When Creating a Table” on page 137.

Note - NIS+ assumes that all column entries are null terminated. Applications and routines that write information to NIS+ tables must be configured to null terminate each column entry.

Creating Additional Automount Table

If you are creating an automount table, the table can have only two columns. The first column must be named `key` and the second column must be named `value`. For example, to create an automount table named `auto1`, you would enter:

```
master% nistbladm -c key-value key=S value= auto1.org_dir.doc.com.
```

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 Table Entries ” on page 224). This example deletes the `divs` table from the `doc.com.` directory:

```
rootmaster% nistbladm -d divs.doc.com.
```

Adding Entries to a Table

To add new entries (rows) to a table, use `nistbladm` with either the `-a` or `-A` options followed by either one or more `column=value` pairs and the table name or an indexed name as described in “`nistbladm` and Indexed Names” on page 215.

```
nistbladm [-a | -A] indexedname
nistbladm [-a | -A] column="value" \
column="value" \
... tablename
```

When adding new entry rows to a table with either `-a` or `-A`:

- Always enclose the *value* element in quotes. For example, to add an entry where the value of the `cname` column is `deneb`, the `column=value` pair would look like:
`cname="deneb"`
- You *must* specify a value for *every* column in the table.
- To specify that a column in the entry row you are adding is empty use `column=" "`. In other words, for the *value*, enclose a space between the quote marks.

Note - NIS+ is a naming service and its tables are designed to store references to objects, not the objects themselves. NIS+ is optimized to support 10,000 objects with a combined total size of all tables not more than 10M bytes. NIS+ does not support individual tables where the sum of field sizes in a single column are greater than approximately 7k. If a table is too large, `rpc.nisd` may fail.

Adding a Table Entry With the `-a` Option

The `-a` option adds an entry to a table unless the entry already exists, in which case it returns an error. An entry is defined as *existing* if its values in the searchable columns exactly match the values in the new entry's searchable columns. (The values in non-searchable columns are not taken into account.)

To use the `-a` option, you must specify a value for every column in the table:

```
nistbladm -a column="value" \
column="value" \
... tablename
nistbladm -a indexedname
```

(To list the names and characteristics of table columns, use the `niscat -o tablename` command.)

For example, to add a new row to a table named `depts` using `column=value` pairs, you would enter:

```
rootmaster% nistbladm -a Name='R&D' Site='SanFran' \  
Name='vattel' depts.doc.com.
```

To add the same entry using an indexed name, you would enter:

```
rootmaster% nistbladm -a [Name='R&D',Site='SanFran',\  
Name='vattel'],depts.doc.com.
```

Both examples would produce a table row that looked like this:

Dept	Site	Name
R&D	SanFran	vattel

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. You must run `nistbladm` once for each entry row you want to add.

If a table row already exists with values in each column that are identical to the entry you are trying to create, `nistbladm -a` will return an error. You cannot have two identical entry rows in a table. In this context, rows are considered *identical* if the values in the *searchable* columns are identical, the values in none search able columns are not considered.

For example, if the `Dept` and `Site` columns are searchable, and the `Name` column is not searchable, `nistbladm` considers the following two rows to be identical:

Dept (searchable)	Site (searchable)	Name (not searchable)
Sales	Vancouver	Hosteen
Sales	Vancouver	Lincoln

In this example, `nistbladm -a` would not allow you to create the `Sales Vancouver Lincoln` row.

However if just *some* of the searchable columns have values identical to the entry you are trying to create, `nistbladm -a` will create a new entry as specified. For example, you could run the following commands to create two similar, but not identical, rows in a `depts` table:

```

rootmaster% nistbladm -a Dept='Sales' \
    Site='Vancouver' Name='hosteen' staff.doc.com.
rootmaster% nistbladm -a Dept='Sales' \
    Site='SanFran' Name='lincoln' staff.doc.com.

```

Which would produce rows that had some, but not all identical values in the searchable columns:

Dept	Site	Name
Sales	Vancouver	hosteen
Sales	SanFran	lincoln

Adding a Table Entry With the `-A` Option

The `-A` option is designed for applications where you need to force `nistbladm` to overwrite an existing entry. Like the `-a` option, `-A` adds a new entry to a table. However, if the entry already exists, instead of exiting with an error, it overwrites the existing entry row.

When using the `-A` option, you must specify all columns in the entry.

For example, suppose the following table exists and the `Dept` and `Site` columns are searchable:

Dept (searchable)	Site (searchable)	Name
Sales	SanFran	Lincoln

Now you run the following command:

```

rootmaster% nistbladm -A Name=Sales Site=SanFran \
    Name=Tsosulu depts.doc.com.

```

The `-a` option would have returned an error, since the entry specified by `Name=Sales Site=SanFran` already exists. But the `-A` option allows you to overwrite the existing row.

Dept	Site	Name
Sales	SanFran	Tsosulu

Modifying Table Entries

Existing table entries are edited (modified) using either the `-e` or `-E` options. The Solaris 2.6 release also supports use of the `-m` option for backwards compatibility with earlier releases. (All new applications and command line operations should use either the `-e` or `-E` options.)

To edit an existing entry (row) in a table, use `nistbladm` with either the `-e` or `-E` options followed by one or more `column=value` pairs that specify the new values and ending with an indexed name that identifies a particular row in a table as described in “`nistbladm` and Indexed Names” on page 215.

```
nistbladm [-e | -E] column="value" \  
column="value" \  
... indexedname
```

When adding new entry rows to a table with either `-e` or `-E`:

- Always enclose the *value* element in quotes. For example, to change the value of the `cname` column to `deneb`, the `column=value` pair would look like:
`cname="deneb"`
- You can only edit values in searchable columns one entry (row) at a time.
- To specify that a column in the entry row that you are editing be empty, use `column=" "`. In other words, for the *value*, enclose a space between the quote marks.

Editing a Table Entry With the `-e` Option

The `-e` option edits an entry in a table unless doing so would result in changing values in searchable columns in more than one entry row, in which case it returns an error. (The values in non-searchable columns are not taken into account.)

```
nistbladm column="value" \  
column="value" \  
... indexedname
```

To use the `-e` option, you only need to specify the column values you are changing.

For example, suppose you had the table:

Dept	Site	Name
Sales	SanFran	Tsosulu

To change the value of the Name column to Chandar, you would enter:

```
master% nistbladm -e Name="Chandar" [Dept='Sales',Site='SanFran'],\
depts.doc.com.
```

Now the table looks like this:

Dept	Site	Name
Sales	SanFran	Chandar

(Note that in the example above, the indexed name did not need to include the Name column because in these examples that column is not searchable.)

C-shell users should also use quotes to set off expressions using square brackets.

You can use the `-e` option to edit the values in searchable columns so long as the new values you specify affect only the single row identified by the indexed name. For example, to change the department to `Manf`, you would enter:

```
master% nistbladm -e Dept="Manf" [Dept='Sales',Site='SanFran'],\
depts.doc.com.
```

Dept (searchable)	Site (searchable)	Name
Manf	SanFran	Chandar

However, if an entry row already existed with `Manf` and `SanFran` in the searchable columns, the `-e` option would return an error.

You can specify changes to multiple columns so long as they all apply to a single entry row. For example, to change both the Dept and Name values, you would enter:

```
master% nistbladm -e Dept="Manf" Name='Thi' \
[Dept='Sales',Site='SanFran'],depts.doc.com.
```

Dept (searchable)	Site (searchable)	Name
Manf	SanFran	Thi

Editing a Table Entry With the `-E` Option

The `-E` option is designed for applications where you need to force `nistbladm` to overwrite an existing entry even if doing so will affect more than one entry.

For example, suppose your table had the following rows:

Dept (searchable)	Site (searchable)	Name
Sales	SanFran	Chandar
Sales	Alameda	Achmed

Now you run the following command:

```
master% nistbladm -E Site="Alameda" Mgr="Chu" \
[Div='Sales',Site='SanFran'],depts.doc.com.
```

Which would change the Sales SanFran Chandar row to Sales Alameda Chu. But Sales Alameda are the key values identifying the Sales Alameda Achmed row, so that row would also be changed. The result would be a single row where once there had been two rows:

Dept (searchable)	Site (searchable)	Name
Sales	Alameda	Chu

The `-e` option would have returned an error, since the edit would affect more than one row. But the `-E` option allows you to affect more than one entry row.

Removing Table Entries

- To remove a single entry from a table, use the `-r` option as described in “Removing Single Table Entries” on page 225.

- To remove multiple entries from a table, use the `-R` option as described in “Removing Multiple Entries From a Table ” on page 225

Removing Single Table Entries

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 [Dept=Manf-1,Site=Emeryville,Name=hosteen],\
depts.doc.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.doc.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`):

Dept	Site	Name
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.doc.com.
```

Dept	Site	Name
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 between the square brackets, as in this example:

```
rootmaster% nistbladm -R [],depts.doc.com.
```

When used with the `nistbladm -R` command, an empty set of square brackets is interpreted as a wildcard specifying all table rows.

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 13-5 niscat 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.doc.com.  
#Name:Site:Name  
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 9.

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:

Table

```
rootmaster# niscat -o hosts.org_dir.doc.com.  
Object Name : hosts  
Owner : rootmaster.doc.com.  
Group : admin.doc.com.  
Domain : org_dir.doc.com.  
Access Rights : ----rmcdr----  
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: -----
```

Table entry

```

rootmaster# niscat -o [name=rootmaster],hosts.org_dir.doc.com.
Object Name : hosts
Owner : rootmaster.doc.com.
Group : admin.doc.com.
Domain : org_dir.doc.com.
Access Rights : ----rmdr----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 13-6 below.

TABLE 13-6 Characteristics of `nismatch` and `nisgrep`

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	<i>column=string ... tablename[column= string, ...], tablename</i>	<i>column=exp ... tablename</i>

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.doc.com`. Only the first two columns are searchable.

Name (S)	Site (S)	Name
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.doc.com.
```

The regular expression symbols are summarized in Table 13-7, below.

TABLE 13-7 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> .

TABLE 13-7 Regular Expression Symbols (continued)

Symbol	Description
+	Find something that appears one or more times.
?	Find any value.
\ 's-char'	Find a special character, such as ? or \$.
x y	Find a character that is either x or y.

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 13-8 nismatch and nisgrep 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.doc.com.
rootmaster% nisgrep -h 'R&D' depts.doc.com.
```

Note - Quotes are used in the 'R&D' expression above 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.doc.com
rootmaster% nisgrep -h Site=SanFran depts.doc.com
```

Searching Multiple Columns

To search for entries with matches in two or more columns, use the following syntax:

```
nismatch [-h] [column=string, ... \
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,Name=jhill], depts.doc.com.
rootmaster% nisgrep -h Site=SanFran Name=jhill depts.doc.com.
```

The nisln Command

The `nisln` command creates symbolic links between NIS+ objects such as tables and directories. All NIS+ administration commands accept the `-L` flag, which directs them to follow links between NIS+ objects.

Note - Do not link table entries. Tables may be linked to other tables, but do not link an entry in one table to an entry in another table.

To create a link to another object (table or directory), 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:

```
nisln source target
```

TABLE 13-9 `nisln` Options

Option	Description
<code>-L</code>	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.
<code>-D</code>	Defaults. Specify a different set of defaults for the linked object. Defaults are described in "Specifying Nondefault Security Values at Creation Time" on page 135.

Creating a Link

To create a link between objects such as tables and directories, specify both object names: first the *source*, and then the *target*. Do not link table entries.

```
nisln source-object target-object
```

The `nissetup` Command

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 236. Expanding a directory into a domain is part of the process of setting up a domain.

Note - When setting up a new NIS+ domain, the `nisserverscript` is easier to use than the `nissetup` command. See *Solaris Naming Setup and Configuration Guide* for a full description of using `nisserver`.

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.

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 doc.com.
org_dir.doc.com. created
groups_dir.doc.com. created
auto_master.org_dir.doc.com. created
auto_home.org_dir.doc.com. created
bootparams.org_dir.doc.com. created
cred.org_dir.doc.com. created
ethers.org_dir.doc.com. created
group.org_dir.doc.com. created
hosts.org_dir.doc.com. created
mail_aliases.org_dir.doc.com. created
sendmailvars.org_dir.doc.com. created
netmasks.org_dir.doc.com. created
netgroup.org_dir.doc.com. created
networks.org_dir.doc.com. created
passwd.org_dir.doc.com. created
protocols.org_dir.doc.com. created
rpc.org_dir.doc.com. created
services.org_dir.doc.com. created
timezone.org_dir.doc.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.doc.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 *Solaris Naming 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, 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
```

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.doc.com.
rootmaster# /usr/lib/nis/nisaddent -f /etc/shadow.xfr shadow sales.doc.com.
```

Note - When creating an 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.

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 an automounter table 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.doc.com.key-value  
master# nisaddent -f /etc/auto_home.xfr \  
-t auto_home.org_dir.doc.com.key-value sales.doc.com.
```

Loading Data From an NIS Map

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*. Note that `/var/yp/nisdomain` must be *local* files.

NIS+ Table Type	NIS Map Name
Hosts	hosts.byaddr
Passwd	passwd.byname
Group	group.byaddr
Ethers	ethers.byname
Netmasks	netmasks.byaddr
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-doc` domain:

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd
```

This example does the same thing, but for the `sales.doc.com.` domain instead of the local domain, `doc.com.`

```
rootmaster# /usr/lib/nis/nisaddent -y old-doc passwd sales.doc.com.
```

If the NIS+ table is an automounter table 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-doc \  
-t auto_home.org_dir.doc.com. key-value  
rootmaster# nisaddent -y old-doc \  
-t auto_home.org_dir.doc.com. key-value sales.doc.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.doc.com.
```

If the NIS map is an automounter map or a non standard map, combine the `-Y` option with the `-t` option:

```
rootmaster# nisaddent -Y auto_home  
-t auto_home.org_dir.doc.com. key-value  
rootmaster# nisaddent -Y auto_home  
-t auto_home.org_dir.doc.com. key-value sales.doc.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:

```
rootmaster# nisaddent -Y auto_home  
-t auto_home.org_dir.doc.com. key-value  
rootmaster# nisaddent -Y auto_home  
-t auto_home.org_dir.doc.com. key-value sales.doc.com.
```

Server-Use Customization

This chapter describes how to customize and control which servers NIS+ clients use.

- “NIS+ Servers and Clients” on page 241
- “NIS+ Over Wide Area Networks” on page 242
- “Optimizing Server-Use—Overview ” on page 243
- “Using the `nisprefadm` Command ” on page 247
- “Viewing Current Server Preferences” on page 249
- “How to Specify Preference Rank Numbers ” on page 250
- “Specifying Global Server Preferences ” on page 250
- “Specifying Local Server Preferences ” on page 253
- “Modifying Server Preferences ” on page 254
- “How to Remove Servers From Preference Lists ” on page 255
- “How to Replace an Entire Preferred Server List ” on page 256
- “Specifying Preferred-Only Servers” on page 256
- “Ending Use of Server Preferences” on page 258
- “Putting Server Preferences Into Immediate Effect ” on page 260

NIS+ Servers and Clients

When client machines, users, applications, or processes need NIS+ information, they seek an active NIS+ server (master or replica) from which to get the needed data. On large networks, networks with many subnets, and networks that span wide-area links, you may be able to improve NIS+ performance by customizing server usage.

Default Client Search Behavior

By default, if no server preferences have been set with the `nisprefadm` command, a client will first try to obtain the information it needs from an NIS+ server on the client's local subnet. If the client finds an active server on the local subnet, it obtains the information it needs from the first local server that responds. If no server is available on the local subnet, the client searches outside the local subnet, and obtains the NIS+ information it needs from the first remote server that responds.

On large, busy networks, this default search behavior may reduce NIS+ performance for one of two reasons:

- When multiple servers on a subnet are serving a large number of clients, the random nature of the client's default search pattern may result in some servers being over worked while others are under used.
- When a client has to seek an NIS+ server beyond the local subnet, it will obtain its information from the first server that responds even if that server is overworked, or linked to the client's subnet by a slower Wide Area Network connection such as a modem or a dedicated line that is already carrying heavy traffic.

Designating Preferred Servers

The Solaris 2.6 release contains a new feature—server-use customization—that allows you to control the order in which clients search for NIS+ servers. With this new feature you can balance and customize server usage by:

- Specifying that clients prefer (search for) certain servers over others.
- Specify whether or not clients are permitted to use remote servers if no local servers are available.

The search criteria that you specify can be applied to all clients within a domain, all clients on a subnet, or to individual clients on a machine-by-machine basis.

Note - When server-use preferences are set for a particular machine, those preferences apply to all users, applications, processes, or other clients running on that machine. You cannot set different server-use patterns for different clients on the *same* machine.

NIS+ Over Wide Area Networks

Server-use customization is particularly valuable for large networks with many subnets and networks that span multiple geographic sites connected by modems or leased lines. To maximize network performance, you want to minimize network

traffic between subnets, and between sites linked by slower connections. You can do that by specifying which NIS+ servers the clients can use, and their order of server preference. In this way you confine as much NIS+ network traffic as possible to the local subnet.

Optimizing Server-Use—Overview

This section provides an overview of server-use customization.

`nis_cachemgr` is Required

Server-use customization requires that a client be running `nis_cachemgr`. If a client machine is not running `nis_cachemgr`, it cannot make use of server-use customization. If there is no `nis_cachemgr` running on a client machine, that client will use the first server it identifies as described in “Default Client Search Behavior” on page 242.

Global Table or Local File

Depending on how you use the `nisprefadm` command, it creates either a local `client_info` file or a domain `client_info` table:

- **File.** You can use `nisprefadm` to create a local, machine-specific `client_info` file that is stored in the machine's `/var/nis` directory. A local file specifies server preferences for that machine only. When a machine has a local `/var/nis/client_info` file, it ignores any server preferences contained in a domain `client_info.org_dir` table. To create a local `client_info` file, you run `nisprefadm` with the `-L` option.
- **Table.** You can use `nisprefadm` to create an NIS+ `client_info` table which is stored in each domain's `org_dir` NIS+ directory object. This table can specify server preferences for:
 - Individual machines. (If a machine has a local `/var/nis/client_info` file, any preferences for that machine that happen to be in the domain `client_info` table are ignored.)
 - All the machines on a particular subnet. (If a machine on the subnet has a local `/var/nis/client_info` file or individual preferences set for it in the table it ignores subnet preferences.)

To create a global `client_info` table that applies to all machine on a subnet, you run `nisprefadm` with the `-G` and `-C` options as described in “Specifying Global Server Preferences ” on page 250.

Note that if a machine has its own local `client_info` file as described below, it will ignore all server preferences set for it in a global `client_info` table. If a machine has either a local `client_info` file or a machine-specific entry for it in the global `client_info` table, it will ignore preferences set for its subnet.



Caution - Use only the `nisprefadm` command to make changes to `client_info` files and tables. Never use other NIS+ commands such as `nistbladm`.

When working with `client_info` tables or files, you *must* use either the `-G` or the `-L` option to specify that your command apply to either the global table (`-G`) or local file (`-L`) of the machine you are running the command on.

Preference Rank Numbers

Server preferences are controlled by giving each server a *preference rank number*. Clients search for NIS+ servers in order of numeric preference, querying servers with lower preference rank numbers before seeking servers with higher numbers.

Thus, a client will first try to obtain namespace information from NIS+ servers with a preference of zero. If there are no preference=0 servers available, then the client will query servers whose preference=1. If no 1's are available, it will try to find a 2, and then a 3, and so on until it either gets the information it needs or runs out of servers.

Preference rank numbers are assigned to servers with the `nisprefadm` command as described in “Specifying Global Server Preferences ” on page 250.

Server preference numbers are stored in `client_info` tables and files. If a machine has its own `/var/nis/client_info` file, it uses the preference numbers stored in that file. If a machine does not have its own `client_info` file, it uses the preference numbers stored in the domain's `client_info.org_dir` table. These `client_info` tables and files are called “preferred server lists” or simply *server lists*.

You customize server usage by controlling the server preferences of each client. For example, suppose a domain has a client machine named `mailer` that makes heavy use of namespace information and the domain has both a master server (`nismaster`) and a replica server (`replica1`). You could assign a preference number of 1 to `nismaster` and a number of 0 to `replica1` for the `mailer` machine. The `mailer` machine would then always try to obtain namespace information from `replica1` before trying `nismaster`. You could then specify that for all the other machines on the subnet the `nismaster` server had a preference number of zero and `replica1` the number 1. This would cause the other machine to always try `nismaster` first.

You can give the same preference number to more than one server in a domain. For example, you could assign both `nismaster1` and `replica2` a preference number of 0, and assign `replica3`, `replica4`, and `replica5` a preference number of 1.

Default Server Preferences

If there is no `client_info` file or table, the cache manager automatically assigns all servers on the local subnet a default preference number of zero (0) and all servers outside the local subnet a preference of infinite. The purpose of `nisprefadm` is to change these default preference numbers to what you want them to be.

Efficiency and Server Preference Numbers

A client must seek all servers with a given preference number before searching for servers with the next higher number. It requires 5 or more seconds for a client to search for all the servers with a given preference number. This means that if you have a master server and 4 replicas in a domain, and you give each one a *different* preference number from 0 to 4, it could take a client more than 25 seconds to run through all of those preference levels.

To maximize performance, you should not use more than two or three levels of server preference. For example, in the case described above, it is better to give one of those five servers a preference=0 and all the others a preference of 1, or give two of them a preference of 1 and the remaining three a preference of 2.

Preferred Only Servers Versus All Servers

Server lists also specify what a client does if it cannot find *any* preferred servers. A *preferred server* is any server with a preference of zero, or any server that you have assigned a preference number with `nisprefadm`.

By default, if a client fails to reach a preferred server, it will then seek out any server it can find anywhere on the network using the search mode described in “Default Client Search Behavior” on page 242. You can change this default behavior with the `nisprefadm -o` option to specify that a client can only use preferred servers and if no servers are available it cannot go to non-preferred servers. See “Specifying Preferred-Only Servers” on page 256 for details.

Note - This option is ignored when the machine's domain is not served by *any* preferred servers.

Viewing Preferences

To view the server preferences currently in effect for a particular client machine, you run `nisprefadm` with the `-l` option as described in “Viewing Current Server Preferences” on page 249.

Server and Client Names

When specifying server or client machines, keep in mind the following points:

- Server and client names do *not* need to be fully qualified so long as they are in the same NIS+ domain and uniquely identify the object. You can simply use the machine name by itself.
- If a server or subnet is in another NIS+ domain, you need to include enough of the domain name to uniquely identify that machine. For example, if you are in the `sales.doc.com` domain and you need to specify the `nismaster2` machine in the `manf.doc.com` domain, you need only enter `nismaster2.manf`.

Server Preferences

To specify a server preference for:

- *Individual client machine*, use the `-L` option to create a local `client_info` file for the machine you are running the `nisprefadm` on. Use the `-G -C machine` options to create machine-specific preferences in the global `client_info` table.
- *All machines on a subnet*, use the `-G -C subnetnumber` option.
- *All machines in the current domain that do not have machine-specific or subnet-specific preferences*, use the `-G` option.

When Server Preferences Take Effect

Changes you make to a machine or subnet’s server preferences normally do not take effect on a given machine until that machine updates its `nis_cachemgr` data. When the `nis_cachemgr` of a machine updates its server-use information depends on whether the machine is obtaining its server preferences from a global `client_info` table or a local `/var/nis/client_info` file (see “Global Table or Local File ” on page 243):

- *Global table*. The cache managers of machines obtaining their server preferences from global tables update their server preferences whenever the machine is booted or whenever the Time-to-live (TTL) value expires for the `client_info` table. By default, this TTL value is 12 hours, but you can change that as described in “Changing the Time-to-Live of an Object” on page 207.

- *Local file.* The cache managers of machines obtaining their server preferences from local files update their server preferences every 12 hours or whenever you run `nisprefadm` to change a server preference. (Rebooting the machine does not update the cache manager's server preference information.)

However, you can force server preference changes to take effect immediately by running `nisprefadm` with the `-F` option. The `-F` option forces `nis_cachemgr` to immediately update its information. See “How to Immediately Implement Preference Changes ” on page 260for details.

Using the `nisprefadm` Command

The following sections describe how to use the `nisprefadm` command to set, modify, and delete server preferences.

The `nisprefadm` command is used to specify the servers that clients are to prefer.

The `nisprefadm` command has the following syntax:

```
nisprefadm -a|-m|-r|-u|-x|-l -L|-G [-o type] \
[-d domain] \
[-C machine] \
servers
nisprefadm -F
```

TABLE 14-1 `nisprefadm` Command Options

Option	Description
<code>-G</code>	Create a global <code>client_info</code> table stored in the domain's <code>org_dir</code> directory. In other words, create a global preferred server list. This option must be used with either <code>-C subnet</code> to specify preferences for all the machines on a given subnet, or <code>-C machine</code> to specify preferences for an individual machine.
<code>-L</code>	Create a local <code>client_info</code> file stored in the local machine's <code>/var/nis</code> directory. In other words, create a preferred server list that applies only to the machine you are running the command on.
<code>-o type</code>	Specify an option. The valid options are: <code>pref_type=all</code> , which specifies that clients can use non-preferred servers if no preferred servers can be contacted, and <code>pref_type=pref_only</code> , which specifies that clients may only use the designated preferred servers.

TABLE 14-1 nisprefadm Command Options *(continued)*

Option	Description
-d <i>domain</i>	Create a global preferred server client_info table for the specified domain or subdomain.
-C <i>subnet</i>	The number of a subnet to which the preferences will apply.
-C <i>machine</i>	The name of a client machine.
<i>servers</i>	One or more NIS+ servers. These are the servers that are to be preferred.
-a	Add the specified servers to the server list.
-m	Modify the server list. For example, you can use the -m option to change the preference number given to one or more servers.
-r	Remove the specified servers from the server list.
-u	Clear the server list, and then add the specified servers. (In other words, replace the current server list with a new list of preferred servers.)
-x	Remove the server list completely.
-l	List (display) the current preferred server information.
-F	Force changes to a preferred server list to take effect immediately.

Note - The `-C` machine option should not be used with the `-L` (local) flag because it has no effect. For example, suppose you are running `nisprefadm` on the `altair` machine. You use the `-L` flag to specify that the preferences you are specifying be written into `altair`'s local `client_info` file. You also use a `-C` `vega` option to specify that the preferences you are creating be applied to the `vega` machine. The `nisprefadm` command then write your preferences for `vega` into `altair`'s file. But `vega` will never see them because `vega` will always get its server preferences from either its own local `client_info` file or the domain's global `client_info` table. Thus, it only makes sense to use the `-C` option when running `nisprefadm` with the `-G` (global) flag

Viewing Current Server Preferences

To view current server preferences, run `nisprefadm` with the `-l` option.

How to View Preferences for a Machine

- ◆ Run `nisprefadm` with the `-L` and `-l` options on the machine.

```
sirius# nisprefadm -L -l
```

This displays any server preferences defined in the machine's local `/var/nis/client_info` file. If there is no local file, no information is displayed and you are returned to your shell prompt.

How to View Global Preferences for Single Machine

- ◆ Run `nisprefadm` with the `-l`, `-G` and `-C` *machinename* options.

```
sirius# nisprefadm -G -l -C machinename
```

Where *machinename* is the IP address (number) of the machine.

This displays the preferences set in the domain's global `client_info` table for that machine.

How to View Global Preferences for a Subnet

- ◆ Run `nisprefadm` with the `-l`, `-G` and `-C subnet` options.

```
sirius# nisprefadm -G -l -C subnet
```

Where *subnet* is the IP address (number) of the subnet.

This displays the preferences set in the domain's global `client_info` table for that machine.

How to Specify Preference Rank Numbers

By default, all servers listed after the `-a` option are given a preference number of zero. To specify a different preference number, enclose the number in parentheses immediately after the server name like this: `-a name(n)`. Where *name* is the name of the server and *n* is the preference number.

For example, assign the `replica2` server a preference number of 3:

```
# nisprefadm -G -a replica2(3)
```

Note - With some shells you may have to enclose the element in quotes like this: `"name(n)"`.

See "Preference Rank Numbers" on page 244 for background information on the server preference rank numbers.

Specifying Global Server Preferences

You can set global server preferences for a local or remote domain. Preferences may be set for individual machines and all the machines on a subnet.

The procedures in this section describe how to specify server preferences in a global `client_info` table residing on the NIS+ domain's master server. Once the table exists on the master server, NIS+ replicates it on to any existing replica servers for the domain.

- See “Specifying Local Server Preferences ” on page 253 for information on how to create a local `client_info` file on an individual machine.
- See “Global Table or Local File ” on page 243 for an explanation of the difference between a global `client_info` table and a local `client_info` file.

To assign server preference numbers, run `nisprefadm` with either the:

- `-a` option to add new or additional preferred servers.
- `-u` option to delete existing server preferences and create new ones.

How to Set Global Preferences for a Subnet

To assign server preferences in the global table for all the machines on a subnet:

- ◆ Run `nisprefadm` with the `-G` and `-C` *subnet* options.

```
#nisprefadm -G -a -C subnet servers (preferences)
```

Where:

- `-C` *subnet* identifies the IP number of the subnet the preferences will apply to.
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to specify that the subnet `123.123.123.123` use the `nismaster` and `replica3` servers with default preference number `s` of zero and the `manf.replica6` server with a preference number of 1:

```
polaris# nisprefadm -a -G -C 123.123.123.123 nismaster1 \  
replica3 "manf.replica6(1)"
```

How to Set Global Preferences for an Individual Machine

- ◆ Run `nisprefadm` with the `-G`, and `-C` *machine* options.

```
#nisprefadm -G -a -C machine servers (preferences)
```

Where:

- `-C machine` identifies the machine the preferences will apply to. (Depending on the shell you are using, you may need to enclose *machine* in quotes.)
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to replace the current preferences for the machine `cygnus` with `replica7` and `replica9` both with a default preference number of zero:

```
polaris# nisprefadm -u -G -C cygnus replica7 replica9
```

How to Set Global Preferences for a Remote Domain

To assign server preferences for an individual machine in a remote domain or all the machines on a subnet in a remote domain:

- ◆ **Run `nisprefadm` with the `-C`, `-G`, and `-d` options.**

```
#nisprefadm -a -G -C name \  
-d domain servers(preferences)
```

Where:

- *name* is the IP number of a subnet or the name of a machine. The modifications you make with this command apply to the subnet or machine that you name.
- *domainname* is the name of the remote domain.
- *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to add the `nismaster2` server with a default preference number of zero to the preferred server list of the `111.11.111.11` subnet in the remote `sales.doc.com` domain:

```
polaris# nisprefadm -a -G -C 111.11.111.11 -d sales.doc.com. nismaster2
```

Specifying Local Server Preferences

These procedures explain how to create or change a local `client_info` file that specifies server preferences for the machine on which it resides.

If a machine has a local `/var/nis/client_info` file, that machine takes its server preferences from its local file rather than the global `client_info` tables on NIS+ servers. In other words, a local file overrides any global table.

- See “Specifying Global Server Preferences ” on page 250 for information on how to create a global `client_info` tables for NIS+ servers.
- See “Global Table or Local File ” on page 243 for an explanation of the difference between a global `client_info` table and a local `client_info` file.

To assign server preferences, run `nisprefadm` with either the:

- `-a` option to add new or additional preferred servers.
- `-u` option to delete existing server preferences and create new ones.

How to Set Preferences on a Local Machine

To assign server preferences for the local machine that you are running the `nisprefadm` command on:

- ◆ **Run `nisprefadm` with the `-L` option and either the `-a` or `-u` options.**

```
#nisprefadm -a -L servers(preferences)
```

Where *servers(preferences)* are one or more servers with optional preference ranking numbers.

For example, to specify that the `deneb` machine first seek NIS+ information from the `replica3` server with a default preference number of zero and then from the `replica6` server (with a preference number of 1) in the `manf.doc.com` domain:

```
deneb# nisprefadm -a -L replica3 replica6.manf(1)
```

Modifying Server Preferences

You can change a server's preference number and switch (replace) the preference numbers assigned to different servers.

To change preferred servers or the preference number assigned to a server, run `nisprefadm` with the `-m oldserver==newserver(n)` option.

How to Change a Server's Preference Number

- ◆ Run `nisprefadm` with the `-m server=server(new)` option.

```
#nisprefadm -L|-G -C name -m oldserver=newserver(n)
```

Where:

- `-L|-G` determines whether you are modifying a local file or a global table.
- `-C name` is the IP number of a subnet or the name of a machine. This option is only used when you are also using the `-G` option. The modifications you make with this command apply to the subnet or machine that you name.
- `-m` is the modify server list option.
- `old server` is the name of the server whose preference number you want to change.
- `new server(n)` is the server name and its new preference number.

For example, on the `deneb` machine, to change the number given to the `replica6.manf` server to 2 in `deneb`'s local `client_info` file:

```
deneb# nisprefadm -L -m replica6.manf=replica6.manf(2)
```

How to Replace One Server With Another in a Preference List

To change one server for another in a preference list:

- ◆ Run `nisprefadm` with the `-m oldserver=newserver` option.

```
#nisprefadm -L|-G -C name -m \  
oldserver=newserver(preference)
```

Where:

- `-L|-G` determine whether you are modifying a local or a domain-wide server list.
- `-C name` is the IP number of a subnet or the name of a machine. This option is only used in when you are also using the `-G` option. The modifications you make with this command apply to the subnet or machine that you name.
- `-m` is the modify-server-list option.
- `oldserver` is the old server you are replacing.
- `newserver(prefnumber)` is the new server (with an optional preference number) that is taking the old server's place in the preferred server list.

Keep in mind that when you replace a server in a global `client_info` table using the `-G` option, the replacement only applies to the subnet or machine identified by the `-C` option. Other listings of the replaced server are not affected.

For example, suppose you have a domain with three subnets, and the `replica1` server is listed as a preferred server for two of those subnets. If `replica1` is obsolete and you take it out of service, you then run `nisprefadm -m` to replace it with the new server for the first subnet. Until you do the same for the second subnet, `replica1` is still listed as a preferred server for that subnet. The same principle applies to preferred servers for individual machines.

For example, to replace the `replica3` server with the `replica6` server for subnet `123.12.123.12` in the domain's global `client_info` table and assign `replica6` a preference number of 1:

```
nismaster# nisprefadm -G -C 123.12.123.12 -m replica3 replica6(1)
```

How to Remove Servers From Preference Lists

To remove one or more servers from a preference list:

- ◆ **Run `nisprefadm` with the `-r` option.**

```
#nisprefadm -L|-G -C name -r servers
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.

- `-C name` is the IP number of a subnet or the name of a machine. This option is only used when you are also using the `-G` option. The preferred servers you remove with this command apply to the subnet or machine that you name.
- `-r` removes the named *servers* from the list.

For example, in the domain's global `client_info` table, to remove the `replica3` and `replica6.manf` servers for the machine `polaris`:

```
polaris# nisprefadm -G -C polaris -r replica3 replica6.manf
```

How to Replace an Entire Preferred Server List

To replace an entire list of preferred servers for a subnet or machine in either a global `client_info` table or a machine in its local `client_info` file, run `nisprefadm` with the `-u` option.

The `-u` option operates the same way as the `-a` option, except that `-u` first deletes any existing server preferences for the machine or subnet before adding the new ones that you specify. (If there are existing preferences, the `-a` option adds the new ones to the old list.)

See “How to Set Global Preferences for an Individual Machine ” on page 251 for an example using the `-u` option.

Specifying Preferred-Only Servers

You can specify what clients do when no preferred servers are available.

By default, if a client cannot reach a preferred server, it uses whatever other server it can find. You can specify that clients may only use preferred servers by setting the preferred-only option. See “Preferred Only Servers Versus All Servers” on page 245 for background information on the preferred-only and all servers options.

To specify what clients do when no preferred servers are available, run `nisprefadm` with the `-o value` option.

How to Specify Preferred-Only Servers

To specify that clients using a server list may only obtain NIS+ information from servers named in the list:

- ◆ **Run** `nisprefadm` **with the** `-o pref_only` **option.**

```
#nisprefadm -L|-G -o pref_only
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.
- `-o -pref_only` specifies that clients can only obtain NIS+ information from servers on the list.

Note - This option is ignored for domains that are not served by *any* preferred servers.

For example, to specify in `altair`'s local `client_info` file that `altair` must always use preferred servers and cannot use any server not on `altair`'s preferred server list:

```
altair# nisprefadm -L -o pref_only
```

How to Revert to Using Non-Preferred Servers

To specify that clients using a server list may obtain NIS+ information from servers not named in the list if no preferred servers are available:

- ◆ **Run** `nisprefadm` **with the** `-o all` **option.**

```
#nisprefadm -L|-G -o all
```

Where:

- `-L|-G` determines whether you are modifying a local or a domain-wide server list.
- `-o -all` specifies that clients may obtain NIS+ information from servers not on the list if no preferred servers are available.

Note - This is the default behavior. You only need to use the `-o all` option if you have previously specified preferred-only servers with the `-o pref_only` option.

For example, to specify in `altair`'s local `client_info` file that `altair` can now use non-preferred servers if no preferred servers can be reached:

```
altair# nisprefadm -L -o all
```

Ending Use of Server Preferences

You can stop using server-use customization and revert to the obtaining NIS+ information as described in “Default Client Search Behavior” on page 242.

To end server preferences, run `nisprefadm` with the `-x` option.

Note - When you end server preferences, clients do not stop using server preferences until the normal course of events as described “When Server Preferences Take Effect” on page 246. You can force an immediate end to server preferences as described in “Putting Server Preferences Into Immediate Effect ” on page 260.

How to Eliminate Global Server Preferences

- ◆ **Run `nisprefadm` with the `-G` and `-x` options.**

```
#nisprefadm -G -x
```

This eliminates global server preferences.

- Client machines that do not have local server preferences will obtain NIS+ information as described in “Default Client Search Behavior” on page 242.
- Client machines that do have local server preferences set by a local `/var/nis/client_info` file will continue to use servers as specified in that file.

How to Eliminate Local Server Preferences

Ending local preferences can mean one of three different things:

- That you want the machine to stop using its local `client_info` file for its server preferences and start using the preferences set for its subnet in the domain’s global `client_info` table.
- That you want this machine to stop using its local `client_info` file for its server preferences and start using the preferences set for it specifically in the domain’s global `client_info` table.
- That you do not want the machine to use server preferences at all. When a machine does not use server preferences, it obtains NIS+ information as described in “Default Client Search Behavior” on page 242.

How to Switch from Local to Global Subnet Preferences

- ◆ **Remove the machine's `/var/nis/client_info` file.**

```
# rm /var/nis/client_info
```

This causes the machine to use the preferences specified for the machine's subnet in the domain's global `client_info` table.

How to Switch from Local to Machine-Specific Global Preferences

1. **Remove the machine's `/var/nis/client_info` file.**

```
# rm /var/nis/client_info
```

2. **Specify preferences for the machine in the global table using the `-G` and `-C` options.**

See “How to Set Global Preferences for an Individual Machine ” on page 251.

How to Stop a Machine from Using Any Server Preferences

1. **Remove the machine's `/var/nis/client_info` file.**

```
# rm /var/nis/client_info
```

If the machine's domain does not have a global `client_info` table, this step is all you have to do. If the domain does have a `client_info` table, continue on to the next step.

2. **Create an empty `/var/nis/client_info` file.**

```
# touch /var/nis/client_info
```

When a machine has its own `/var/nis/client_info` file, it does not use global preferences from any `client_info` table. If the machine has an empty `/var/nis/client_info` file, it will not use any preferences at all and will obtain NIS+ information as described in “Default Client Search Behavior” on page 242.

Putting Server Preferences Into Immediate Effect

Server-use changes normally go into effect whenever the client machine is rebooted or updates its cache manager.

When you use `nisprefadm` to set or change server preferences on a local machine using a local `client_info` file (the `-L` option), your changes go into effect immediately.

For machines obtaining their server preferences from a global `client_info` table (the `-G` option) you can force server preference changes into immediate effect by running `nisprefadm` a particular with the `-F` option.

```
# nisprefadm -F
```

The `-F` option forces the machine's cache manager to immediately update its server preference information from the domain's global `client_info` table. (If the machine on which you run `nisprefadm -F` has its own local `client_info` file in `/var/nis`, running `nisprefadm -F` on it will have no effect.)

Note - You cannot use the `-F` option with any other `nisprefadm` options. The `nisprefadm -F` command must always be run by itself on the machine you want it to apply to. You cannot use the `-G` option to update the cache managers of all machines in a domain. The `nisprefadm -F` command must be run on each machine individually.

How to Immediately Implement Preference Changes

To force a newly created or modified server list into immediate effect on a given machine:

- ◆ **Run `nisprefadm` with the `-F` option on that machine.**

```
# nisprefadm -F
```

For example, to force immediate implementation of changes to `vega`'s preferred server list (whether local or global):

```
vega# nisprefadm -F
```

NIS+ Backup and Restore

This chapter describes how to backup and restore an NIS+ namespace.

- “Backing Up Your Namespace With `nisbackup` ” on page 261
- “Restoring Your NIS+ Namespace With `nisrestore`” on page 267
- “Using Backup/Restore to Set Up Replicas ” on page 269
- “Replacing Server Machines ” on page 270

The NIS+ backup and restore capabilities provide a quick and easy method of preserving and reinstalling your NIS+ namespace. These features can also be used to simplify creation of new replica servers and reduce the time it takes to bring them online. These tasks are performed by two commands:

- `nisbackup`. Backups up NIS+ directory objects
- `nisrestore`. Restores NIS+ directory objects.

Backing Up Your Namespace With `nisbackup`

The `nisbackup` command backups up one or more NIS+ directory objects or an entire namespace to a specified UNIX file system directory.

Note - The `nisbackup` command is always run on a master server. Never run it on a replica server.

The `nisbackup` command copies the NIS+ namespace data set as of the time the backup command is run. This recording includes all current NIS+ data *and also* any

changes entered into the NIS+ namespace by an authorized network administrator but not yet checkpointed (posted) to the NIS+ tables. The back up operation does not check or correct NIS+ data. If data in a table is corrupt, the corrupt data is backed up as if it were valid data.

The `nisbackup` command only backs up those directory objects that the machine is master server for. In other words, you can only use `nisbackup` on a master server. You cannot use `nisbackup` on a replica server.

- If a machine is a master server for both a domain and a subdomain's NIS+ directory objects, you can run `nisbackup` on that machine to back up both domain and subdomain directory objects.
- However, if a machine is a master server one directory object, and a replica server for a different directory object, you can run `nisbackup` to back up the directory object that the machine is master server for, but it will not back any objects that the machine is only a replica server for.

If the backup process is interrupted or unable to successfully complete its operation it halts and restores all previous backup files that were stored in the target directory.

`nisbackup` Syntax

The `nisbackup` command uses the following syntax:

```
nisbackup [-v][-a] backupdir objects
```

Where:

- *Backupdir* is the target directory where the backup files are to be stored. For example, `/var/master1_backup`.
- *Objects* are the NIS+ directory objects that you want to back up. For example, `org_dir.doc.com`. Multiple NIS+ directory objects can be listed separated by spaces.

The `nisbackup` command takes the following options:

TABLE 15-1 Options for the nisbackup Command

Option	Purpose
-v	Verbose mode. This mode provides additional information
-a	All. Backs up all NIS+ directory objects that the server is master of. This includes any sub-domain directory objects that this server is the master for. Note that directory objects of subdomains that have their own master servers will not be backed up.

The `nisbackup` command must be run on the master server for the NIS+ directory objects you are backing up.

When specifying NIS+ directory objects to be backed up, you can use full or partially qualified directory names.

When you back up multi-level directories, the backup files for lower level directories are automatically placed in subdirectories of the target backup directory.

What nisbackup Backs Up

When using `nisbackup`, keep in mind that `nisbackup` is server specific. Regardless of whether or not you use the `-a` option, `nisbackup` only backs up those directories that the server you are running it on is master of. NIS+ directory objects that have some other master server will not be backed up.

For example, suppose the `submaster1` server is master server for the `sales.doc.com.` directory objects and also a replica for the `west.sales.doc.com.` directory objects. When you run `nisbackup` on `submaster1`, only the `sales.doc.com.` directory objects will be backed up.

Some of the implications of this server specific principle are:

- *Entire NIS+ namespace.* If you want to perform an NIS+ back up for an entire multi-domain namespace, *and* your root master server is *also* the master server of all subdomains, you can run `nisbackup` on the root master with the `-a` option. However, if the root master server is *not* the master server of all subdomains, you must also run `nisbackup` on each of the other master servers in order to obtain a complete back up of your entire namespace.
- *Sub-domains.* If you are performing an NIS+ back up for one or more sub-domains, you must run `nisbackup` on the subdomain's master server. If one machine, such as the root master, is also master of one or more subdomains, you can run `nisbackup` on that machine with the `-a` option.

- *FNS ctx_dir*. If you are running FNS, `nisbackup` will only back up your `ctx_dir` directories if you run it on the `ctx_dir` master server and either specify that the `ctx_dir` be backed up or use the `-a` option. If, as is common practice, your `ctx_dir` and NIS+ directory objects are served by different master servers, you must run `nisbackup` on both machines to back up all directories.

The Backup Target Directory

While the backup target directory must be available to the server being backed up, it is good practice to use a target directory that is not physically mounted on the server. That way you ensure that if the server is damaged the backup directory is still available.

A separate target directory must be used for each master server being backed up. It is good practice to avoid confusion by including the master server's machine name in the target directory name. For example, the target directory for a `nisbackup` run on the `master1` machine might be named `/var/master1_backup`



Caution - Never back up more than one master server to a given target directory. Always use different target directories for different master servers. This is because each time you backup one or more NIS+ directory objects to a given target directory, previous backup files for those NIS+ directory objects in that directory are overwritten.

Maintaining a Chronological Sequence of NIS+ Backups

There are at least two ways to maintain an historic sequence of backup files:

- *Different target directories*. You can maintain separate target directories for each date's backup. For example, `/var/master1_backup/July14`, and `/var/master1_backup/July15`, and so on. While this method is simple it wastes disk storage space.
- *File system backup*. The most common method of maintaining an historical sequence of NIS+ backups is to simply include the backup target directory in whatever regular file system backup method that you use. To facilitate this, the `nisbackup` command can be run from a `crontab` file, or from within the Solstice backup routine. See your Solstice documentation for information on how to specify that commands like `nisbackup` be automatically run as part of the system backup procedure.

Backing Up Specific NIS Directories

To back up specific NIS+ directory objects, you list those directories after the target backup directory.

For example, to backup the three `org_dir` directory objects for the root, sales, and manf domains to a `/master1_backup` directory, you would run `nisbackup` on the master1 machine as follows:

```
master1# nisbackup /var/master1_backup org_dir org_dir.sales org_dir.manf
```

Backing Up an Entire NIS+ Namespace

To back up an entire NIS+ namespace you run the `nisbackup` command on the root master server with the `-a` option.

When you use the `-a` option, you do not specify the NIS+ directory objects to be backed up. All NIS+ directory objects on the server and all those of subdomains below it will be automatically backed up.

For example, to backup the `doc.com.` namespace to a `/master1_backup` directory, you would run `nisbackup` on the root master as follows:

```
rootmaster# nisbackup -a /var/master1_backup
```

Backup Directory Structure

When you perform a back up on a domain, a subdirectory for each NIS+ directory object is created in the backup target directory. The names of these subdirectories match the fully qualified NIS+ directory object name including the trailing period.

If you perform a full backup of an entire NIS+ object using the `-a` option, then all three of the associated directory objects (*domain.org_dir.domain.*, and *groups_dir.domain.*) are backed up and three target subdirectories are created. If you are backing up multiple objects, subdirectories are created for every object that you are backing up.

Note that the backup subdirectories for multiple NIS+ directory object are all subdirectories of the parent target backup directory regardless of whether or not they are subdomains. In other words, `nisbackup` does not reproduce a domain hierarchy under the parent backup target directory, instead all of the back up subdirectories are simple subdirectories of the target directory.

For example, if you backed up the root, sales, and manf directory objects of `doc.com.` to a `/var/master1_backup` directory, nine subdirectories would be created in the `/var/master1_backup` directory as shown in Figure 15-1:

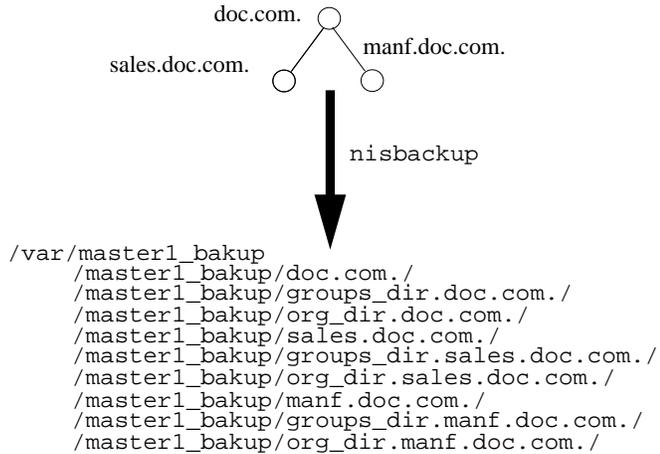


Figure 15-1 Example directories created by nisbackup

Backup Files

The backup target directory contains a `backup_list` file that lists the NIS+ directory objects most recently backed up to this target directory.

Each of the subdirectories contain two files and a `/data` subdirectory. The three files are:

- `data.dict.` An XDR encoded file containing an NIS+ data dictionary for the NIS+ directory objects backed up to this directory.
- `last.upd.` A binary file containing time-stamp information about the NIS+ directory object backed up to this directory.

Each of the `/data` subdirectories contain one or more of the following files:

- `root.object.` An XDR encoded data file containing a description of the NIS+ root directory object. For example, `/master1_bakup/doc.com/data/root.object.`
- `root_dir.` An XDR encoded file containing a description of NIS+ objects contained in the root directory and server information for those objects. For example, `/master1_bakup/doc.com/data/root_dir.`
- `table.directory.` An XDR encoded file containing the data that was present in an NIS+ table at the time the backup was performed *and also* any data contained in any associated NIS+ log files. If there is an NIS+ table in the NIS+ directory object being backed up, a corresponding `table.directory` backup file will be created in the `/data` subdirectory for that directory object.

For example, every NIS+ `org_dir` directory contains a `hosts` table so there will be a `hosts.org_dir` file in each `target/org_dir.domain/data` subdirectory. For example, `/master1_bakup/org_dir.doc.com./data/hosts.org_dir`

User-created NIS+ tables present in a given directory object are backed up in the same way as the standard NIS+ tables.

- `groups_dir`. An XDR encoded file containing NIS+ groups information. This file is stored in the corresponding NIS+ `groups_dir` target directory.

Restoring Your NIS+ Namespace With `nisrestore`

The `nisrestore` command recreates NIS+ directory objects to match the data stored in backup files created with the `nisbackup` command. This command can be used to restore NIS+ servers, replace directory objects that have become corrupted, or download NIS+ data on to a new NIS+ server.

Prerequisites to Running `nisrestore`

In order to use `nisrestore` the target machine that will be receiving the NIS+ data from `nisrestore` must have already been set up as an NIS+ server. (See *Solaris Naming Setup and Configuration Guide* for a detailed description of setting up NIS+ servers.) This means that:

- The machine must have already been initialized as an NIS+ client.
- If the machine will be running in NIS-compatibility mode and support DNS forwarding, it must have a properly configured `/etc/resolv.conf` file.
- If you are using `nisrestore` on a server while other servers in the namespace are up and running, `nisrestore` will verify with those other servers that this server is configured to serve the backed up NIS+ objects that you are restoring to it. If no other servers are up and running in your namespace, then you must run `nisrestore` with the `-f` option. In other words, if there are other servers that `nisrestore` can check with, you do not need to use the `-f` option. If no other servers are available, for example if you are restoring a single master server and there are no functioning replica servers, then you must use the `-f` option.



Caution - In addition to the three pre-requisites listed above, the `rpc.nisd` daemon must *not* be running on the machine. If necessary, you must kill `rpc.nisd` before running `nisrestore`.

nisrestore Syntax

The `nisrestore` command uses the following syntax:

```
nisrestore [-fv][--a][--t] backupdir [directory_objects]
```

Where:

- *Backupdir* is the directory containing the backup files to be used to restore the NIS+ objects. For example, `/var/master1_backup`.
- *Directory_objects* are the NIS+ directory objects that you want to restore. For example, `org_dir.doc.com`. Multiple NIS+ directory objects can be listed separated by spaces. (If you run `nisrestore` with the `--a` option, you do not specify specific directory objects.)

The `nisrestore` command takes the following options:

TABLE 15-2 Options for the `nisbackup` Command

Option	Purpose
<code>--a</code>	All. Restores all of the NIS+ directory objects contained in the backup directory.
<code>--f</code>	Forces the restoration without validating that the server is listed in the directory object's serving list. This option must be used when restoring a root master server or if you get an "unable to lookup object" type of error.
<code>--v</code>	Verbose mode. This mode provides additional information
<code>--t</code>	This option lists all of the NIS+ directory objects stored in the backup directory. No restoration of objects takes place.

Using `nisrestore`

To restore NIS+ data from NIS+ backup files, use the `nisrestore` command.

For example, to restore the `org_dir.doc.com` directory object on the `replica1` server, you would log in as root on `replica1`, make sure that the prerequisites described in "Prerequisites to Running `nisrestore`" on page 267 have been met and then run `nisrestore` as shown below:

```
replica1# nisrestore /var/master1_backup org_dir.doc.com.
```

The following points apply to `nisrestore`:

- *Damaged namespace.* To restore a damaged or corrupted NIS+ namespace, the `nisrestore` command must be run on all of the servers for the NIS+ directory objects you are restoring.
- *Lookup error.* If you get an error message telling you that `nisrestore` cannot verify or look up needed data, then you must use the `-f` option.

For example, to reload NIS+ data on a root master server named `master1`, you would enter:

```
master1# nisrestore -f -a /var/master1_backup
```

- *Directory names.* When specifying the NIS+ directory objects to be restored, you can use full or partially qualified directory names.

Using Backup/Restore to Set Up Replicas

The NIS+ backup and restore features can be used to quickly download NIS+ data on to a new replica server. For large namespaces this is much faster than `nisping` to obtain data from the master server.

Using `nisbackup` and `nisrestore` to set up a new replica is described in detail in *Solaris Naming Setup and Configuration Guide*. Briefly, the steps are:

1. **Run `nisserver` on the master to create the new replica.**
2. **Kill `rpc.nisd` on the new replica server.**
This interrupts the automatic transfer for namespace data from the master to the replica using the `nisping` command.
3. **Run `nisbackup` on the master server.**
4. **Run `nisrestore` on the new replica to download the NIS+ data.**
5. **Restart `rpc.nisd` on the new replica**

Replacing Server Machines

You can use `nisbackup` and `nisrestore` to quickly replace a machine that you are using as a server with another machine. For example, you want to improve network performance by replacing an older server with a newer, faster machine.

Machine Replacement Requirements

To replace a machine being used as an NIS+ server with another machine, you *must*:

- Assign the new machine the same IP address as the older machine it is replacing.
- Assign the new machine the same machine name as the older machine it is replacing.
- Connect the new machine to the same subnet as the older machine it is replacing.

How to Replace Server Machines

To replace a server machine, follow these steps:

1. **Run `nisbackup` on the master server for the domain that the old server serves.**
See “Backing Up an Entire NIS+ Namespace ” on page 265. (Note that the old server you are replacing could be the master server for the domain, in which case you would run `nisbackup` on this old master server.)
2. **Copy the old server’s `/var/nis/NIS_COLD_START` file to the backup directory.**
3. **Copy the old server’s `/etc/.rootkey` file to the backup directory.**
4. **Disconnect the old server from the network.**
5. **Connect the new server to the network.**
6. **Assign the new server the same IP address (number) as the old server.**
7. **Assign the new server the same machine name as the old server.**
8. **If necessary, kill `rpc.nisd` on the new server.**
9. **Run `nisrestore` on the new server to download the NIS+ data.**
See “Restoring Your NIS+ Namespace With `nisrestore`” on page 267.

10. Copy the `.rootkey` file from the backup directory to `/etc` on the new server.

11. Copy the `NIS_COLD_START` file from the backup directory to `/var/nis` on the new server.

12. Reboot the new server.

Removing NIS+

This chapter describes how to use the NIS+ directory administration commands to remove NIS+ from clients, servers, and the namespace as a whole.

- “Removing NIS+ From a Client Machine” on page 273
- “Removing NIS+ From a Server” on page 274
- “Removing the NIS+ Namespace” on page 276

For information on disassociating an NIS+ replica server from a directory so that it no longer acts as a replica for that domain, see “The `nisrmdir` Command ” on page 192.

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 276 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 *Solaris Naming Setup and Configuration Guide*, simply run `nisclient` with the `-r` option:

```
client# 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 *Solaris Naming Setup and Configuration Guide*, perform the following steps:

1. **Remove the `.rootkey` file.**

```
client# rm -f /etc/.rootkey
```

2. **Locate and kill the `keyserv`, `nis_cachemgr`, and `nscd` processes.**

```
client# ps -ef | grep keyserv
root 714 1 67 16:34:44 ? keyserv
client# kill -9 714
client# ps -ef | grep nis_cachemgr
root 123 1 67 16:34:44 ? nis_cachemgr
client# kill -9 123
client# ps -ef | grep nscd
root 707 1 67 16:34:44 ? nscd
client# kill -9 707
```

3. **Remove the `/var/nis` directory and files.**

```
clientmachine# 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 276 for

information on removing the NIS+ name service from a network and returning to either NIS or `/etc` files for naming purposes.

Note - You can replace a machine that you are using as an NIS+ server with another machine. See “Replacing Server Machines ” on page 270.

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 273 or the NIS+ command set as described in “Removing NIS+ That Was Installed Using NIS+ Commands” on page 274.

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, `doc.com`.

3. Remove the root domain.

```
rootmaster# nisrmdir -f domainname
```

Where *domainname* is the name of the root domain, for example, `doc.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 key serv process.

```
rootmaster# key serv
```

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 **IV** Administering NIS

This part describes the *Network Information Service* (NIS) and how to administer it.

- Chapter 17
- Chapter 18

Network Information Service (NIS)

This chapter describes NIS, the Network Information Service.

- “NIS Introduction” on page 281
- “NIS Machine Types ” on page 284
- “NIS Elements” on page 285
- “NIS Binding” on page 292
- “Differences Between This and Earlier NIS Versions ” on page 294

See *Solaris Naming Setup and Configuration Guide* for information on how to initially setup and configure NIS.

NIS Introduction

NIS is a distributed name service. It is a mechanism for identifying and locating network objects and resources. It provides a uniform storage and retrieval method for network-wide information in a transport-protocol and media-independent fashion.

By running the service, the system administrator can distribute administrative databases, called *maps*, among a variety of servers (*master* and *slaves*), and update those databases from a centralized location in an automatic and reliable fashion to ensure that all clients share the same name service information in a consistent manner throughout the network. For additional overview and background information on NIS, see “NIS” on page 9.

NIS was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names

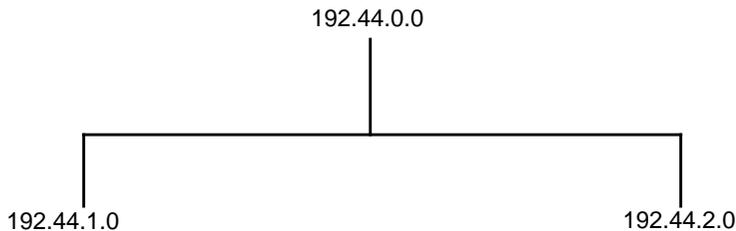
instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information not only about machine names and addresses, but also about users, the network itself, and network services. This collection of network *information* is referred to as the NIS *namespace*.

Note - In some contexts *machine* names are referred to as *host* names or *workstation* names. This discussion uses *machine*, but some screen messages or NIS map names may use *host* or *workstation*.

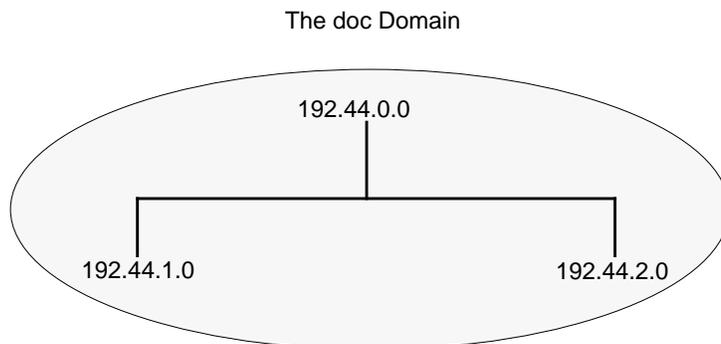
NIS Architecture

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

NIS uses domains to arrange the machines, 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:



A NIS domain cannot be connected directly to the Internet using just NIS. However, organizations that want to use NIS and also be connected to the Internet can combine NIS with DNS. You can use NIS to manage all local information and use

DNS for Internet host lookup. NIS provides a forwarding service that forwards host lookups to DNS if the information cannot be found in a NIS map. The Solaris operating system also allows you to set up the `nsswitch.conf` file so that hosts lookup requests go only to DNS, or to DNS and then NIS if not found by DNS, or to NIS and then DNS if not found by NIS. (See Chapter 2, for details.)

NIS and NIS+

Both NIS and NIS+ perform some of the same tasks. NIS+, however, allows for hierarchical domains, namespace security, and other features that NIS does not provide. For a more detailed comparison between NIS and NIS+, see “How NIS+ Differs From NIS” on page 29.

You can use NIS in conjunction with NIS+ under the following principles and conditions:

- Servers within a domain. While you can have both NIS and NIS+ servers operating in the same domain, doing so is not recommended for long periods. As a general rule, using both services in the same domain should be limited to a relatively short transition period from NIS to NIS+. If some clients need NIS service, you can run NIS+ in NIS compatibility mode as explained “Solaris 1.x Releases and NIS-Compatibility Mode” on page 33.
- Subdomains. If the master server of your root domain is running NIS+, you can set up subdomains whose servers are all running NIS. (If your root domain master server is running NIS, you cannot have subdomains.) This might be useful in situations where you are moving from NIS to NIS+. For example, suppose your enterprise had separate, multiple NIS domains, possibly at different sites. Now you need to link them all together into a single, hierarchical multi-domain namespace under NIS+. By first setting up the root domain under NIS+, you can then designate the legacy NIS domains as sub-domains that continue to run NIS until it is convenient to switch them over to NIS+.
- Machines within a domain.
 - If a domain’s servers are running NIS+, individual machines within that domain can be set up to use either NIS+, NIS, or `/etc` files for their name service information. In order for an NIS+ server to supply the needs of an NIS client, the NIS+ server must be running in NIS-Compatibility mode as described in *Solaris Naming Setup and Configuration Guide*.
 - If a domain’s servers are running NIS, individual machines within that domain can be set up to use either NIS or `/etc` files for name services (they cannot use NIS+).

Which service a machine uses for various name services is controlled by the machine’s `nsswitch.conf` file. This file is called the *switch* file. See Chapter 2 for further information.

NIS and FNS

Under certain conditions, FNS commands can be used by NIS clients to update naming information that pertains to them such as file systems and printers. (See “NIS Clients Can Update Contexts With FNS if SKI is Running” on page 368 for details.)

NIS Machine Types

There are three types of NIS machines:

- Master server
- Slave servers
- Clients of NIS servers

Any machine can be an NIS client, but only machines with disks should be NIS servers, either master or slave. Servers are also clients, typically of themselves.

NIS Servers

By definition, an NIS server is a machine storing a set of maps that are available to network machines and applications. The NIS server does not have to be the same machine as the NFS file server.

NIS servers come in two varieties, master and slave. The machine designated as master server contains the set of maps that you, the NIS administrator, create and update as necessary. Each NIS domain must have one, and only one, master server. This should be a machine that can propagate NIS updates with the least performance degradation.

You can designate additional NIS servers in the domain as slave servers. A slave server has a complete copy of the master set of NIS maps. Whenever the master server maps are updated, the updates are propagated among the slave servers. The existence of slave servers allows the system administrator to evenly distribute the load resulting from answering NIS requests. It also minimizes the impact of a server becoming unavailable.

Normal practice is to designate one master server for all NIS maps. However, because each individual NIS map has the machine name of the master server encoded within it, you could designate different servers to act as master and slave servers for different maps. Note, however, that randomly designating a server as master of one map and another server as master of another map can cause a great deal of administrative confusion. For that reason it is best to have a single server be the master for all the maps you create within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

NIS Clients

NIS clients run processes that request data from maps on the servers. Clients do not make a distinction between master and slave servers, since all NIS servers should have the same information.

NIS servers are also clients, typically though not necessarily, of themselves. For information on how to create NIS clients, refer to the `ybind` man page.

NIS Elements

The NIS service is composed of the following elements:

- Domains (see “The NIS Domain ” on page 285)
- Maps (see “NIS Maps” on page 287)
- Daemons (see “NIS Daemons ” on page 285)
- Utilities (see “NIS Utilities” on page 286)
- NIS Command Set (see “Summary of NIS-Related Commands” on page 291)

The NIS Domain

An NIS *domain* is a collection of machines that share a common set of NIS maps. Each domain has a domain name and each machine sharing the common set of maps belongs to that domain. Domain names are case-sensitive.

Any machine can belong to a given domain, as long as there is a server for that domain’s maps in the same network. Solaris Release 2 machines do not require the server to be on the same subnet, but earlier implementations of NIS historically have required that a server exist on every subnet using NIS. A NIS client machine obtains its domain name and binds to a NIS server as part of its boot process.

NIS Daemons

NIS service is provided by five daemons as shown in Table 17-1.

TABLE 17-1 NIS Daemons

Daemon	Function
<code>ypserv</code>	Server process
<code>ypbind</code>	Binding process
<code>ypxfr</code>	High speed map transfer
<code>rpc.yppasswdd</code>	NIS password update daemon
<code>rpc.yupdated</code>	Modifies other maps such as <code>publickey</code>

NIS Utilities

NIS service is supported by nine utilities as shown in Table 17-2.

TABLE 17-2 NIS Utilities

Utility	Function
<code>makedbm</code>	Creates <code>dbm</code> file for an NIS map
<code>ypcat</code>	Lists data in a map
<code>ypinit</code>	Builds and installs an NIS database and initializes NIS client's <code>ypservers</code> list.
<code>yppmatch</code>	Finds a specific entry in a map
<code>yppoll</code>	Gets a map order number from a server
<code>yppush</code>	Propagates data from NIS master to NIS slave server
<code>ypset</code>	Sets binding to a particular server

TABLE 17-2 NIS Utilities (continued)

Utility	Function
<code>ypwhich</code>	Lists name of the NIS server and nickname translation table
<code>ypxfr</code>	Transfers data from master to slave NIS server

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. On a network running NIS, the NIS master server for each NIS domain maintains a set of NIS maps for other machines in the domain to query. NIS slave servers also maintain duplicates of the master server's maps. NIS client machines can obtain name space information from either master or slave servers.

NIS maps are one type of implementation of Solaris databases. (Other types, not necessarily overlapping, are the files generally found in the `/etc` directory, the DNS resource records (RRs), and NIS+ tables.)

NIS Maps Overview

NIS maps are essentially two-column tables. One column is the *key* and the other column is information value related to the key. NIS finds information for a client by searching through the keys. Some information is stored in several maps because each map uses a different key. For example, the names and addresses of machines are stored in two maps: `hosts.byname` and `hosts.byaddr`. When a server has a machine'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.

Maps for a domain are located in each server's `/var/yp/domainname` directory. For example, the maps that belong to the domain `test.com` are located in each server's `/var/yp/test.com` directory.

An NIS `Makefile` is stored in the `/var/yp` directory of machines designated as a NIS server at installation time. Running `make` in that directory causes `makedbm` to create or modify the default NIS maps from the input files. (See *Solaris Naming Setup and Configuration Guide* for a description of using this process to initially set up your NIS name space.)

Note - Never make the maps on a slave server. Always run `make` on the master server.

The information in NIS maps is stored in `ndbm` format. The `yfiles` and `ndbm man` pages explain the format of the map file.

Default NIS Maps

A default set of NIS maps are provided for you. You may want to use all these maps or only some of them. NIS can also use whatever maps you create or add when you install other software products.

Table 17-3 describes the default NIS maps, information they contain, and whether the operating system consults the corresponding administrative files when NIS is running.

TABLE 17-3 NIS Map Descriptions

Map Name	Corresponding NIS Admin File	Description
<code>bootparams</code>	<code>bootparams</code>	Contains path names of files clients need during boot: <code>root</code> , <code>swap</code> , possibly others.
<code>ethers.byaddr</code>	<code>ethers</code>	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
<code>ethers.byname</code>	<code>ethers</code>	Same as <code>ethers.byaddr</code> , except the key is machine name instead of the Ethernet address.
<code>group.bygid</code>	<code>group</code>	Contains group security information with group ID as key.
<code>group.byname</code>	<code>group</code>	Contains group security information with group name as key.
<code>hosts.byaddr</code>	<code>hosts</code>	Contains machine name, and IP address, with IP address as key.
<code>hosts.byname</code>	<code>hosts</code>	Contains machine name and IP address, with machine (host) name as key.
<code>mail.aliases</code>	<code>aliases</code>	Contains aliases and mail addresses, with aliases as key.

TABLE 17-3 NIS Map Descriptions *(continued)*

Map Name	Corresponding NIS Admin File	Description
mail.byaddr	aliases	Contains mail address and alias, with mail address as key.
netgroup.byhost	netgroup	Contains group name, user name and machine name.
netgroup.byuser	netgroup	Same as netgroup.byhost, except that key is user name.
netgroup	netgroup	Same as netgroup.byhost, except that key is group name.
netid.byname	passwd, hosts group	Used for UNIX-style authentication. Contains machine name and mail address (including domain name). If there is a netid file available it is consulted in addition to the data available through the other files.
netmasks.byaddr	netmasks	Contains network mask to be used with IP submitting, with the address as the key.
networks.byaddr	networks	Contains names of networks known to your system and their IP addresses, with the address as the key.
networks.byname	networks	Same as networks.byaddr, except key is name of network.
passwd.adjunct.byname	passwd and shadow	Contains auditing information and the hidden password information for C2 clients.
passwd.byname	passwd and shadow	Contains password information with user name as key.
passwd.byuid	passwd and shadow	Same as passwd.byname, except that key is user ID.
protocols.byname	protocols	Contains network protocols known to your network.
protocols.bynumber	protocols	Same as protocols.byname, except that key is protocol number.

TABLE 17-3 NIS Map Descriptions (continued)

Map Name	Corresponding NIS Admin File	Description
<code>rpc.bynumber</code>	<code>rpc</code>	Contains program number and name of RPCs known to your system. Key is RPC program number.
<code>services.byname</code>	<code>services</code>	Lists Internet services known to your network. Key is port or protocol.
<code>services.byservice</code>	<code>services</code>	Lists Internet services known to your network. Key is service name.
<code>ypservers</code>	N/A	Lists NIS servers known to your network.

Using NIS Maps

NIS makes updating network databases much simpler than with the `/etc` files system. You no longer have to change the administrative `/etc` files on every machine each time you modify the network environment.

For example, when you add a new machine to a network running NIS, you only have to update the input file in the master server and run `make`. This automatically updates the `hosts.byname` and `hosts.byaddr` maps. These maps are then transferred to any slave servers and are made available to all of the domain's client machines and their programs. When a client machine or application requests a machine name or address, the NIS server refers to the `hosts.byname` or `hosts.byaddr` map as appropriate and sends the requested information to the client.

You can use the `ypcat` command to display the values in a map. The `ypcat` basic format is:

```
% ypcat mapname
```

Where *mapname* is the name of the map you want to examine or its *nickname*. If a map is composed only of keys, as in the case of `ypservers`, use `ypcat -k`; otherwise, `ypcat` prints blank lines. The `ypcat` man page describes more options for `ypcat`.

You can use the `ypwhich` command to determine which server is the master of a particular map. Type the following:

```
% ypwhich -m mapname
```

Where *mapname* is the name or the nickname of the map whose master you want to find. `ypwhich` responds by displaying the name of the master server. For complete information, refer to the `ypwhich` man page.

NIS Map Nicknames

Nicknames are aliases for full map names. To obtain a list of available map nicknames, such as `passwd` for `passwd.byname`, type `ypcat -x` or `ypwhich -x`.

Nicknames are stored in the `/var/yp/nicknames` file, which contains a map nickname followed by the fully specified name for the map, separated by a space. This list may be added to or modified. Currently, there is a limit of 500 nicknames.

Summary of NIS-Related Commands

The NIS service includes specialized daemons, system programs, and commands, which are summarized in Table 17-4. Refer to their man pages for details about how to use them.

TABLE 17-4 NIS Command Summary

Command	Description
<code>ypserv</code>	Serves NIS clients' requests for information from a NIS map. <code>ypserv</code> is a daemon that runs on NIS servers with a complete set of maps. At least one <code>ypserv</code> daemon must be present on the network for NIS service to function.
<code>ypbind</code>	Provides NIS server binding information to clients. It provides binding by finding a <code>ypserv</code> process that serves maps within the domain of the requesting client. <code>ypbind</code> must run on all servers and clients.
<code>ypinit</code>	Automatically creates maps for an NIS server from the input files. It is also used to construct the initial <code>/var/yp/binding/domain/ypservers</code> file on the clients. Use <code>ypinit</code> to set up the master NIS server and the slave NIS servers for the first time.
<code>make</code>	Updates NIS maps by reading the <code>Makefile</code> (when run in the <code>/var/yp</code> directory). You can use <code>make</code> to update all maps based on the input files or to update individual maps. The <code>ypmake(1M)</code> man page describes the functionality of <code>make</code> for NIS.
<code>makedbm</code>	<code>makedbm</code> takes an input file and converts it into <code>dbm.dir</code> and <code>dbm.pag</code> files—valid <code>dbm</code> files that NIS can use as maps. You can also use <code>makedbm -u</code> to disassemble a map, so that you can see the key-value pairs that comprise it.

TABLE 17-4 NIS Command Summary (continued)

Command	Description
<code>ypxfr</code>	Pulls an NIS map from a remote server to the local <code>/var/yp/domain</code> directory, using NIS itself as the transport medium. You can run <code>ypxfr</code> interactively, or periodically from a <code>crontab</code> file. It is also called by <code>ypserv</code> to initiate a transfer.
<code>ypxfrd</code>	Provides map transfers service for <code>ypxfr</code> requests (generally slave servers). It is run only on the master server.
<code>yppush</code>	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
<code>ypset</code>	Tells a <code>ypbind</code> process to bind to a named NIS server. This is not for casual use and its use is discouraged because of security implications. See the <code>ypset(1M)</code> and <code>ypbind(1M)</code> man pages for information about the <code>ypset</code> and <code>ypsetme</code> options to the <code>ypbind</code> process.
<code>yppoll</code>	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.
<code>ypcat</code>	Displays the contents of an NIS map.
<code>ypmatch</code>	Prints the value for one or more specified keys in an NIS map. You cannot specify which version of the NIS server map you are seeing.
<code>ypwhich</code>	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the <code>-m mapname</code> option, which NIS server is master of each of the maps. If only <code>-m</code> is used, it displays the names of all the maps available and their respective master servers.

NIS Binding

NIS clients get information from a NIS server through the binding process, which can work in one of two modes: server-list or broadcast.

- **Server-list.** In the server-list mode, the `ypbind` process queries the `/var/yp/binding/domain/ypservers` list for the names of all of the NIS servers in the domain. The `ypbind` process binds only to servers in this file. The file is created by running `ypinit -c`.
- **Broadcast.** The `ypbind` process can also use an RPC broadcast to initiate a binding. Since broadcasts are only local subnet events that are not routed further,

there must be at least one server (master or slave) on the same subnet as the client. The servers themselves may exist throughout different subnets since map propagation works across subnet boundaries. In a subnet environment, one common method is to make the subnet router an NIS server. This allows the domain server to serve clients on either subnet interface.

Server-List Mode

The binding process in server-list mode works as follows:

1. Any program, running on the NIS client machine that needs information provided by an NIS map, asks `ypbind` for the name of a server.
2. `ypbind` looks in the `/var/yp/binding/domainname/ypservers` file for a list of NIS servers for the domain.
3. `ypbind` initiates binding to the first server in the list. If the server does not respond, `ypbind` tries the second, and so on, until it finds a server or exhausts the list.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Broadcast Mode

The broadcast mode binding process works as follows:

1. `ypbind` must be started with the broadcast option set (`broadcast`).
2. `ypbind` issues an RPC broadcast in search of an NIS server.

Note - In order to support such clients, it is necessary to have an NIS server on each subnet requiring NIS service.

1. `ypbind` initiates binding to the first server that responds to the broadcast.
2. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
3. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
4. `ypserv` sends the requested information back to the client.

Normally, once a client is bound to a server it stays bound to that server until something causes it to change. For example, if a server goes out of service, the clients it served will then bind to new servers.

To find out which NIS server is currently providing service to a specific client, use the following command:

```
% ypwhich machinename
```

Where *machinename* is the name of the client. If no machine name is mentioned, `ypwhich` defaults to the local machine (that is, the machine on which the command is run).

Differences Between This and Earlier NIS Versions

The following features are new or different in Solaris Release 2.6 NIS.

NSKit Discontinued

The most recent Solaris releases have not included NIS service. Up to now, NIS service had to be installed from the unbundled NSKit. NIS has now been included in the Solaris Release 2.6 and there is no 2.6 Release NSKit.

Because NIS service is now part of the Solaris 2.6 Release, the `SUNWnsktu` and `SUNWnsktr` packages no longer exist. Instead, NIS is now installed via the NIS Server cluster (containing the `SUNWypu` and `SUNWyptr` packages).

NIS service is no longer started with the `/etc/init.d/yp` script which no longer exists. With the Solaris 2.6 Release, you first configure your master server NIS maps with the `ypinit` script, and then start NIS with `ypstart`. NIS service is stopped with the `ypstop` command.

The `ypupdated` Daemon

The most recent versions of NSKit did not include the `ypupdated` daemon. The `ypupdated` daemon is now included in this Solaris release.

`/var/yp/securenets`

As with the previous NSKit release, the `/var/yp/securenets` file is now used to limit access to NIS services. If such a file exists on an NIS server, the server only

answers queries or supplies maps to machines and networks listed in the file. For the file format, see the `securenets` man page.

The following is an example of a `securenets` file.

```
255.255.255.0 13.13.13.255
host 13.13.14.1
host 13.13.14.2
```

Multihomed Machine Support

As with the previous NSKit release, the `ypserv` process provides support for machines which have more than one network address. When the machine maps are created, the `Makefile` creates a `YP_MULTI_HOSTNAME` entry in the map for any machine that has more than one address. This entry lists all the addresses for that machine. When the machine address is needed, an attempt is made to use the closest address on the list. See the `ypserv` man page for more details.

The determination of closest address is an arithmetic one and as such there is no check for address validity. For example, suppose that a multihomed machine has six IP addresses and only five of the interfaces on the machine are operating normally. Machines on a network that is not directly connected to this multihomed machine can receive the IP address for the down interface from `ypserv`. Thus, this hypothetical client can not reach the multihomed machine.

Note - All addresses for a multihomed machine should normally be active. If a particular address or machine is going to be out of service, remove it from the NIS maps.

Sun Operating System 4.X Compatibility Mode

Solaris 2.6 release NIS supports password configuration files in both the Sun Operating System 4.x (Solaris release 1) password format and the Solaris Release 2 password and shadow file formats.

The mode of operation is determined by the existence of the file `$PWDIR/shadow`, where `$PWDIR` is the `Makefile` macro set in the `/var/yp/Makefile` file. If the `shadow` file exists, NIS operates in the Solaris Release 2 mode. If this file does not exist, NIS operates in the SunOS 4.x mode.

In the SunOS 4.x mode, all password information is kept in the `passwd` file. In the Solaris Release 2 mode, password information is kept in the `shadow` file and the user account information is kept in the `passwd` file.

If the `make` macro `PWDIR` is set to the `/etc` directory, NIS can operate only in the Solaris Release 2 mode because of the Solaris Release 2 `passwd` processing

requirements. However, if `PWDIR` points to any directory other than `/etc`, the user has the option of keeping `passwd` configuration files in either the SunOS 4.x format or in the Solaris Release 2 format. The `rpc.yppasswdd` daemon understands both password formats. The Solaris Release 2 format is recommended.

Using the Name Service Switch

The name service switch is designed to simplify name service administration. Client machines and applications use this switch to select a name service. The switch mechanism is implemented using the `/etc/nsswitch.conf` file, which specifies the source(s) used to resolve references for each information type.

This section discusses only those elements that are needed to properly configure the name service switch for NIS operation. For a more detailed description of the `nsswitch.conf` file, see Chapter 2.

An `nsswitch.conf` file is automatically loaded into every machine's `/etc` directory by the Solaris 2.6 release software, along with three alternate (template) versions:

- `/etc/nsswitch.nisplus`
- `/etc/nsswitch.nis`
- `/etc/nsswitch.files`

These alternate template files contain the default switch configurations used by the NIS+ service, NIS, and local files. (See “The `nsswitch.conf` Template Files” on page 17.) No default file is provided for DNS, but you can edit any of these files to use DNS (see “DNS Forwarding for NIS Clients” on page 20).

Note that this switch functionality does not exist under SunOS 4.x. Thus, DNS forwarding for 4.x clients must be done on the NIS server. In this situation, if a 4.x client requests information for a host that is not listed in the NIS server's NIS map, the NIS server forwards the client's host request to a DNS server on the client's behalf.

When Solaris 2.6 release software is first installed on a machine, the installer selects the machine's default name service: NIS+, NIS, or local files. During installation, the corresponding template file is copied to `/etc/nsswitch.conf`. For a machine client using NIS, the installation process copies `nsswitch.nis` to `nsswitch.conf`. Unless you have an unusual NIS database setup, the default `/etc/nsswitch.nis` template file as copied to `nsswitch.conf` should be sufficient for NIS operation.

When changing a machine client from naming system (`/etc`, NIS or NIS+) to another, you copy the corresponding template file to `nsswitch.conf`. You can also change the sources of particular types of network information used by the client by editing the appropriate lines of the `/etc/nsswitch.conf` file. See *Solaris Naming Setup and Configuration Guide*, and Chapter 2.



Caution - If the `/etc/nsswitch.conf` file is set to `files` and not `nis` for host information, and the server is not included in the `/etc/hosts` file, then the `ypcat` command generates the following error
message: `RPC failure: ``RPC failure on yp operation```

Administering NIS

This chapter describes how to administer NIS.

- “Password Files and Namespace Security ” on page 300
- “Administering NIS Users ” on page 300
- “Netgroups ” on page 303
- “Working With NIS Maps” on page 304
- “Obtaining Map Information” on page 304
- “Changing a Map’s Master Server” on page 305
- “Modifying Configuration Files” on page 307
- “Modifying and Using the `Makefile` ” on page 307
- “Updating Existing Maps ” on page 310
- “Adding a New Slave Server ” on page 316
- “Using NIS with C2 Security ” on page 318
- “Changing a Machine’s NIS Domain ” on page 318
- “Using NIS in Conjunction With DNS ” on page 318
- “Turning Off NIS Services” on page 320
- “NIS Problem Solving and Error Messages” on page 321

See Chapter 17, for a general description of NIS.

See *Solaris Naming Setup and Configuration Guide* for information on how to initially set up and configure NIS.

Password Files and Namespace Security

For security reasons:

- It is best to limit access to the NIS maps on the master server.
- The files used to build the NIS password maps should not contain an entry for `root` to protect against unauthorized access. To accomplish this, the password files used to build the password maps should have the `root` entry removed from them and be located in a directory other than the master server's `/etc` directory. This directory should be secured against unauthorized access.

For example, the master server password input files could be stored in a directory such as `/var/yp`, or any directory of your choice, as long as the file itself is not a link to another file and is specified in the Makefile. The `/usr/lib/netsvc/yp/ypstart` script automatically sets the correct directory option according to the configuration specified in your Makefile.

Note - In addition to the older Solaris 1.x version `passwd` file format, this implementation of NIS accepts the Solaris Release 2 `passwd` and `shadow` file formats as input for building the NIS password maps.

Administering NIS Users

This section includes information about setting user passwords, adding new users to an NIS domain, and assigning users to netgroups.

Adding a New User to an NIS Domain

To add a new NIS user:

1. **Log in as root on the master NIS server.**
2. **Create the new user's login ID with the `useradd` command.**

For Solaris Release 2 systems, type the following:

```
# useradd userID
```

Where *userID* is the login ID of the new user. This command creates entries in the `/etc/passwd` and `/etc/shadow` files on the master NIS server.

3. Create the new user's initial password.

To create an initial password that the new user can use to log in, run the `passwd` command in the form:

```
# passwd userID
```

Where *userID* is the login ID of the new user. You will be prompted for the password to assign to this user.

This step is necessary because the password entry created by the `useradd` command is locked, which means that the new user cannot log in. By specifying an initial password, you unlock the entry.

4. If necessary, copy the new entry into the server's `passwd` map input files.

If the map source files on your master server are in a directory other than `/etc` (as they should be), you have to copy and paste the new lines from the `/etc/passwd` and `/etc/shadow` files into the `passwd` map input files on the server. (See "Password Files and Namespace Security" on page 300 for additional information on this matter.)

For example, if you added the new user `baruch`, the line from `/etc/passwd` that you would copy to your `passwd` input file would look like:

```
baruch:x:123:10:User baruch:/home/baruch:/bin/csh:
```

The line for `baruch` that you would copy from `/etc/shadow` would look like:

```
baruch:Wl2345GkHic:6445::::::
```

Note - If you are using a Solaris Release 1 `passwd` file format as input for your NIS maps, you must use a text editor to add the new user to your `passwd` file, manually.

5. Make sure that the Makefile correctly specifies the directory where the password input file resides.

6. If appropriate, delete the new user's entries from `/etc/passwd` and `/etc/shadow` input files.

For security reasons, it is not good practice to maintain user entries in the NIS master server `/etc/passwd` and `/etc/shadow` files. After copying the entries for the new user to the NIS map source files that are stored in some other directory, use the `userdel` command on the master server to delete the new user.

For example, to delete the new user `baruch` from the master server's `/etc` files, you would enter:

```
# userdel baruch
```

For more information about `userdel`, see the `userdel` man page.

7. Update the NIS passwd maps.

After you have updated the `passwd` input file on the master server, update the `passwd` maps by running `make` in the directory containing the source file.

```
# userdel baruch
# cd /var/yp
# /usr/ccs/bin/make passwd
```

8. Tell the new user the initial password you have assigned to his or her login ID.

After logging in, the new user can run `passwd` at any time to establish a different password.

User Passwords

Users run `passwd` to change their passwords.

```
% passwd username
```

(See “Using Passwords” on page 144 for a complete description of password matters from the users point of view.)

Before users can change their passwords, you must start the `rpc.yppasswdd` daemon on the master server to update the password file. The commands for starting the daemon are already present in the `/usr/lib/netsvc/yp/ypstart` file.

The `rpc.yppasswdd` daemon is started automatically by `ypstart` on the master server. Notice that when the `-m` option is given to `rpc.yppasswd`, a `make` is forced in `/var/yp` immediately following a modification of the file. If you want to avoid having this `make` take place each time the `passwd` file is changed, remove the `-m` option from the `rpc.yppasswd` command in the `ypstart` script and control the pushing of the `passwd` maps through the `crontab` file.

Note - No arguments should follow the `rpc.yppasswd -m` command. Although you can edit the `ypstart` script file to achieve a different action, it is not recommended that you modify this file other than optionally removing the `-m` option. All commands and daemons invoked by this file with the proper set of command line parameters. If you choose to edit this file, be especially careful when editing the `rpc.yppasswdd` command. If you add an explicit call to the `passwd.adjunct` file, the exact `$PWDIR/security/passwd.adjunct` path must be used; otherwise, incorrect processing results.

Netgroups

NIS netgroups are groups (sets) of users or machines that you define for your administrative purposes. For example, you can create netgroups that:

- Define a set of users who can access a specific machine
- Define a set of NFS client machines to be given some specific filesystem access.
- Define a set of users who are to have administrator privileges on all the machines in a particular NIS domain.

Each netgroup is given a netgroup name. Netgroups do not directly set permissions or access rights. Instead, the netgroup names are used by other NIS maps in places where a user name or machine name would normally be used. For example, suppose you created a netgroup of network administrators called `netadmins`. To grant all members of the `netadmins` group access to a given machine, you need only add a `netadmin` entry to that machine's `/etc/passwd` file. Netgroup names can also be added to the NIS group map. See the netgroup man page for more detailed information on using netgroups.

On a network using NIS, the `netgroup` input file on the master NIS server is used for generating three maps: `netgroup`, `netgroup.byuser`, and `netgroup.byhost`. The `netgroup` map contains the basic information in the `netgroup` input file. The two other NIS maps contain information in a format that speeds lookups of netgroup information, given the machine or user.

Entries in the `netgroup` input file are in the format: *name ID*, where *name* is the name you give to a netgroup, and *ID* identifies a machine and/or user who belongs to the netgroup. You can specify as many ids (members) to a netgroup as you want, separated by commas. For example, to create a netgroup with three members, the `netgroup` input file entry would be in the format: *name ID, ID, ID*. The member IDs in a `netgroup` input file entry are in the format:

<code>([- <i>machine</i>], [- <i>user</i>], [<i>domain</i>])</code>

Where *machine* is a machine name, *user* is a user ID, and *domain* is the machine or user's NIS domain with each element separated by a comma. The domain element is optional and should only be used to identify machines or users in some other NIS

domain. The *machine* and *user* element of each member's entry are required, but a dash (-) is used to denote a null. There is no necessary relationship between the machine and user elements in an entry.

For example, below are two sample `netgroup` input file entries, each of which create a netgroup named `admins` composed of the users `hauri` and `juanita` who is in the remote domain `sales` and the machines `altair` and `sirius`.

```
admins (altair, hauri), (sirius,juanita,sales)
```

```
admins (altair,-), (sirius,-), (-,hauri), (-,juanita,sales)
```

Various programs use the `netgroup` NIS maps for permission checking during login, remote mount, remote login, and remote shell creation. These programs include: `mountd`, `login`, `rlogin`, and `rsh`. The `login` command consults the `netgroup` maps for user classifications if it encounters `netgroup` names in the `passwd` database. The `mountd` daemon consults the `netgroup` maps for machine classifications if it encounters `netgroup` names in the `/etc/dfs/dfstab` file. `rlogin` and `rsh` (in fact, any program that uses the `ruserok` interface) consults the `netgroup` maps for both machine and user classifications if they encounter `netgroup` names in the `/etc/hosts.equiv` or `.rhosts` files.

If you add a new NIS user or machine to your network, be sure to add them to appropriate `netgroups` in the `netgroup` input file. Then use the `make` and `yppush` commands to create the `netgroup` maps and push them to all of your NIS servers. See the `netgroup` man page for detailed information on using `netgroups` and `netgroup` input file syntax.

Working With NIS Maps

The following sections describe how to administer NIS maps.

Obtaining Map Information

Users can obtain information from and about the maps at any time by using the `ypcat`, `ypwhich`, and `ypmatch` commands. In the examples that follow, `mapname` refers both to the official name of a map and to its nickname, if any.

To list all the values in a map, type:

```
% ypcat mapname
```

To list both the keys and the values (if any) in a map, type:

```
% ypcat -k mapname
```

To list all the map nicknames, type any of the following commands:

```
% ypcat -x
```

```
% ypwhich -x
```

```
% ypmatch -x
```

To list all the available maps and their master(s), type:

```
% ypwhich -m
```

To list the master server for a particular map, type:

```
% ypwhich -m mapname
```

To match a key with an entry in a map, type:

```
% ypmatch key mapname
```

If the item you are looking for is not a key in a map, type:

```
% ypcat mapname | grep item
```

Where *item* is the information you are searching for. To obtain information about other domains, use the `-d domainname` options of these commands.

If the machine requesting information for a domain other than its default does not have a binding for the requested domain, it causes `ypbind` to consult the `/var/yp/binding/domainname/ypservers` file for a list of servers for that domain. If this file doesn't exist it issues an RPC broadcast for a server. In this case, there must be a server for the requested domain on the same subnet as the requesting machine.

Changing a Map's Master Server

To change the master server for a selected map, you first have to build the map on the new NIS master. Since the old master server name occurs as a key-value pair in the existing map (this pair is inserted automatically by `makedbm`), copying the map to the new master or transferring a copy to the new master with `ypxfr` is insufficient. You have to reassociate the key with the new master server name. If the map has an ASCII source file, you should copy this file to the new master.

Here are instructions for remaking a sample NIS map called `sites.byname`.

1. Log in to the new master as superuser and type:

```
newmaster# cd /var/yp
```

2. Makefile must have an entry for the new map before you specify the map to make. If this is not the case, edit the Makefile now.
3. To update or remake the map, type:

```
newmaster# make sites.byname
```

4. If the old master remains an NIS server, remote log in (rlogin) to the old master and edit Makefile. Comment out the section of the Makefile that made sites.byname so that it is no longer made there.
5. If sites.byname only exists as an ndbm file, remake it on the new master by disassembling a copy from any NIS server, then running the disassembled version through makedbm:

```
newmaster# cd /var/yp
newmaster# ypcat -k sites.byname | makedbm -domain/sites.byname
```

After making the map on the new master, you must send a copy of the new map to the other slave servers. However, do not use `yppush`, because the other slaves will try to get new copies from the old master, rather than the new one. A typical method for circumventing this is to transfer a copy of the map from the new master back to the old master. To do this, become superuser on the old master server and type:

```
oldmaster# /usr/lib/netsvc/yp/ypxfr -h newmaster sites.byname
```

Now it is safe to run `yppush`. The remaining slave servers still believe that the old master is the current master. They attempt to get the current version of the map from the old master. When they do so, they will get the new map, which names the new master as the current master.

If this method fails, you can try this cumbersome but sure-fire option: log in as root on each NIS server and execute the `ypxfr` command shown above.

Modifying Configuration Files

NIS intelligently parses the setup files. Although this makes NIS administration easier, it does make the behavior of NIS more sensitive to changes in the setup and configuration files.

Use the procedures in this section when modifying any of the following:

- `/var/yp/Makefile` to add or delete supported maps
- Add or delete `/etc/resolv.conf` to allow or deny DNS forwarding
- Add or delete `$PWDIR/security/passwd.adjunct` to allow or deny C2 security. (`$PWDIR` is defined in `/var/yp/Makefile`.)

To modify any of the listed files:

1. Stop the NIS server by typing:

```
# /etc/init.d/yp stop
```

2. Make the necessary changes to your files.

3. Restart the NIS server by typing:

```
# /etc/init.d/yp start
```

You do not have to stop and start NIS when changing NIS maps or the map source files.

Keep in mind the following points:

- Deleting a map or source file from a NIS master server does not automatically result in corresponding deletions from slave servers. You must delete maps and source files from slave servers by hand.
- New maps do not automatically get pushed to existing slave servers. You must run `ypxfr` from the slaves.

Modifying and Using the Makefile

You can modify the `Makefile` provided by default in `/var/yp` to suit your needs. (Be sure to keep an unmodified copy of the original `Makefile` for future reference.)

You can add or delete maps, and you can change the names of some of the directories.

To add a new NIS map, you must get copies of the `ndbm` files for the map into the `/var/yp/domainname` directory on each of the NIS servers in the domain. This is normally done for you by the `Makefile`. After deciding which NIS server is the

master of the map, modify the `Makefile` on the master server so that you can conveniently rebuild the map. Different servers can be masters of different maps, but in most cases this leads to administrative confusion, and it is strongly recommended that you set only one server as the master of all maps.

Typically a human-readable text file is filtered through `awk`, `sed`, or `grep` to make it suitable for input to `makedbm`. Refer to the default `Makefile` for examples. See the `make` man page for general information about the `make` command.

Use the mechanisms already in place in the `Makefile` when deciding how to create dependencies that `make` will recognize. Be aware that `make` is very sensitive to the presence or absence of tabs at the beginning of lines within the dependency rules, and a missing tab can invalidate an entry that is otherwise well formed.

Adding Makefile Entries

To add an entry to the `Makefile`, do the following:

- Add the name of the database to the `all` rule
- Write the `time` rule
- Add the rule for the database

For example, in order for the `Makefile` to work on automounter input files, you would have to add the `auto_direct.time` and `auto_home.time` maps to the NIS database.

To add these maps to the NIS database:

1. **Modify the line that starts with the word `all` by adding the name(s) of the database you want to add:**

```
all: passwd group hosts ethers networks rpc services protocols \  
netgroup bootparams aliases netid netmasks \  
auto_direct auto_home auto_direct.time auto_home.time
```

The order of the entries is not relevant, but the blank space at the beginning of the continuation lines must be a Tab, not spaces.

2. **Add the following lines at the end of the `Makefile`:**

```
auto_direct: auto_direct.time
auto_home: auto_home.time
```

3. Add an entry for `auto_direct.time` in the middle of the file.

```
auto_direct.time: $(DIR)/auto_direct
@(while read L; do echo $$L; done < $(DIR)/auto_direct
$(CHKPIPE) | \ (sed -e "/^#/d" -e "s/#.*$$/" -e "/^ *$$/d"
$(CHKPIPE) | \ $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto_direct;
@touch auto_direct.time;
@echo "updated auto_direct";
@if [ ! $(NOPUSH) ]; then $(YPPUSH) auto_direct; fi
@if [ ! $(NOPUSH) ]; then echo "pushed auto_direct"; fi
```

Where:

- `CHKPIPE` makes certain that the operations to the left of the pipe (`|`) are successfully completed before piping the results to next commands. If the operations to the left of the pipe do not successfully complete, the process is terminated with a “NIS make terminated” message.
- `NOPUSH` prevents the makefile from calling `yppush` to transfer the new map to the slave servers. If `NOPUSH` is not set, the push is done automatically.

The `while` loop at the beginning is designed to eliminate any backslash-extended lines in the input file. The `sed` script eliminates comment and empty lines, and feeds the output to

The same procedure should be followed for all other automounter maps such as `auto_home`, or any other nondefault maps.

4. Run `make`.

```
# make name
```

Where *name* is the name of the map you want to make. For example, `auto_direct`.

Deleting Makefile Entries

If you do not want the Makefile to produce maps for a specific database, edit the Makefile as follows:

1. Delete the name of the database from the `all` rule.
2. Delete or comment out the database rule for the database you want to delete.
For example, to delete the `hosts` database, the `hosts.time` entry should be removed.

3. Remove the time rule.

For example, to delete the `hosts` database, the `hosts: hosts.time` entry should be removed.

4. Remove the map from the master and slave servers.

Changing Makefile Macros/Variables

You can change the settings of the variables defined at the top of the `Makefile` simply by changing the value to the right of the equal sign (=). For instance, if you do not want to use the files located in `/etc` as input for the maps, but you would rather use files located in another directory, such as `/var/etc/domainname`, you should change the value of `DIR` from `DIR=/etc` to `DIR=/var/etc/domainname`. You may also change the value of `PWDIR` from `PWDIR=/etc` to `PWDIR=/var/etc/domainname`.

The variables are:

- **DIR=** The directory containing all of the NIS input files except `passwd` and `shadow`. The default value is `/etc`. Since it is not good practice to use the files in the master server's `/etc` directory as NIS input files, you should change this value.
- **PWDIR=** The directory containing the `passwd` and `shadow` NIS input files. Since it is not good practice to use the files in the master server's `/etc` directory as NIS input files, you should change this value.
- **DOM=** The NIS domain name. The default value of `DOM` is set using the `domainname` command. Remember that most NIS commands use the current machine's domain which is set in the machine's `/etc/defaultdomain` file.

Updating Existing Maps

After you have installed NIS, you may discover that some maps require frequent updating while others never need to change. For example, the `passwd.byname` map may change frequently on a large company's network. On the other hand, the `auto_master` map changes little, if at all.

When you need to update a map, you can use one of two updating procedures, depending on whether it is a default map or not.

- A default map is a map in the default set created by `ypinit` from the network databases.
- Nondefault maps may be any of the following:
 - Maps included with an application purchased from a vendor
 - Maps created specifically for your site

- Maps created from a nontext file

The following sections explain how to use various updating tools. In practice, you may decide to only use them if you add nondefault maps or change the set of NIS servers after the system is already up and running.

Modifying Default Maps

Use the following procedure for updating maps supplied with the default set.

1. Become root on the master server.

Always modify NIS maps only on the master server.

2. Edit the source file for the map you want to change, whether that file resides in `/etc` or in some other directory of your choice.

3. Type the following:

```
# cd /var/yp# make mapname
```

The `make` command then updates your map according to the changes you made in its corresponding file. It also propagates the changes among the other servers.

Modifying Nondefault Maps

To update a nondefault map, you must:

1. Create or edit its corresponding text file.
2. Build (or rebuild) the new or updated map. There are two ways to build a map:
 - Use the Makefile. Using the Makefile is the preferred method of building a non-default map. If the map has an entry in the `Makefile`, simply run `make name` where `name` is the name of map you want to build. If the map does not have a `Makefile` entry, try to create one following the instructions in “Modifying and Using the Makefile ” on page 307.
 - Use the `/usr/sbin/makedbm` program. (The `makedbm` man page fully describes this command.)

Using `makedbm` to Modify a Non-Default Map

There are two different methods for using `makedbm` to modify maps if you do not have an input file:

- Redirect the `makedbm -u` output to a temporary file, modify the file, then use the modified file as input to `makedbm`.

- Have the output of `makedbm -u` operated on within a pipeline that feeds into `makedbm`. This is appropriate if you can update the disassembled map with either `awk`, `sed`, or a `cat` append.

Creating New Maps

To create new maps, you can use one of two possible procedures: the first method uses an existing text file as input; the second uses standard input.

Creating Maps From Text Files

Assume that a text file `/var/yp/mymap.asc` was created with an editor or a shell script on the master. You want to create an NIS map from this file and locate it in the *homedomain* subdirectory. To do this, type the following on the master server:

```
# cd /var/yp
# makedbm mymap.asc homedomain/mymap
```

The *mymap* map now exists on the master server in the directory *homedomain*. To distribute the new map to slave servers run `ypxfr`.

Adding Entries to a File-Based Map

Adding entries to *mymap* is simple. First, you must modify the text file `/var/yp/mymap.asc`. (If you modify the actual `dbm` files without modifying the corresponding text file, the modifications are lost.) Then run `makedbm` as shown above.

Creating Maps From Standard Input

When no original text file exists, create the NIS map from the keyboard by typing input to `makedbm`, as shown below (end with Control-D):

```
ypmaster# cd /var/yp
ypmaster# makedbm - homedomain/mymapkey1 value1 key2 value2 key3 value3
ypmaster#
```

Modifying Maps Made From Standard Input

If you later need to modify the map, you can use `makedbm` to disassemble the map and create a temporary text intermediate file. To disassemble the map and create a temporary file, type the following:

```
% cd /var/yp
% makedbm -u homedomain/mymap > mymap.temp
```

The resulting temporary file `mymap.temp` has one entry per line. You can edit this file as needed, using any text editor.

To update the map, give the name of the modified temporary file to `makedbm` by typing the following:

```
% makedbm mymap.temp homedomain/mymap
% rm mymap.temp
```

Then propagate the map to the slave servers, by becoming root and typing:

```
# yppush mymap
```

The preceding paragraphs explained how to use `makedbm` to create maps; however, almost everything you actually have to do can be done by `ypinit` and `Makefile` unless you add nondefault maps to the database or change the set of NIS servers after the system is already up and running.

Whether you use the `Makefile` in `/var/yp` or some other procedure the goal is the same: a new pair of well-formed `dbm` files must end up in the `maps` directory on the master server.

Propagating an NIS Map

After a map is changed, the `Makefile` uses `yppush` to propagate a new map to the slave servers (unless `NOPUSH` is set in the `Makefile`). It does this by informing the `ypserv` daemon and sending a map transfer request. The `ypserv` daemon on the slave then starts a `ypxfr` process, which in turn contacts the `ypxfrd` daemon on the master server. Some basic checks are made (for example did the map really change?) and then the map is transferred. `ypxfr` on the slave then sends a response to the `yppush` process indicating whether the transfer succeeded.

Note - The above procedure will *not* work for newly created maps that do not yet exist on the slave servers. New maps must be sent to the slave servers by running `ypxfr` on the slaves.

Occasionally, maps fail to propagate and you must to use `ypxfr` manually to send new map information. You may choose to use `ypxfr` in two different ways: periodically through the root `crontab` file, or interactively on the command line. These approaches are discussed in the following sections.

Using `cron` For Map Transfers

Maps have different rates of change. For instance, some may not change for months at a time, such as `protocols.byname` among the default maps and `auto_master` among the nondefault maps; but `passwd.byname` may change several times a day. Scheduling map transfer using the `crontab` command allows you to set specific propagation times for individual maps.

To periodically run `ypxfr` at a rate appropriate for the map, the root `crontab` file on each slave server should contain the appropriate `ypxfr` entries. `ypxfr` contacts the master server and transfers the map only if the copy on the master server is more recent than the local copy.

Note - If your master server runs `rpc.yppasswdd` with the default `-m` option, then each time someone changes their `yp` password, the `passwd` daemon runs `make`, which rebuilds the `passwd` maps.

Using Shell Scripts with `cron` and `ypxfr`

As an alternative to creating separate `crontab` entries for each map, you may prefer to have the root `crontab` command run a shell script that periodically updates all maps. There are sample map-updating shell scripts in the `/usr/lib/netsvc/yp` directory. The script names are `ypxfr_1perday`, `ypxfr_1perhour`, and `ypxfr_2perday`. You can easily modify or replace these shell scripts to fit your site requirements. Code Example 18-1 shows the default `ypxfr_1perday` shell script.

CODE EXAMPLE 18-1 `ypxfr_1perday` Shell Script

```
#!/bin/sh
#
# ypxfr_1perday.sh - Do daily yp map check/updates
PATH=/bin:/usr/bin:/usr/lib/netsvc/yp:$PATH
export PATH
# set -xv
ypxfr group.byname
ypxfr group.bygid
ypxfr protocols.byname
ypxfr protocols.bynumber
ypxfr networks.byname
```

(continued)

```
ypxfr networks.byaddr
ypxfr services.byname
ypxfr ypservers
```

This shell script updates the maps once per day, if the root `crontab` is executed daily. You can also have scripts that update maps once a week, once a month, once every hour, and so forth, but be aware of the performance degradation implied in frequently propagating the maps.

Run the same shell scripts as root on each slave server configured for the NIS domain. Alter the exact time of execution from one server to another to avoid bogging down the master.

If you want to transfer the map from a particular slave server, use the `-h machine` option of `ypxfr` within the shell script. Here is the syntax of the commands you put in the script:

```
/usr/lib/netsvc/yp/ypxfr -h machine [ -c ] mapname
```

Where *machine* is the name of the server with the maps you want to transfer, and *mapname* is the name of the requested map. If you use the `-h` option without specifying a machine, `ypxfr` tries to get the map from the master server. If `ypserv` is not running locally at the time `ypxfr` is executed, you must use the `-c` flag so that `ypxfr` does not send a clear current map request to the local `ypserver`.

You can use the `-s domain` option to transfer maps from another domain to your local domain. These maps should be the same across domains. For example, two domains might share the same `services.byname` and `services.byaddr` maps. Alternatively, you can use `rcp`, or `rdist` for more control, to transfer files across domains.

Directly Invoking `ypxfr`

The second method of invoking `ypxfr` is to run it as a command. Typically, you do this only in exceptional situations—for example, when setting up a temporary NIS server to create a test environment or when trying to quickly get an NIS server that has been out of service consistent with the other servers.

Logging `ypxfr` Activity

The transfer attempts and results of `ypxfr` can be captured in a log file. If a file called `/var/yp/ypxfr.log` exists, results are appended to it. No attempt to limit

the size of the log file is made. To prevent it from growing indefinitely, empty it from time to time by typing:

```
# cd /var/yp
# cp ypxfr.log ypxfr.log.old
# cat /dev/null > /var/yp/ypxfr.log
```

You can have `crontab` execute these commands once a week. To turn off logging, remove the log file.

Adding a New Slave Server

After NIS is running, you may need to create a new NIS slave server that you did not include in the initial list given to `ypinit`.

To add a new NIS server:

1. **Log in to the master server as root.**
2. **Change to the NIS domain directory by typing:**

```
# cd /var/yp/domainname
```

3. **Disassemble the `ypservers` file, as follows:**

```
# makedbm -u ypservers >/tmp/temp_file
```

The `makedbm` command converts `ypservers` from `ndbm` format to a temporary ASCII file `/tmp/temp_file`.

4. **Edit the `/tmp/temp_file` file using a text editor. Add the name of the new slave server to the list of servers. Then save and close the file.**
5. **Run the `makedbm` command with `temp_file` as the input file and `ypservers` as the output file:**

```
# makedbm /tmp/temp_file ypservers
```

`makedbm` then converts `ypservers` back into `ndbm` format.

6. **Verify that the `ypservers` map is correct (since there is no ASCII file for `ypservers`) by typing:**

```
slave3# makedbm -u ypservers
```

The `makedbm` command displays each entry in `ypservers` on your screen.

Note - If a machine name is not in `ypservers`, it will not receive updates to the map files because `yppush` consults this map for the list of slave servers.

7. Set up the new slave server's NIS domain directory by copying the NIS map set from the master server.

To do this, log in to the new NIS slave as superuser and run the `ypinit` and `ypbind` commands:

```
slave3# cd /var/yp
slave3# ypinit -c list of servers
slave3# /usr/lib/netsvc/yp/ypbind
```

8. To initialize this machine as a slave, type the following:

```
# /usr/sbin/ypinit -s ypmaster
```

Where `ypmaster` is the machine name of the existing NIS master server.

9. Run `ypstop` to stop the machine running as a NIS client.

```
#/usr/lib/netsvc/up/ypstop
```

10. Run `ypstart` to start NIS slave service.

```
#/usr/lib/netsvc/up/ypstart
```

See the *Solaris Naming Setup and Configuration Guide* for a more detailed description of setting up NIS slave servers.

Using NIS with C2 Security

If the `$PWDIR/security/passwd.adjunct` file is present, C2 security is started automatically. (`$PWDIR` is defined in `/var/yp/Makefile`.) The C2 security mode uses the `passwd.adjunct` file to create the `passwd.adjunct` NIS map. In this implementation, NIS allows you to use both the `passwd.adjunct` file and `shadow` file to manage security. The `passwd.adjunct` file is only processed when you type:

```
# make passwd.adjunct
```

The `make passwd` command only processes the `passwd` map not the `passwd.adjunct` map when you run `make` manually in the C2 security mode.

Changing a Machine's NIS Domain

To change the NIS domain name of a machine:

1. **Edit the machine's `/etc/defaultdomain` file, exchanging its present contents with the new domain name for the machine.**

For example, if the current domain name is `sales.doc.com`, you might change it to `research.doc.com`.

2. **Run `domainname 'cat /etc/defaultdomain'`**
3. **Then set the machine up as a NIS client, slave, or master server.**
See *Solaris Naming Setup and Configuration Guide* for details.

Using NIS in Conjunction With DNS

Typically, NIS clients are configured with the `nsswitch.conf` file to use only NIS for machine name and address lookups. If this type of lookup fails, an NIS server may forward these lookups to DNS.

To configure machine name and address lookup to occur through NIS and then through DNS:

1. **The two maps `hosts.byname` and `hosts.byaddr` must have the `YP_INTERDOMAIN` key in them; to set this key, edit the `Makefile` and modify the lines (at the top of the file) from:**

```
#B=-b  
B=
```

to:

```
B=-b  
#B=
```

This tells `makedbm` to start with the `-b` flag when making the maps, and the `YP_INTERDOMAIN` key will be inserted into the `ndbm` files.

2. **Run `make` to rebuild that maps.**

```
# /usr/ccs/bin/make hosts
```

3. **Make sure that all NIS servers have an `/etc/resolv.conf` file that points to valid name server(s).**

4. **To enable DNS forwarding, stop each server with the `ypstop` command**

```
# /usr/lib/netsvcs/yp/ypstop
```

5. **Restart each server with the `ypstart` command:**

```
# /usr/lib/netsvcs/yp/ypstart
```

In this implementation of NIS, if a `/etc/resolve.conf` file exists on the server, `ypstart` automatically starts the `ypserv` daemon with the `-d` option to forward requests to DNS.

Note - If you have NIS servers that are not running the Solaris Release 2, then you must make sure that the `YP_INTERDOMAIN` key is present in the host maps for DNS to be consulted.

Problems in Mixed NIS Domains

Most of the preceding information assumes that both master and slave servers in the NIS domain are running the Solaris Release 2. If that is not the case, problems may arise. Table 18-1 summarizes how to successfully avoid problems in mixed NIS domains. The notation “4.0.3+” means “release 4.0.3 of the SunOS operating system or later.” The command `makedbm -b` is a reference to the “-B” variable in the `Makefile`.

TABLE 18-1 NIS/DNS in Heterogeneous NIS Domains

Slave	Master		
	4.0.3+	Solaris NIS	
4.0.3+	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr -b</code>	Master: <code>ypserv -d</code> Slave: <code>ypxfr -b</code>
Solaris NIS	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>makedbm -b</code> Slave: <code>ypxfr</code>	Master: <code>ypserv -d</code> Slave: <code>ypxfr</code> with <code>resolve.conf</code> or <code>ypxfr -b</code>

Turning Off NIS Services

If `ypserv` on the master is disabled, you can no longer update any of the NIS maps. If you choose to turn off NIS on a network currently running it, you can disable NIS after the next reboot by simply renaming the `ypbind` file to `ypbind.orig`, as follows:

```
% mv /usr/lib/netsvc/yp/ypbind /usr/lib/netsvc/yp/ypbind.orig
```

To disable NIS after the next reboot on a particular NIS slave or master, type the following on the server in question:

```
% mv /usr/lib/netsvc/yp/ypserv /usr/lib/netsvc/yp/ypserv.orig
```

To stop NIS immediately, type:

```
% /usr/lib/netsvc/yp/ypstop
```

The NIS service is automatically restarted after the next reboot unless the `ypbind` and `ypserv` files are renamed as described above.

NIS Problem Solving and Error Messages

- See “NIS Problems and Solutions” on page 537 and “NIS+ and NIS Compatibility Problems” on page 512 for problem solving information.
- Appendix B, for an alphabetic list of the more common namespace error messages and their meanings.

PART **V** Administering FNS

This part describes the Federated Naming Service (FNS) and how to administer it.

- Chapter 19
- Chapter 20
- Chapter 21
- Chapter 22
- Chapter 23
- Chapter 24
- Chapter 25
- Chapter 26

FNS Quickstart

This chapter provides a summary overview of FNS, a brief description of set up and configuration steps, and a programming example. Experienced administrators may find that this quick start chapter is all that they need.

- “Federated Naming Service (FNS)” on page 326
- “Composite Names and Contexts” on page 326
- “Enterprise Naming Services” on page 328
- “Global Naming Services” on page 329
- “FNS Naming Policies” on page 329
- “Organization Names” on page 330
- “Site Names” on page 331
- “User Names” on page 331
- “Host Names” on page 332
- “Service Names” on page 332
- “File Names” on page 332
- “Getting Started” on page 333
- “Browsing the FNS Namespace” on page 335
- “Updating the Namespace” on page 338
- “Federating a Global Namespace” on page 346
- “Namespace Browser Programming Examples” on page 348

For more detailed information, see the remaining chapters of this part. For more detailed initial FNS set up and configuration information, see *Solaris Naming Setup and Configuration Guide*.

Federated Naming Service (FNS)

Federated Naming Service (FNS) provides a method for federating multiple naming services under a single, simple interface for naming and directory operations. Naming services that can be linked with FNS include: NIS+, NIS, files, DNS, and X.500/LDAP.

X/Open Federated Naming (XFN)

The programming interface and policies that FNS supports are specified by XFN (X/Open Federated Naming).

Why FNS?

FNS is useful for the following reasons:

- A single uniform naming and directory interface is provided to clients for accessing naming and directory services. Consequently, the addition of new naming and directory services does not require changes to applications or existing services.
- Names can be composed in a uniform way. FNS defines a way to uniformly compose names from different naming systems so that applications can uniformly address objects in these different naming systems.
- Coherent naming is encouraged through the use of shared contexts and shared names. Different applications can use these shared names and contexts and need not duplicate the work.

Composite Names and Contexts

Fundamental to FNS are the notions of composite names and contexts.

Composite Names

A composite name is a name that spans multiple naming systems.

A composite name consists of an ordered list of components. Each component is a name from the namespace of a single naming system. Individual naming systems are

responsible for the syntax of each component. FNS defines the syntax for constructing a composite name using names from component naming systems. Composite names are composed left to right using the slash character (/) as the component separator.

For example, the composite name `.../doc.com/site/bldg-5.alameda` consists of four components: `...`, `doc.com`, `site`, and `bldg-5.alameda`.

Contexts

A context provides operations for:

- Associating (binding) names to objects
- Resolving names to objects
- Removing bindings
- Listing names
- Renaming
- Associating attributes with named objects
- Retrieving and updating attributes associated with named objects
- Searching for objects using attributes

A context contains a set of name-to-reference bindings. Each reference contains a list of communication end-points or addresses. The federated naming system is formed by contexts from one naming system being bound in the contexts of another naming system. Resolution of a composite name proceeds from contexts within one naming system to those in the next until the name is resolved.

Attributes

Attributes may be applied to named objects. Attributes are optional. A named object can have no attributes, one attribute, or multiple attributes.

Each attribute has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values.

XFN defines the base attribute interface for examining and modifying the values of attributes associated with existing named objects. These objects can be contexts or any other types of objects. Associated with a context are syntax attributes that describe how the context parses compound names.

The extended attribute interface contains operations that search for specific attributes and that create objects and their associated attributes.

Enterprise Naming Services

Enterprise-level naming services are used to name objects within an enterprise. FNS currently supports three enterprise-level naming services:

- NIS+ (see “NIS+” on page 328 below).
- NIS (see “NIS” on page 328).
- Files (see “Files-Based” on page 329).

NIS+

NIS+ is the preferred enterprise-wide information service in the Solaris 2.6 release environment. FNS organization units correspond to NIS+ domains and subdomains. There is one `orgunit` context for each domain and subdomain.

Under NIS+, FNS context and attribute data are stored in NIS+ tables. These tables are stored in NIS+ directory objects named `ctx_dir`. There is a `ctx_dir` directory object for each NIS+ domain and subdomain, residing at the same level as the domain's `groups_dir` and `org_dir` directory objects. Thus, the directory object `ctx_dir.sales.doc.com` contains FNS tables which store FNS context and attribute data for the `sales.doc.com` domain.

Under NIS+, you use FNS and NIS+ commands to work with the information in FNS tables. Do not edit these tables directly or manipulate them with UNIX commands.

NIS

NIS is an enterprise-wide information service in the Solaris environment. Each enterprise is a single NIS domain. There is one FNS organizational unit which corresponds to the single NIS domain.

Under NIS, FNS context and attribute data are stored in NIS maps. These maps are stored in a `/var/yp/domainname` directory on a NIS server. Under NIS, the super user can use FNS commands to work with the information in FNS maps.

NIS Clients Can Update Contexts With FNS if SKI is Running

If certain conditions are met, any NIS client (machine, process, or user) can use FNS commands such as `fncreate_fs` or `fncreate_printer` to update the client's own contexts. This allows NIS clients to use FNS commands to update applications such as Printer Administrator, CDE Calendar Manager, Admin Tool and others.

For non-super-users to update their own contexts with FNS commands, the following conditions must be met:

- Secure Key_management Infrastructure (SKI) must be available on the NIS master server.
- The `fnssydpd` daemon must be running on the NIS master server. This daemon must be started by someone with super user privileges.
- A client user or machine is only allowed to update its own context.
- The client must be authorized to perform the requested update.

Files-Based

Files refers to the naming files normally found in a machine's `/etc` directory. These machine-based files contain UNIX user and password information, host information, mail aliases, and so forth. They also support Solaris-specific data such as the automount maps.

Under a files-based naming system, FNS context and attribute data is stored in files. These FNS files are stored in machine's `/var/fn` directory. (The `/var/fn` directory does not have to be on each machine, it could be exported from an NFS file server.)

Under a files naming system, you use FNS commands to work with the information in FNS files.

Global Naming Services

FNS also supports federating NIS+ and NIS with DNS and X.500. This means that you can connect enterprise level namespaces with global namespaces to make the enterprise objects accessible in the global scope.

FNS currently supports the following global naming services:

- DNS
- X.500 (via DAP or LDAP)

FNS Naming Policies

FNS defines naming policies so that users and applications can depend on and use the shared namespace.

Within an enterprise, there are namespaces for organizational units, sites, hosts, users, files and services, referred to by the names `orgunit`, `site`, `host`, `user`, `fs` (for file system), and `service`. These namespaces can also be named by preceding each name with an underscore (`_`). For example, `host` and `_host` are considered identical.

Table 19-1 summarizes the FNS policies for enterprise-level namespaces.

TABLE 19-1 FNS Policy Summary

Context Type	Subordinate Contexts	Parent Contexts
<code>orgunit _orgunit</code>	<code>site user host fs service</code>	enterprise root
<code>site _site</code>	<code>user host fs service</code>	enterprise root orgunit
<code>user _user</code>	<code>service fs</code>	enterprise root orgunit
<code>host _host</code>	<code>service fs</code>	enterprise root orgunit
<code>service _service</code>	Printer and other applications	enterprise root orgunit site user host
<code>fs _fs</code> (file system)	(none)	enterprise rootorgunit site user host

Organization Names

The binding of an FNS `orgunit` is determined by the underlying naming service:

- Under NIS+, an organizational unit corresponds to an NIS+ domain or subdomain. For example, assume that the NIS+ root domain is `doc.com.` and `sales` is a subdomain of `doc.com.` Then, the FNS names `org/sales.doc.com.` and `org/sales` both refer to the organizational unit corresponding to the NIS+ domain `sales.doc.com.` (Note the trailing dot in `sales.doc.com.` which is required for fully qualified NIS+ names.)
- Under NIS, an organizational unit is the NIS domain which is always identified by the FNS name `org//` or `org/domainname` where *domainname* is a fully qualified

domain name such as `doc.com`. Under NIS, there is no hierarchy in organizational unit names.

- Under a files-based naming system, the organizational unit is the system which is always identified by the FNS name `org//`.

The types of objects that may be named relative to an organizational unit name are: `user`, `host`, `service`, `fs`, and `site`. For example:

- `org/sales/site/conference1.bldg-6` names a conference room `conference1` located in building #6 of the site associated with the organizational unit `sales`. In this example, if `org/sales` corresponds to `sales.doc.com`, another way to name this object would be: `org/sales.doc.com./site/conference1.bldg-6` (note the trailing dot in `sales.doc.com`.)
- `org/finance/user/mjones` names a user `mjones` in the organizational unit `finance`.
- `org/finance/host/inmail` names a machine `inmail` belonging to the organizational unit `finance`.
- `org/accounts.finance/fs/pub/reports/FY92-124` names a file `pub/reports/FY92-124` belonging to the organizational unit `accounts.finance`.
- `org/accounts.finance/service/calendar` names the calendar service of the organizational unit `accounts.finance`. This might manage the meeting schedules of the organizational unit.

Site Names

Site names are created as needed. The types of objects that may be named relative to a site name are: `user`, `host`, `service` and `fs`. For example:

- `site/alameda/user/mjones` names a user `mjones` at the site `alameda`.
- `site/alameda/host/sirius` names a machine `sirius` at the site `alameda`.
- `site/alameda/service/printer/Sparc-2` names the printer `Sparc-2` at the site `alameda`.
- `site/alameda/fs/usr/dist` names a file directory `usr/dist` available in the site `alameda`.

User Names

User names correspond to names in the corresponding `passwd` table in NIS+, the `passwd` map in NIS, or the `/etc/passwd` file under files. A user's file context is obtained from his or her `passwd` entry.

The types of objects that may be named relative to a user name are: `service`, and `fs`. For example:

- `user/chou/service/fax` names the fax service of the user `chou`.
- `user/esperanza/fs/projects/conf96.doc` names the file `conf96.doc` in the `projects` subdirectory of the user `esperanza`'s file system.

Host Names

Host names correspond to names in the corresponding `hosts` table in NIS+, the `hosts` map in NIS, or the `/etc/hosts` file under `files`. The host's file context corresponds to the files systems exported by the host.

The types of objects that may be named relative to a host name are: `service`, and `fs`. For example:

- `host/smtp-1/service/mailbox` names the mailbox service associated with the machine `smtp-1`.
- `host/deneb/fs/etc/.cshrc` names the file `.cshrc` in the `/etc` directory on the host `deneb`.

Service Names

Service names correspond to, and are determined by, service applications. The service context must be named relative to an organization, user, host, or site context. For example:

- `org//service/printer` names the organization's printer service.
- `host/deneb/service/printer` names the printer service associated with the machine `deneb`.
- `host/deneb/service/printer/Sparc-2` names the printer associated with the machine `deneb`.
- `user/charlie/service/calendar` names the user `charlie`'s calendar service.
- `site/conf_pine.bldg-7.alameda/service/calendar` names the calendar service for the `conf_pine` conference room in Building 7 at the Alameda site.

File Names

File system names correspond to file names. For example:

- `host/altair/fs/etc/.login` names the `.login` file on the machine `altair`.

- `user/prasad/fs/projects/96draft.doc` names the file `96draft.doc` in the user `prasad`'s `projects` directory.

Getting Started

To begin using FNS with your underlying name service, you run the `fncreate` command.

The `fncreate` command recognizes the underlying naming service in which FNS contexts are to be created (such as, NIS+, NIS, or files). To specify a specific naming service, you must run the `fnselect` command as explained in “Designating a Non-Default Naming Service” on page 333, below.

Designating a Non-Default Naming Service

By default:

- If `fncreate` is executed on a machine that is an NIS+ client or server, the FNS namespace will be set up in NIS+. (See *Solaris Naming Setup and Configuration Guide* if you want or need to designate some other machine as an FNS NIS+ master server.)
- If the machine is an NIS client, the namespace will be set up in NIS.
- If the machine is neither, the namespace will be set up in the machine's `/var/fn` directory. When your underlying naming system is files-based, the common practice is to create `/var/fn` by running `fncreate` on each machine. It is possible however to create `/var/fn` on one machine and export it by NFS to be mounted by other clients.

You can also explicitly specify a non-default target naming service by using the `fnselect` command. For example the following command selects the target naming service to be NIS.

```
# fnselect nis
```

Creating the FNS Namespace

Once the naming service has been selected either using the default policy or explicitly via `fnselect`, you can execute the following command to create the FNS namespace:

```
# fncreate -t org org//
```

This creates all the necessary contexts for users and hosts in the corresponding naming service.

NIS+ Considerations

When your primary enterprise-level naming service is NIS+, take into account the following points.

NIS+ Domains and Subdomains

The command syntax shown above creates the FNS namespace for the root NIS+ domain. To specify a domain other than the root, add the domain name between the double slashes, as in:

```
# fncreate -t org org/sales.doc.com./
```

Note the trailing dot after the fully qualified `sales.doc.com.` domain name.

Space and Performance Considerations

The `fncreate` commands creates NIS+ tables and directories in the `ctx_dir` directory. The `ctx_dir` directory object resides at the same level as the NIS+ `groups_dir` and `org_dir` directory objects of the domain.

- With a large domain, the additional space required on the NIS+ server could be substantial and in a large installation performance might be improved by using separate servers for FNS and the standard NIS+ tables. See *Solaris Naming Setup and Configuration Guide* for information on how to use separate servers for FNS and NIS+.
- In a large, or mission-critical domain, FNS service should be replicated. See *Solaris Naming Setup and Configuration Guide* for information on how to replicate FNS service.

NIS+ Security Requirements

The user who runs `fncreate` and other FNS commands is expected to have the necessary NIS+ credentials.

The environment variable `NIS_GROUP` specifies the group owner for the NIS+ objects created by `fncreate`. In order to facilitate administration of the NIS+ objects, `NIS_GROUP` should be set to the name of the NIS+ group responsible for FNS administration for that domain prior to executing `fncreate` and other FNS commands.

Changes to NIS+ related properties, including default access control rights, could be effected using NIS+ administration tools and interfaces after the context has been created. The NIS+ object name that corresponds to an FNS composite name can be obtained using `fnlookup` and `fnlist`, described later in this document.

NIS Considerations

The `fncreate` command must be executed by superuser on the NIS system that will serve as the NIS master server for the FNS maps.

The NIS maps used by FNS are stored in `/var/yp/domainname`.

Any changes to the FNS information can only be done by the superuser on the FNS NIS master server using FNS commands.

Files Considerations

When using `fncreate` with the `-t org` option to create your FNS namespace, the command must be executed by superuser on the machine that owns the file system on which `/var` is located. The files used by FNS are stored in the `/var/fn` directory.

Once users' contexts are created, users are allowed to modify their own contexts based on their UNIX credentials.

If exported, the file system `/var/fn` can be mounted by other systems to access the FNS namespace.

Browsing the FNS Namespace

Once the namespace has been set up, you can browse using the following commands:

- `fnlist` to list context contents (see “Listing Context Contents” on page 335 below)
- `fnlookup` to display the bindings of a composite name (see “Displaying the Bindings of a Composite Name” on page 336).
- `fnattr` to show the attributes of a composite name (see “Showing the Attributes of a Composite Name” on page 337).

Listing Context Contents

The `fnlist` command displays the names and references bound in the context of *name*.

```
fnlist [-lvA] [name]
```

TABLE 19-2 `fnlist` Command Options

Option	Description
<i>name</i>	A composite name. Displays the names bound in the context of <i>name</i>
<code>-v</code>	Verbose. Displays the binding in more detail
<code>-l</code>	Also displays the bindings of the names bound in the named context
<code>-A</code>	Forces <code>fnlist</code> to obtain its information from the authoritative server. Under NIS and NIS+, that is the domain master server. The <code>-A</code> option has no effect when the primary naming service is files.

For example:

To list names in the initial context:

```
% fnlist
```

To list in detail all the users in the current organizational unit:

```
% fnlist -v user
```

To list the contents of the `service` context for the user `pug`:

```
% fnlist user/pug/service
```

To list names and bindings from the authoritative server:

```
% fnlist -l -A
```

Displaying the Bindings of a Composite Name

The `fnlookup` command shows the binding of the given composite name.

```
fnlookup [-vAL] [name]
```

TABLE 19-3 `fnlookup` Command Options

Option	Description
<i>name</i>	The name of a context. Displays the binding and XFN link of <i>name</i>
<code>-v</code>	Verbose. Displays the binding in more detail
<code>-L</code>	Also displays the XFN link that the name is bound to
<code>-A</code>	Forces <code>fnlist</code> to obtain its information from the authoritative server. Under NIS and NIS+, that is the domain master server. The <code>-A</code> option has no effect when the primary naming service is files-based.

For example: to display the binding of `user/ana/service/printer`:

```
# fnlookup user/ana/service/printer
```

Showing the Attributes of a Composite Name

The `fnattr` command displays (and updates) the attributes of the given composite name.

For example, to search for the attributes associated with a user named `ada`:

```
# fnattr user/ada
```

To search for the attributes associated with a printer named `laser-9`:

```
# fnattr thisorgunit/service/printer/laser-9
```

See “Working With Attributes” on page 345 for more details.

Searching for FNS Information

The `fnsearch` command displays the names and, optionally, the attributes and references of objects bound at or below a composite name whose attributes satisfy the given search criteria.

For example:

To list the users and their attributes who have an attribute called `realname`:

```
% fnsearch user realname
```

To list the users with the attribute `realname` whose value is `Ravi Chattha`:

```
% fnsearch user ``realname == 'Ravi Chattha''
```

The `fnsearch` command uses the common Boolean operators. Note the use of double and single quotes and double equals sign in the above example.

Updating the Namespace

Once the namespace has been set up, you can add, delete, and modify elements using the following commands:

- `fnbind` to bind new references to a composite name (see “Binding a Reference to a Composite Name” on page 339, below).
- `fnunbind` to remove bindings (see “Removing Bindings” on page 341).
- `fncreate` to create new organization, user, host, site, and service contexts (see “Creating New Contexts” on page 341).
- `fncreate_fs` to create new file system contexts (see “Creating File Contexts” on page 342).
- `fncreate_printer` to create new printer contexts (see “Creating Printer Contexts” on page 343).
- `fndestroy` to destroy contexts (see “Destroying Contexts” on page 345).
- `fnattr` to display, create, modify, and remove attributes (see “Working With Attributes” on page 345).
- `fncopy` to copy FNS contexts and attributes from one naming service to another (see “Copying and Converting FNS Contexts” on page 347).

FNS Administration Privileges

FNS System administration varies according to the underlying naming service:

- *NIS+*. Under NIS+, FNS system administration tasks can only be performed by those with authorization to do so. The usual method of granting system administration privileges is to create an NIS+ group and assign that group the necessary privileges for that domain. Any member of the group can then perform system administration functions.
- *NIS*. Under NIS, FNS administration tasks must be performed by `root` on the NIS master server.
- *Files*. Under a files-based naming system, FNS administration tasks must be performed by someone with `root` access to the `/var/fn` directory.

The ability of users to make changes to their own user sub-contexts varies according to the underlying naming service:

- *NIS+*. Under NIS+, a user's context (and associated sub-contexts) are owned by them. When logged in as an NIS+ principle, users who have the appropriate credentials and privileges can make changes to their own context using the `fncreate`, `fnbind`, `fnunbind`, and similar commands.
- *NIS*. Under NIS, users cannot make any changes to any FNS data. Only those with `root` access on the NIS master server can change FNS data.
- *Files*. Under a files-based naming system, users own their own contexts. Standard UNIX access controls apply to FNS files.

Binding a Reference to a Composite Name

The `fnbind` command is used to bind an existing reference (name) to a new composite name.

```
fnbind -r [-s][-v][-L] name [-O|-U] newname reftype addrtype [-c|-x] address
```

TABLE 19-4 `fnbind` Command Options

Option	Description
<i>name</i>	The existing composite name
<i>newname</i>	The composite name of the new binding
<i>addrtype</i>	Address type to use. Applications-specific such as <code>onc_cal_str</code> .
<i>address</i>	Address contents to use. For example, <code>tsvi@altair</code> .
<i>reftype</i>	Reference type to use. Applications-specific such as <code>one_calendar</code> .
<code>-s</code>	Bind to <i>newname</i> even if it is already bound. This replaces the previous binding of <i>newname</i> . Without <code>-s</code> , <code>fnbind</code> fails if <i>newname</i> is already bound.
<code>-v</code>	Display the reference that will be bound to <i>newname</i> .
<code>-L</code>	Create an XFN link using <i>oldname</i> and bind it to <i>newname</i> .

TABLE 19-4 `fnbind` Command Options (continued)

Option	Description
<code>-r</code>	Bind <i>newname</i> to the reference constructed by the command line arguments.
<code>-c</code>	Store <i>address</i> contents in the form as entered, do not use XDR-encoding.
<code>-x</code>	Convert <i>address</i> to a hexadecimal string without converting it to XDR-encoding.
<code>-O</code>	The identifier format is <code>FN_ID_ISO_OID_STRING</code> , an ASN.1 dot-separated integer list string.
<code>-U</code>	The identifier format is <code>FN_ID_DCE_UUID</code> , a DCE UUID in string form.

For example:

To add a calendar binding for the user `jamal`:

```
# fnbind -r user/jamal/service/calendar onc_calendar onc_cal_str
jamal@cygnus
```

To replace the existing binding of `org//service/Sparc-4` with that of `org//service/printer`:

```
# fnbind -s org//service/printer org//service/Sparc-4
```

To copy the reference `site/bldg-5/service/printer` to `user/ando/service/printer`:

```
# fnbind site/bldg-5/service/printer user/ando/service/printer
```

To bind the reference `site/bldg-5/service/printer` to `user/ando/service/printer` using a symbolic link:

```
# fnbind -L site/bldg-5/service/printer user/ando/service/printer
```

To bind the name `thisens/service/calendar` to the address `staff@altair`, when `staff@altair` is a reference of the type `onc_cal` and an address of the type `onc_cal_str`:

```
# fnbind -r thisens/service/calendar onc_calendar onc_cal_str staff@altair
```

To bind *newname* to the reference constructed by its command line *address*

```
# fnbind -r [-sv] newname [-O|-U] reftype {[-O|-U] addrtype [-c|-x] address}
```

Removing Bindings

The `fnunbind` *name* command is used to remove bindings.

For example: to remove the binding for `user/jsmith/service/calendar`:

```
# fnunbind user/jsmith/service/calendar
```

Creating New Contexts

The `fncreate` command is used to create contexts.

```
fncreate -t context [-f file] [-o] [-r reference] [-s] [-v] [-D] name
```

TABLE 19-5 `fncreate` Command Options

Option	Description
<code>-t context</code>	Create context of type <i>context</i> . <i>Context</i> types can be: <code>org</code> , <code>hostname</code> , <code>host</code> , <code>username</code> , <code>user</code> , <code>service</code> , <code>fs</code> , <code>site</code> , <code>nsid</code> , and <code>generic</code> .
<code>-f file</code>	Use an input file to list users and hosts for whom to create contexts.
<code>-r reference</code>	Type of reference. The <code>-r reference</code> option can only be used with <code>-t generic</code> .
<i>name</i>	A composite name
<code>-o</code>	Create only the context identified by <i>name</i> .
<code>-s</code>	Overwrite (supersede) any existing binding. If <code>-s</code> is not used, <code>fncreate</code> will fail if <i>name</i> is already bound.
<code>-D</code>	Display information about each context and corresponding tables, directories, and files as it is created.
<code>-v</code>	Verbose. Display information about each context as it is displayed.

For example:

To create a context and subcontexts for the root organization:

```
# fncreate -t org org//
```

To create a context, and subcontexts, for the host `deneb`:

```
# fncreate -t host host/deneb
```

To create a context, service and file subcontexts, and then add a calendar binding for the user `sisulu`:

```
# fncreate -t user user/sisulu
# fnbind -r user/sisulu onc_calendar onc_cal_str sisulu@deneb
```

To create a site context for the `sales` organization:

```
# fncreate -t site org/sales/site/
```

The site context supports a hierarchal namespace, with dot-separated right-to-left names, which allows sites to be partitioned by their geographical coverage relationships. For example, to create a site context `alameda` and a site subcontext `bldg-6.alameda` for it:

```
# fncreate -t site org/sales/site/alameda
# fncreate -t site org/sales/site/bldg-6.alameda
```

Creating File Contexts

- The `fncreate_fs` command creates file contexts for organizations and sites with the description of the binding entered from the command line.

```
fncreate_fs [-r] [-v] name [options] [mount]
```

- The `fncreate_fs` command creates file contexts for organizations and sites with the description of the bindings supplied by an input file.

```
fncreate_fs [-r] [-v] -f file name
```

TABLE 19-6 `fncreate_fs` Command Options

Option	Description
<i>name</i>	The name of the file context
<i>options</i>	Mount options

TABLE 19-6 `fncreate_fs` Command Options (continued)

Option	Description
<code>mount</code>	Mount location
<code>-f file</code>	Input file
<code>-v</code>	Verbose. Displays information about the contexts being created
<code>-r</code>	Replace the bindings in the context <i>name</i> with those specified in the input.

For example:

To create a file system context named `data` for the `sales` organization bound to the `/export/data` path of an NFS server named `server4`.

```
# fncreate_fs org/sales/fs/data server4:/export/data
```

To create a hierarchy of file system contexts for the `sales` organization named `buyers` and `buyers/orders` mounted on two different servers:

```
# fncreate_fs org/sales/fs/buyers server2:/export/buyers
# fncreate_fs org/sales/fs/buyers/orders server3:/export/orders
```

To create a file system context named `leads` for the `sales` organization bound to a server and path specified by an input file named `input_a`:

```
# fncreate_fs -f input_a org/sales/fs/leads
```

(See the `fncreate_fs` man page for information on input file format.)

Creating Printer Contexts

The `fncreate_printer` command creates printer contexts for organizations, users, hosts and site contexts. The printer context is created under the service context of the respective composite name.

```
fncreate_printer [-vs] name printer [prtaddr]
```

```
fncreate_printer [-vs] [-f [file]] name
```

TABLE 19-7 `fncreate_printer` Command Options

Option	Description
<i>name</i>	The name of the org, host, user, or site of the printer
<i>printer</i>	The name of the printer
<i>prntaddr</i>	The printer address in the form <addresstype>=<address>
<code>-f file</code>	Use the named <i>file</i> as input for a list of printers to be created. The input file is in the format of the <code>/etc/printers.conf</code> file. If neither a printer <i>name</i> nor a <code>-f file</code> is specified, <code>fncreate_printer</code> uses the <code>/etc/printer.conf</code> file on the machine where <code>fncreate_printer</code> is run as a default input file.
<code>-s</code>	Replace an existing address with the same address-type.
<code>-v</code>	Verbose. Displays the binding in more detail

For example:

To create printers for the `sales` organization based on the printers listed in the `/etc/printers.conf` file of the machine on which `fncreate_printer` is run:

```
# fncreate_printer -s org/sales/
```

Assume that the machine `altair` is the server for a printer named `Sparc-5`. To create a printer named `invoices` for the user `nguyen` that is actually the `Sparc-5` printer:

```
# fncreate_printer user/nguyen invoices bsdaddr=altair,Sparc-5
```

It is also possible to organize printers hierarchically. For example, the `fncreate_printer` command can create printer contexts for the printers, `color`, `color/inkjet` and `color/Sparc` with the resulting contexts:

```
org/doc.com/service/printer/color
org/doc.com/service/printer/color/inkjet
org/doc.com/service/printer/color/Sparc
```

To create the above contexts, you would run:

```
# fcreate_printer org/doc.com color bsdaddr=colorful,color
# fcreate_printer org/doc.com color/inkjet bsdaddr=colorjet,inkjet
# fcreate_printer org/doc.com color/Sparc bsdaddr=colorprt,Sparc
```

Destroying Contexts

The `fndestroy` command is used to destroy empty contexts.

For example, to destroy the `service` context of the user `patel`:

```
# fndestroy user/patel/service
```

Working With Attributes

The `fnattr` command can be used to add, delete or modify attributes associated with a name. You can make modifications one at a time, or batch several within the same command.

- `fnattr [-l] name` to list attributes for *name*.
- `fnattr name -a-s -U -O attrib values` to add an attribute
- `fnattr name -m -O -U attrib oldvalue newvalue` to modify an attribute
- `fnattr name -d -O | -U [values attrib]` to destroy an attribute

TABLE 19-8 `fnattr` Command Options

Option	Description
<i>name</i>	The composite name
<i>attrib</i>	The identifier of an attribute
<i>values</i>	One or more attribute values
<i>oldvalue</i>	An attribute value to be replaced by a new value
<i>newvalue</i>	The attribute value that replaces an old value
<code>-a</code>	Add an attribute
<code>-d</code>	Destroy an attribute

TABLE 19-8 `fnattr` Command Options (continued)

Option	Description
<code>-l</code>	List attributes
<code>-m</code>	Modify an attribute
<code>-s</code>	Replace all old attribute values with the new values for the attribute specified.
<code>-O</code>	The identifier format is <code>FN_ID_ISO_OID_STRING</code> , an ASN.1 dot-separated integer list string.
<code>-U</code>	The identifier format is <code>FN_ID_DCE_UUID</code> , a DCE UUID in string form.

For example:

To show all of the attributes associated with the user name `rosa`:

```
# fnattr user/rosa
```

To display the `size` attribute associated with the user `uri`:

```
# fnattr user/uri/ size
```

For a user named `devlin`, to add an attribute named `shoesize` with a value of `small`, delete the `hatsize` attribute, and change the `dresssize` attribute value from `12` to `8`:

```
# fnattr user/devlin -a shoesize small -d hatsize -m dresssize 12 8
```

Federating a Global Namespace

You can federate NIS+ or NIS to a global naming service like DNS and X.500.

To federate an NIS+ or NIS namespace under DNS or X.500, you first need to obtain the root reference for the NIS+ hierarchy or NIS domain.

From the point of view of the global name service, the root reference is known as the *next naming system reference* because it refers to the next naming system beneath the DNS domain or X.500 entry. To federate NIS+ or NIS with a global name service, you add the root reference information to that global service.

Once you have added the root reference information to the global service, clients outside of your NIS+ hierarchy or NIS domain can access and perform operations on the contexts in the NIS+ hierarchy or NIS domain. Foreign NIS+ clients access the hierarchy as unauthenticated NIS+ clients.

For example:

If NIS+ is federated underneath the DNS domain `doc.com`, you can now list the root of the NIS+ enterprise using the command

```
# fnlist ../doc.com/
```

If NIS+ is federated underneath the X.500 entry `/c=us/o=doc`, you can list the root of the NIS+ enterprise using the command:

```
# fnlist ../c=us/o=doc/
```

Note the mandatory trailing slash in both examples.

Copying and Converting FNS Contexts

The `fncopy` command can be used to copy or convert an FNS context and attributes to a new FNS context.

By using the `-i` and `-o` options, you can copy FNS contexts based on one underlying enterprise-level name service to a context based on a different underlying name service. For example, if you have an FNS installation running on top of NIS, and you upgrade your NIS service to NIS+, you can use `fncopy` to create a new context using NIS+.

Note that:

- If the new FNS context that you are copying an old context to already exists for the target name service, only new contexts and bindings are copied. The contexts are not over-written or changed
- `fncopy` does not follow links, but copies the FNS link bound to a name to the new context namespace.

TABLE 19-9 `fnccopy` Command Options

Option	Description
<code>-i oldservice</code>	The old (input) underlying enterprise-level name service. For example, <code>-i nis</code> specifies that the old service is NIS. Allowed values are <code>files</code> , <code>nis</code> , <code>nisplus</code> .
<code>-o newservice</code>	The new (output) underlying enterprise-level name. For example, <code>o nisplus</code> specifies that the new service is NIS+. Allowed values are <code>files</code> , <code>nis</code> , <code>nisplus</code> .
<code>-f filename</code>	A text file listing FNS contexts to be copied. In the absence of the <code>-i</code> and <code>-o</code> options, contexts must be identified using global names.
<code>oldcontext</code>	The name of the context being copied
<code>newcontext</code>	The name of the context being created or copied to.

For example, to copy the `doc.com` printer contexts (and sub-contexts) and bindings to `orgunit/east/doc.com`:

```
# fnccopy ../doc.com/service/printer ../doc.com/orgunit/east/service/printer
```

To copy the NIS FNS users' contexts specified in the file `user_list` to a NIS+ FNS users' context of the `orgunit west/doc.com`:

```
# fnccopy -i nis -o nisplus -f /etc/user_list thisorgunit/user org/doc.com/user
```

Namespace Browser Programming Examples

The programming examples in this section show the usage of XFN APIs to perform the following operations:

- “Listing Names Bound in a Context” on page 349.
- “Creating a Binding” on page 350.
- “Listing and Working With Object Attributes” on page 351.
- “Adding, Deleting, and Modifying an Object's Attributes” on page 353.

- “Searching for Objects in a Context” on page 355.

Listing Names Bound in a Context

The example below shows XFN operations to list a context.

```
#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
#include <stdlib.h>
/*
   This routine returns the list of names
   bound under the given context (ctx_name).
   Examples of ctx_name are "user", "thisorgunit/service",
   host/alto/service, user/jsmit/service/calendar, etc..
 */
typedef struct fns_listing {
    char *name;
    struct fns_listing *next;
} fns_listing;
fns_listing *
fns_list_names(const char *ctx_name)
{
    FN_status_t *status;
    FN_ctx_t *initial_context;
    FN_composite_name_t *context_name;
    FN_namelist_t *name_list;
    FN_string_t *name;
    unsigned int stat;
    fns_listing *head = 0, *current, *prev;
    int no_names = 0;
    status = fn_status_create();
    /* Obtain the initial context */
    initial_context = fn_ctx_handle_from_initial(0, status);
    if (!fn_status_is_success(status)) {
        fprintf(stderr, "Unable to obtain initial context\n");
        return (0);
    }
    context_name = fn_composite_name_from_str((unsigned char *)
        ctx_name);
    /* FNS call to list names */
    name_list = fn_ctx_list_names(initial_context, context_name,
        status);
    if (!fn_status_is_success(status)) {
        fprintf(stderr, "Unable to list names\n");
        return (0);
    }
    /* Obtain the names individually */
    while (name = fn_namelist_next(name_list, status)) {
        no_names++;
        current = (fns_listing *) malloc(sizeof(fns_listing));
        current->name = (char *)
            malloc(strlen((char *) fn_string_str(name, &stat)) + 1);
        strcpy(current->name, (char *) fn_string_str(name, &stat));
    }
}
```

(continued)

```

current->next = 0;
if (head) {
    prev->next = current;
    prev = current;
} else {
    head = current;
    prev = current;
}
fn_string_destroy(name);
}
fn_namelist_destroy(name_list);
fn_status_destroy(status);
fn_ctx_destroy(initial_context);
return (head);

```

Creating a Binding

CODE EXAMPLE 19-1 Creating a Binding

The example below shows how to create a binding.

```

#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
/*
This routine creates a binding with a name provided by "name"
and having a reference type "reference_type" and address type
"address_type".
An example of using the function could be:
    fns_create_bindings(
        "user/jsmith/service/calendar",
        "onc_calendar",
        "onc_cal_str",
        "jsmith&calserver");
*/
int fns_create_bindings(
    char *name,
    char *reference_type,
    char *address_type,
    char *data)
{
    int return_status;
    FN_composite_name_t *binding_name;
    FN_identifier_t ref_id, addr_id;
    FN_status_t *status;
    FN_ref_t *reference;
    FN_ref_addr_t *address;
    FN_ctx_t *initial_context;
    /* Obtain the initial context */

```

(continued)

```

status = fn_status_create();
initial_context = fn_ctx_handle_from_initial(0, status);
/* Check status for any error messages */
if ((return_status = fn_status_code(status)) != FN_SUCCESS) {
    fprintf(stderr, "Unable to obtain the initial context\n");
    return (return_status);
}
/* Get the composite name for the printer name */
binding_name = fn_composite_name_from_str((unsigned char *) name);
/* Construct the Address */
addr_id.format = FN_ID_STRING;
addr_id.length = strlen(address_type);
addr_id.contents = (void *) address_type;
address = fn_ref_addr_create(&addr_id,
    strlen(data), (const void *) data);
/* Construct the Reference */
ref_id.format = FN_ID_STRING;
ref_id.length = strlen(reference_type);
ref_id.contents = (void *) reference_type;
reference = fn_ref_create(&ref_id);
/* Add Address to the Reference */
fn_ref_append_addr(reference, address);

/* Create a binding */
fn_ctx_bind(initial_context, binding_name, reference, 0, status);
/* Check the error status and return */
return_status = fn_status_code(status);
fn_composite_name_destroy(binding_name);
fn_ref_addr_destroy(address);
fn_ref_destroy(reference);
fn_ctx_destroy(initial_context);
return (return_status);
}

```

Listing and Working With Object Attributes

The examples below show techniques to list and work with attributes of an object.

Listing an Object's Attributes

The example below shows how to list the attributes of an object.

```

#include <stdio.h>
#include <xfn/xfn.h>
/*
    This routine prints all the attributes associated
    with the named object to the standard output.

```

(continued)

```

Examples of using the function:
    fns_attr_list("user/jsmith");
    fns_attr_list("thisorgunit/service/printer/color");
*/
void fns_attr_list(const char *name)
{
    FN_composite_name_t *name_comp;
    const FN_identifier_t *identifier;
    FN_attribute_t *attribute;
    const FN_attrvalue_t *values;
    char *id, *val;
    FN_multigetlist_t *attrset;
    void *ip;
    FN_status_t *status;
    FN_ctx_t *initial_context;
    name_comp = fn_composite_name_from_str((unsigned char *) name);
    status = fn_status_create();
    /* Obtain the initial context */
    initial_context = fn_ctx_handle_from_initial(0, status);
    if (!fn_status_is_success(status)) {
        fprintf(stderr, "Unable to obtain intial context\n");
        return;
    }
    /* Obtain all the attributes */
    attrset = fn_attr_multi_get(initial_context, name_comp, 0, 0,
                               status);
    if (!fn_status_is_success(status)) {
        fprintf(stderr, "Unable to obtain attributes\n");
        return;
    }
    /* List all attributes */
    while (attribute = fn_multigetlist_next(attrset, status)) {
        identifier = fn_attribute_identifier(attribute);
        switch(identifier->format) {
            case FN_ID_STRING:
                id = (char *) malloc(identifier->length + 1);
                memcpy(id, identifier->contents, identifier->length);
                id[identifier->length] = '\0';
                printf("Attribute Identifier: %s", id);
                free(id);
                break;
            default:
                printf("Attribute of non-string format\n\n");
                continue;
        }
        for (values = fn_attribute_first(attribute, &ip);
            values != NULL;
            values = fn_attribute_next(attribute, &ip)) {
            val = (char *) malloc(values->length + 1);
            memcpy(val, values->contents, values->length);
            val[values->length] = '\0';
            printf("Value: %s", val);
            free(val);
        }
    }
}

```

(continued)

```

    }
    fn_attribute_destroy(attribute);
    printf("\n");
}
fn_multigetlist_destroy(attrset);
fn_ctx_destroy(initial_context);
fn_status_destroy(status);
fn_composite_name_destroy(name_comp);
}

```

Adding, Deleting, and Modifying an Object's Attributes

The example below shows how to add, delete, or modify an object's attributes.

```

#include <stdio.h>
#include <xfn/xfn.h>
/*
  This routine modifies an attribute associated
  with the named object. The modify operation supported are:
  FN_ATTR_OP_ADD
  FN_ATTR_OP_ADD_EXCLUSIVE
  FN_ATTR_OP_REMOVE
  FN_ATTR_OP_ADD_VALUES
  FN_ATTR_OP_REMOVE_VALUES
  The function assumes the attribute values to be strings.
  Examples of using the function:
  The following function add an attribute of identifier "realname"
  with value "James Smith" to the user object "user/jsmith".
  fns_attr_modify(
    "user/jsmith",
    "realname",
    "James Smith",
    FN_ATTR_OP_ADD);
  The following function removes an attribute of identifier
  "location" from the printer object
  "thisorgunit/service/printer/color".
  fns_attr_modify(
    "thisorgunit/service/printer/color",
    "location",
    NULL,
    FN_ATTR_OP_REMOVE);
*/
static const char *attr_id_syntax = "fn_attr_syntax_ascii";
void fns_attr_modify(const char *name,
  const char *attr_id,
  const char *attr_value,
  unsigned int operation)
{
  FN_composite_name_t *name_comp;
  FN_identifier_t identifier, syntax;

```

(continued)

```

FN_attrvalue_t *values;
FN_attribute_t *attribute;
FN_status_t *status;
FN_ctx_t *initial_context;
name_comp = fn_composite_name_from_str((unsigned char *) name);
status = fn_status_create();
/* Obtain the initial context */
initial_context = fn_ctx_handle_from_initial(0, status);
if (!fn_status_is_success(status)) {
    fprintf(stderr, "Unable to obtain initial context\n");
    return;
}
/* Create the attribute to be added */
/* First, the identifier */
identifier.format = FN_ID_STRING;
identifier.length = strlen(attr_id);
identifier.contents = (void *) strdup(attr_id);
/* Second, the syntax */
syntax.format = FN_ID_STRING;
syntax.length = strlen(attr_id_syntax);
syntax.contents = (void *) strdup(attr_id_syntax);
/* Third, the attribute value */
if (attr_value) {
    values = (FN_attrvalue_t *) malloc(sizeof(FN_attrvalue_t));
    values->length = strlen(attr_value);
    values->contents = (void *) strdup(attr_value);
} else
    values = NULL;
/* Fourth, create the attribute */
attribute = fn_attribute_create(&identifier, &syntax);
/*Fifth, add the attribute value */
if (values)
    fn_attribute_add(attribute, values, 0);

/* Perform the XFN operation */
fn_attr_modify(initial_context, name_comp, operation, attribute, 0,
status);
if (!fn_status_is_success(status))
    fprintf(stderr, "Unable to perform attribute operation\n");
fn_ctx_destroy(initial_context);
fn_status_destroy(status);
fn_composite_name_destroy(name_comp);
fn_attribute_destroy(attribute);
free(identifier.contents);
free(syntax.contents);
if (values) {
    free(values->contents);
    free(values);
}
]

```

Searching for Objects in a Context

The example below shows how to search for objects in a context with a specific attribute identifier and value.

```
#include <stdio.h>
#include <xfn/xfn.h>
#include <string.h>
#include <stdlib.h>
/*
   This routine searches for objects in a context
   which has the specified attribute identifier and value.
 */
typedef struct fns_search_results {
    char *name;
    struct fns_search_results *next;
} fns_search_results;
static const char *attr_id_syntax = "fn_attr_syntax_ascii";
fns_search_results *
fns_attr_search(const char *name,
               const char *attr_id,
               const char *attr_value)
{
    FN_status_t *status;
    FN_ctx_t *initial_context;
    FN_composite_name_t *context_name;
    FN_searchlist_t *search_list;
    FN_string_t *search_name;
    FN_attribute_t *attribute;
    FN_attrset_t *attrset;
    FN_identifier_t identifier, syntax;
    FN_attrvalue_t *values;
    unsigned stat;
    fns_search_results *head = 0, *current, *prev;
    int no_names = 0;
    context_name = fn_composite_name_from_str((unsigned char *) name);
    status = fn_status_create();
    initial_context = fn_ctx_handle_from_initial(0, status);
    if (!fn_status_is_success(status)) {
        fprintf(stderr, "Unable to obtain initial context\n");
        return (0);
    }
    /* Construct the attrset with attributes to be searched */
    /* First, the identifier */
    identifier.format = FN_ID_STRING;
    identifier.length = strlen(attr_id);
    identifier.contents = (void *) strdup(attr_id);
    /* Second, the syntax */
    syntax.format = FN_ID_STRING;
    syntax.length = strlen(attr_id_syntax);
    syntax.contents = (void *) strdup(attr_id_syntax);
    /* Third, the attribute value */
    values = (FN_attrvalue_t *) malloc(sizeof(FN_attrvalue_t));
    values->length = strlen(attr_value);
    values->contents = (void *) strdup(attr_value);
    /* Fourth, create the attribute */
    attribute = fn_attribute_create(&identifier, &syntax);
```

(continued)

```

/* Fifth, add the attribute value */
fn_attribute_add(attribute, values, 0);
/* Sixth, create attrset, and add the attribute */
attrset = fn_attrset_create();
fn_attrset_add(attrset, attribute, 0);
search_list = prelim_fn_attr_search(initial_context,
context_name, attrset, 0, 0, status);
if (!fn_status_is_success(status)) {
    fprintf(stderr, "Unable to list names\n");
    return (0);
}
while (search_name = prelim_fn_searchlist_next(search_list,
0, 0, status)) {
    no_names++;
    current = (fns_search_results *)
        malloc(sizeof(fns_search_results));
    current->name = (char *)
        malloc(strlen((char *) fn_string_str(search_name, &stat)) + 1);
    strcpy(current->name, (char *) fn_string_str(search_name, &stat));
    current->next = 0;
    if (head) {
        prev->next = current;
        prev = current;
    } else {
        head = current;
        prev = current;
    }
    fn_string_destroy(search_name);
}
fn_searchlist_destroy(search_list);
fn_status_destroy(status);
fn_ctx_destroy(initial_context);
fn_attrset_destroy(attrset);
fn_attribute_destroy(attribute);
free(identifier.contents);
free(syntax.contents);
free(values->contents);
free(values);
return (head);
}

```

Federated Naming Overview

This chapter describes the Federated Naming Service (FNS) which is Sun's implementation of the X/Open XFN federated naming standard.

- "XFN and FNS" on page 357
- "The XFN Model" on page 359
- "Federated Naming Service" on page 364
- "FNS in the Solaris Environment" on page 366
- "Solaris Enterprise-Level Naming Services" on page 366
- "FNS and NIS+ Naming" on page 367
- "FNS and NIS Naming" on page 367
- "FNS and Files-Based Naming" on page 368
- "Global Naming Services" on page 368
- "FNS and Applications" on page 370
- "FNS File Naming" on page 370
- "FNS Printer Naming" on page 370
- "FNS Application Support" on page 370
- "Administering FNS" on page 371
- "Troubleshooting and Error Messages" on page 372

XFN and FNS

Different name services are often embedded in different applications and services in a computing environment. Working with different name services presents significant

difficulties to the application developer. Most applications are designed to use a single name service and have very limited access to objects in a distributed computing environment. Because different applications use different name services they expect names to be composed differently. They often use different names for what the user considers very similar objects. For example, you might be able to send mail to your friend Johanna using her name `johanna@admin.doc.com`, but be required to use another name, `jsmith@altair`, to access her calendar.

FNS, Sun's implementation of the XFN standard, allows you to name objects in a uniform way, yet still provide the functionality that applications and developers need.

- *XFN* is X/Open Federated Naming. XFN is a standard actively supported by organizations such as SunSoft, Inc., IBM, Hewlett-Packard, DEC, Siemens, and OSF. FNS, the Solaris 2.6 release implementation of XFN, is compliant with the *X/Open Preliminary Specification for Federated Naming* (July 1994). Applications that use FNS are portable across platforms because the interface exported by FNS is XFN, a public, open interface endorsed by other vendors and X/Open. The X/Open Co. Ltd. is an international standards organization committed to defining computing standards that are endorsed and adhered to by the major computer vendors.
- *FNS* provides a method for federating multiple name services under a single, simple uniform interface for the basic naming operations. The service supports resolution of composite names—names that span multiple name systems—through the naming interface. Each member of a federation has autonomy in its choice of naming conventions, administrative interfaces, and its particular set of operations other than name resolution.

In the Solaris environment, the FNS implementation consists of a set of enterprise-level name services (see “Solaris Enterprise-Level Naming Services” on page 366) with specific policies and conventions for naming organizations, users, hosts, sites, and services as well as support for global name services (see “Global Naming Services” on page 368) such as DNS and X.500.

FNS is useful for the following reasons:

- A single uniform naming interface is provided to clients for accessing different name services. As a consequence, the addition of new name services does not require changes to applications or to existing member name services.
- Names can be composed in a uniform way, and the resulting composite names can have any number of components.
- Coherent naming is encouraged through the use of shared contexts and shared names.

Note - In this manual it is important to distinguish between XFN and FNS. The FNS policies include some extensions to XFN policies, and these are explicitly defined with notes. Objects belonging to the XFN programming interface are designated as XFN objects to avoid confusion with other programming interfaces.

The XFN Model

This section describes the XFN naming model from several perspectives.

XFN Architectural Model

The primary services provided by a federated naming system are mapping a composite name to a reference and providing access to attributes associated with a named object. This section defines the elements of the XFN naming model.

Atomic Names

The smallest, indivisible component of a name is called an *atomic name*. For example, the machine name `nismaster` or the user name `chou`. An atomic name may have one or more attributes, or no attributes at all (see “Attributes” on page 360).

References

A reference is the information on how to reach an object. A reference contains a list of addresses. An address identifies a communication endpoint (an object). For example, the address of a machine such as `nismaster.doc.com`, or a user’s email address such as `chou@doc.com`.

A reference might contain multiple addresses that identify multiple communication endpoints for a single conceptual object or service. For example, a list of addresses might be required because the object is distributed or because the object can be accessed through more than one communication mechanism.

Note - XFN cannot guarantee specific properties of addresses such as their stability, validity, or reachability. A client might be able to look up a name but not be able to use the returned reference because the client might not have support for any of the necessary communication mechanisms or might lack the necessary network connectivity to reach the address. Further, the address might be invalid from that origin or stale; these issues are the province of the convention between the name’s binder, the clients, and the service provider specified in the address.

Contexts

A context is a set of atomic names bound to references, as shown in Figure 20-1. Every context has an associated naming convention. A context provides a lookup

(resolution) operation, which returns the reference, and may provide operations such as binding names, unbinding names, and listing bound names. Contexts are at the heart of the lookup and binding operations.

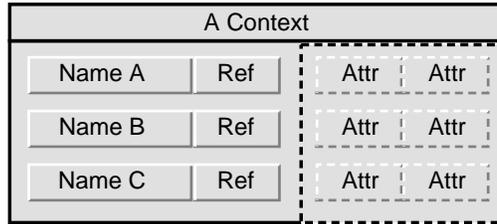


Figure 20-1 An XFN 'Context

Attributes

Attributes may be applied to named objects. Attributes are optional. A named object can have no attributes, one attributes, or multiple attributes.

Each attribute has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values. Attributes are indicated by the dotted lines in Figure 20-1 above.

XFN defines the base attribute interface for examining and modifying the values of attributes associated with existing named objects. These objects can be contexts or any other types of objects. Associated with a context are syntax attributes that describe how the context parses compound names.

The extended attribute interface contains operations that search for specific attributes and that create objects and their associated attributes.

Compound Names

A compound name is a sequence of one or more atomic names. An atomic name in one context can be bound to a reference to another context of the same type, called a subcontext. Objects in the subcontext are named using a compound name.

Compound names are resolved by looking up each successive atomic name in each successive context.

A familiar analogy for UNIX users is the file naming model, where directories are analogous to contexts, and path names serve as compound names. Furthermore, contexts can be arranged in a "tree" structure, just as directories are, with the compound names forming a hierarchical namespace.

For example:

- *UNIX*: `usr/local/bin`. UNIX atomic names are ordered from left to right and are delimited by slash (/) characters. The name `usr` is bound to a context in which `local` is bound. The name `local` is bound to a context in which `bin` is bound.

- *DNS*: `sales.doc.com`. DNS atomic names are ordered from right to left, and are delimited by dot (.) characters. The domain name `com` is bound to a context in which `doc` is bound. `doc` is bound to a context in which `sales` is bound.
- *X.500*: `c=us/o=doc/ou=sales`. An X.500 atomic name comprises an attribute type and an attribute value. Atomic names are known as *relative distinguished names* in X.500. In this string representation, X.500 atomic names are ordered from left to right, and are delimited by slash (/) characters. An attribute type is separated from an attribute value by an equal sign (=) character. Abbreviations are defined for commonly used attribute types (for example, “c” represents country name). The country name `US` is bound to a context in which `doc` is bound. The organization name `doc` is bound to a context in which the organizational unit name `sales` is bound.

Composite Names

A composite name is a name that spans multiple naming systems. Each component is a name from the namespace of a single naming system. Composite name resolution is the process of resolving a name that spans multiple naming systems.

Components are separated by slashes (/) and ordered from left to right, according to XFN composite name syntax. For example, the composite name

```
sales.doc.com/usr/local/bin
```

has two components, a DNS name (`sales.doc.com`) and a UNIX path name (`usr/local/bin`).

FNS Namespaces

Atomic names and reference addresses may also be resolved relative to one or more *namespaces*. By default, FNS provides six namespaces: `org` (for organization), `site`, `host`, `user`, `service`, and `fs` (for files).

FNS policies are used to determine how names associated with namespaces relate to each other. For example; a user is named `sergei` in the user namespace and is identified as `/user/sergei`. A calendar application is named in the service namespace and is identified as `/service/calendar`. With this system, you can then identify Sergei’s calendar service as: `/user/sergei/service/calendar`. (See “Introduction to FNS and XFN Policies” on page 374 for more information on namespaces and how they are used.)

If an application is expecting you to type a user name, the application can include the namespace identifier `user/` in front of names that you enter. If the application needs to name one of the user’s services, such as the user’s default fax machine, it can append the `service` namespace and the name of the service (`/service/fax`), to the input supplied. Hence, a fax tool might take as input the user name `jacques` and then compose the full name `user/jacques/service/fax` for the default fax

of the user `jacques`. Similarly, to access a person's calendar, you just need to type the person's user name. The application takes the input, `raj`, and uses it to construct the composite name, in this case, `user/raj/service/calendar`.

XFN Links

An XFN link is a special form of a reference that is bound to an atomic name in a context. Instead of an address, a link contains a composite name. Many naming systems support a native notion of link that can be used within the naming system itself. XFN does not specify whether there is any relationship between such native links and XFN links.

Initial Context

Every XFN name is interpreted relative to some context, and every XFN naming operation is performed on a context object. The *initial context* object provides a starting point for the resolution of composite names. The XFN interface provides a function that allows the client to obtain an initial context.

The policies described in Chapter 21, specify a set of names that the client can expect to find in this context and the semantics of their bindings. This provides the initial pathway to other XFN contexts.

User's View

Users experience federated naming through applications. Typically, the user does not need to compose or know the full composite name of objects because the application takes care of constructing the composite names. This allows the user to interact with XFN-aware applications in a simple, intuitive, and consistent manner.

File System View

Users and applications also experience federated naming through the file system. The initial context is located under `/xfn` in the root directory. For example, user `ingrid`'s `to_do` file has the XFN name, `xfn/user/ingrid/fs/to_do`.

To read this file, you could type:

```
% cat /xfn/user/ingrid/fs/to_do
```

Applications access the files under `/xfn` just as they do any other files. Applications do not need to be modified in any way, nor do they need to use the XFN API.

Application View

The way that client applications interact with XFN to access different naming systems is illustrated in a series of figures. Figure 20-2 shows an application that uses the XFN API and library.

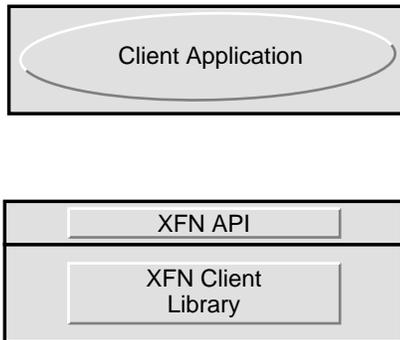


Figure 20-2 Client Application Interaction With XFN

Figure 20-3 shows the details beneath the API. A name service that is federated is accessed through the XFN client library and a *context shared object module*. This module translates the XFN calls into name service-specific calls.

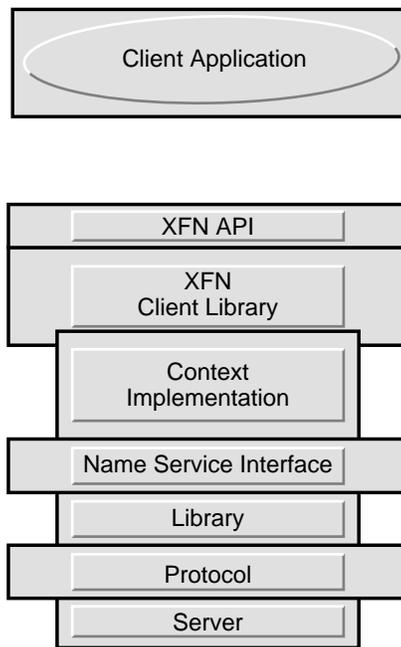


Figure 20-3 Details Beneath XFN API

API Usage Model

Many clients of the XFN interface are only interested in lookups. Their usage of the interface amounts to:

- Obtaining the initial context
- Looking up one or more names relative to the initial context

Once the client obtains a desired reference from the lookup operation, it constructs a client-side representation of the object from the reference.

Federated Naming Service

Within the Solaris environment, name services are integrated into other services such as the file system, the network information service, the mail system, and the calendar service. For example, the file system includes a naming system for files and directories; NIS+ service combines a naming system with a specialized information service.

Without FNS, users of the Solaris environment must use different, inconsistent names to refer to objects. For example: you might use the name `jsmith@admin` to send mail to Joan, the name `jsmith@altair` to access her calendar, and the name `/home/jsmith/.cshrc` to reach a file in her home directory. This disparity makes it hard for users to formulate names and hard for applications to automatically generate names on behalf of users. FNS policies define a coherent way for naming these objects.

FNS and Application Development

Applications also need naming services and applications in the Solaris environment must often deal with a diversity of name service interfaces. An application might also be exposed to a variety of often incompatible naming systems external to the Solaris environment. Local- and wide-area networks connect a heterogeneous array of hardware and operating systems, increasing the variety of potential interfaces. Not only do these naming interfaces differ widely, but the essential naming operations are often obscure.

FNS simplifies these problems in two ways:

- It provides a single standard interface to the basic naming functions that developers can use for their applications.
- It permits changes or additions to network name services without changing existing applications.

FNS and Composite Names

Some applications use composite names to access objects in the Solaris environment. The commands `mail` and `rcp` are examples of such applications.

`rcp` uses composite names such as `sirius:/usr/jsmith/memo`, which has two components: the host name `sirius` and the file name (path) `/usr/jsmith/memo`. The `mail` program uses composite names such as `jsmith@altair`, which has two components: the user name `jsmith` and the host name `altair`.

Each application defines its own composition rule for names, parses the composite names, and resolves composite names. Composition rules often differ from one application to another.

Without FNS, the user must remember which applications permit composite naming and which do not. For example, the composite name `sirius:/tmp/saleslist` is accepted by the `rcp` command, but not by the `cp` command.

Without FNS, the user must also remember the different composition rules used among different applications. Applications that support composite names on their own can use only a small and specific set of naming systems, and must be changed whenever a new type of naming system is added.

Incorporating a uniform policy for composite naming into the computing platform permits any application to support composite names in a uniform way. The application passes *one* name to *one* interface.

FNS Policy Principles

The following principles were used to arrive at FNS policies:

- *When it is natural to name other objects relative to a certain object, that object should provide a naming context.* For example, because it is natural to want to name various things relative to a user, a user object should be a naming context.
- *It should be possible to compose names using common components.* This reduces the number of names that users need to remember and makes it easier for applications and users to construct names based on their knowledge of common components and how they can be logically composed.
- *Names should be intuitive and self-evident.* For example, the FNS name `/user/wong/service/calendar` clearly identifies the calendar service used by Wong. In contrast, the calendar name `wong@deneb` names the host (`deneb`) where the calendar service for Wong is being provided. But to other users, there is no obvious connection between the user's calendar and a host. The host name is extraneous and difficult to discover and remember.
- *Never use two contexts when one context will do.* In the example above, we would like to name a mail address, a calendar, and a file's directory relative to the user `wong`. Sharing contexts and their names make naming more coherent and simplifies administration.

FNS in the Solaris Environment

In the Solaris environment, the FNS implementation currently consists of name services implemented on top of:

- Enterprise-level name services such NIS+, NIS, and/or local files. (See “Solaris Enterprise-Level Naming Services” on page 366, below.)
- File naming, printer naming, and support for other applications. (see Chapter 24.)
- Global-level naming systems using DNS and X.500/LDAP. (See Chapter 25.)

FNS will become increasingly more visible to Solaris users as more applications and systems use FNS.

Solaris Enterprise-Level Naming Services

An *enterprise-level* naming service identifies (names) machines (hosts), users and files within an enterprise-level network. FNS also allows naming of organizational units, geographic sites, and application services. An “enterprise-level” network can be a single Local Area Network (LAN) communicating over cables, infra-red beams, or radio broadcast; or a cluster of two or more LANs linked together by cable or direct phone connections. Within an enterprise-level network, every machine is able to communicate with every other machine without reference to a global naming service such as DNS or X.500/LDAP.

FNS currently supports three enterprise-level naming services:

- NIS+. See “FNS and NIS+ Naming” on page 367, below, “How FNS Policies Relate to NIS+” on page 397, and “Advanced FNS and NIS+ Issues” on page 413.
- NIS. See “FNS and NIS Naming” on page 367, “How FNS Policies Relate to NIS” on page 399, and “Advanced FNS and NIS Issues” on page 415.
- Files. See “FNS and Files-Based Naming” on page 368, “How FNS Policies Relate to Files-Based Naming” on page 400, and “Advanced FNS and File-Based Naming Issues ” on page 418.

See Chapter 22, for administration information regarding FNS and enterprise-level naming services.

FNS and NIS+ Naming

If you are not familiar with NIS+ and its terminology, refer to Part 1 and *Glossary* of this guide. You will find it helpful to be familiar with the structure of a typical NIS+ environment.

NIS+ is the preferred enterprise-wide information service in the Solaris environment. Both NIS and local files can be used along with NIS+. NIS+ allows an enterprise to be divided into hierarchical organizational levels composed of domains and subdomains.

FNS organization units correspond to NIS+ domains and subdomains. There is one `orgunit` context for each domain and subdomain.

FNS federates NIS+, NIS, and local files to support naming policies in the Solaris environment. To do this, FNS provides the XFN interface for performing naming operations on `organization`, `site`, `user`, and `host` objects. It implements these operations using the appropriate programming interface for accessing files, directories, and tables.

Under NIS+, FNS context and attribute data is stored in NIS+ type tables. These tables are stored in NIS+ type directory objects named `ctx_dir`. There is an `ctx_dir` directory object for each NIS+ domain and subdomain, residing at the same level as the domain's `groups_dir` and `org_dir` directory objects. Thus, the directory object `ctx_dir.sales.doc.com` contains FNS tables which store FNS context and attribute data for the `sales.doc.com` domain.

Under NIS+, you use FNS and NIS+ commands to work with the information in FNS tables. Do not edit these tables directly or manipulate them with UNIX commands.

FNS and NIS Naming

NIS is an enterprise-wide information service in the Solaris environment. Local files can be used along with NIS. Under NIS, an enterprise is organized as a single NIS domain.

Each enterprise is a single NIS domain. There is one FNS organizational unit which corresponds to the single NIS domain.

FNS federates NIS and local files to support naming policies in the Solaris environment. To do this, FNS provides the XFN interface for performing naming operations on `organization`, `site`, `user`, and `host` maps. It implements these operations using the appropriate programming interface for accessing files, directories.

Under NIS, FNS context and attribute data are stored in NIS maps. These maps are stored in a `/var/yp/domainname` directory on a NIS server. Under NIS, the super user can use FNS commands to work with the information in FNS maps.

NIS Clients Can Update Contexts With FNS if SKI is Running

If certain conditions are met, any NIS client (machine, process, or user) can use FNS commands such as `fncreate_fs` or `fncreate_printer` to update the client's own contexts. This allows NIS clients to use FNS commands to update applications such as Printer Administrator, CDE Calendar Manager, Admin Tool and others.

For non-super-users to update their own contexts with FNS commands, the following conditions must be met:

- Secure Key_management Infrastructure (SKI) must be available on the NIS master server.
- The `fnssypd` daemon must be running on the NIS master server. This daemon must be started by someone with super user privileges.
- A client user or machine is only allowed to update its own context.
- The client must be authorized to perform the requested update.

FNS and Files-Based Naming

Files refers to the naming files normally found in a machine's `/etc` directory. These machine-based files contain UNIX user and password information, host information, mail aliases, and so forth. They also support Solaris-specific data such as the automount maps.

FNS federates local files to support naming policies in the Solaris environment. To do this, FNS provides the XFN interface for performing naming operations on `organization`, `site`, `user`, and `host` files. It implements these operations using the appropriate programming interface for accessing files, directories.

Under a files-based naming system, FNS context and attribute data is stored in files. These files are stored in a `/var/fn` directory exported from an NFS file server.

Under a files-based naming system, you use FNS commands to work with the information in FNS files.

Global Naming Services

A global naming service identifies (names) those enterprise-level networks around the world that are linked together via phone, satellite, or other communication systems. This world-wide collection of linked networks is known as the "Internet." In addition to naming networks, a global naming service also identifies individual machines and users within a given network.

FNS currently supports two global naming services:

- DNS. See “FNS and DNS” on page 369, below and “Federating Under DNS” on page 448.
- X.500/LDAP. See “FNS and X.500” on page 369, below and “Federating Under X.500/LDAP” on page 449.

Note - You can only federate a global naming service if your enterprise-level name service is NIS+ or NIS. If you are using a files-based name service for your enterprise, you cannot federate either DNS or X.500/LDAP.

See Chapter 25, for administration information regarding FNS and enterprise-level naming services.

FNS and DNS

The Internet Domain Name System (DNS) is a hierarchical collection of name servers that provide the world Internet with host and domain name resolution. FNS uses DNS to name enterprise objects globally.

A domain name is the name DNS uses to identify an enterprise-level network (LAN or WAN). Networks using NIS+ permit creation of subdomains within the parent domain, and DNS can identify such subdomains.

Names can be constructed for any enterprise that is accessible on the Internet; consequently, names can also be constructed for objects exported by these enterprises. For more information about FNS and DNS, see “Federating Under DNS” on page 448.

FNS and X.500

X.500 is a global directory service. Its components cooperate to manage information about objects in a worldwide scope. Such objects include countries, organizations, people, and machines. FNS federates X.500 to enable global access to enterprise name services. You can choose to use one of two APIs to access the X.500 global directory service:

- XDS/XOM API
- LDAP (Lightweight Directory Access Protocol) API.

See “Federating Under X.500/LDAP” on page 449 for information on federating X.500.

FNS and Applications

FNS supports:

- Solaris NFS file service (see “FNS File Naming” on page 370, below).
- Printer naming (see “FNS Printer Naming” on page 370).
- Other applications (see “FNS Application Support” on page 370).

FNS File Naming

FNS-based file naming integrates FNS naming into the Solaris file service. FNS-based file naming enables files to be named relative to users, hosts, sites, and organizations, using the FNS policies shared with other non-file applications.

FNS-based file naming gives clients a common view of the global and enterprise-wide file namespaces. Solaris applications that access the file system will, without modification, have access to the file namespaces supported by FNS.

FNS Printer Naming

FNS-based printer naming provides the basic naming support for the unbundled SunSoft Print Client (SSPC). FNS-based printer naming enables printers to be named relative to users, hosts, sites, and organizations, using the FNS policies shared with other non-printing-related applications.

FNS-based printer naming gives clients a common view of the global and enterprise-wide printer namespaces and allows centralized administration of the printer namespaces.

FNS Application Support

Applications that are aware of FNS can expect the namespace to be arranged according to the FNS policies, and applications that bind names in the FNS namespace are expected to follow these policies.

Applications use FNS three ways:

- *Applications can be direct clients of the FNS interface and policies.* Application-level utilities such as the file system, the printing service, and the desktop tools (calendar manager, file manager) are examples of clients that use the FNS interface directly.

- *Applications can use FNS through existing interfaces.* A significant proportion of FNS use is through existing application programming interfaces. For example, consider a UNIX application that obtains a file name that it later supplies to the UNIX `open()` function. With FNS support for resolution of file names, the application need not be aware that the strings it deals with are composite names rather than the traditional local path names. Many applications can thereby support the use of composite names without modification.
- *Systems can export the FNS interface.* Naming systems, such as DNS and X.500, and naming systems embedded in other services, like the file system and printing service, are examples of naming systems that export the FNS interface.

Administering FNS

FNS System administration varies according to the underlying naming service:

- *NIS+.* Under NIS+, FNS system administration tasks can only be performed by those with authorization to do so. The usual method of granting system administration privileges is to create an NIS+ group and assign that group the necessary privileges for that domain. Any member of the group can then perform system administration functions.
- *NIS.* Under NIS, FNS administration tasks must be performed by `root` on the NIS master server.
- *Files.* Under a files-based naming system, FNS administration tasks must be performed by someone with `root` access to the `/var/fn` directory.

The ability of users to make changes to their own user sub-contexts varies according to the underlying naming service:

- *NIS+.* Under NIS+, a user's context (and associated sub-contexts) are owned by them. When logged in as an NIS+ principle, users who have the appropriate credentials and privileges can make changes to their own context using the `fncreate`, `fnbind`, `fnunbind`, and similar commands.
- *NIS.* Under NIS, users cannot make any changes to any FNS data. Only those with `root` access on the NIS master server can change FNS data.
- *Files.* Under a files-based naming system, users own their own contexts. Standard UNIX access controls apply to FNS files.

Troubleshooting and Error Messages

For troubleshooting common FNS problems and solving them, see “FNS Problems and Solutions” on page 549. FNS error messages are included in Appendix B.

FNS Policies

This chapter describes FNS policies.

- “Introduction to FNS and XFN Policies” on page 374
- “Policies for the Enterprise Namespace” on page 375
- “Enterprise Namespace Identifiers” on page 376
- “Default FNS Enterprise Namespaces” on page 375
- “Organizational Unit Namespace” on page 377
- “Site Namespace” on page 379
- “Host Namespace” on page 379
- “User Namespace” on page 380
- “File Namespace” on page 380
- “Service Namespace” on page 380
- “FNS Reserved Names” on page 381
- “Composite Name Examples” on page 382
- “Structure of the Enterprise Namespace” on page 383
- “Enterprise Root” on page 386
- “Initial Context Bindings for Naming Within the Enterprise” on page 391
- “FNS and Enterprise Level Naming” on page 396
- “Target Client Applications of FNS Policies” on page 400
- “FNS File System Namespace” on page 403
- “The FNS Printer Namespace” on page 405
- “Policies for the Global Namespace” on page 405
- “Federating DNS” on page 406

- “Federating X.500/LDAP” on page 407

Introduction to FNS and XFN Policies

XFN defines policies for naming objects in the federated namespace. The goals of these policies are

- To allow easy and uniform composition of names
- To promote coherence in naming across applications and services
- To provide a simple, yet sufficiently rich, set of policies so that applications need not invent and implement ad hoc policies for specific environments
- To enhance an application’s portability
- To promote cross-platform interoperability in heterogeneous computing environments

What FNS Policies Specify

FNS policies contain all the XFN policies plus extensions for the Solaris environment.

Computing environments now offer worldwide scope and a large range of services. Users expect to have access to services at every level of the computing environment. FNS policies provide a common framework for the three levels of services: global, enterprise, and application.

FNS provides to applications a set of policies on how name services are arranged and used:

- Policies that specify how to federate the enterprise namespace so that it is accessible in the global namespace.
- Policies that specify the names and bindings present in the initial context of every process.
- Name service policies for enterprise objects: organizations, hosts, users, sites, files, and services.
- Policies that define the relationships among the organization, host, user, site, files, and service enterprise objects.
- Policies that specify the syntax of names used to refer to those enterprise objects.

What FNS Policies Do Not Specify

The FNS policies do not specify:

- The actual names used within name services.
- Naming within applications. Application-level naming is left to individual applications or groups of related applications.
- The attributes to use once the object has been named.

Policies for the Enterprise Namespace

FNS policies specify the types and arrangement of namespaces within an enterprise and how such namespaces can be used by applications. For example, which namespaces can be associated with which other namespaces. The FNS policies described here include some extensions to XFN policy. These are explicitly defined with notes.

Default FNS Enterprise Namespaces

The FNS enterprise policies deal with the arrangement of enterprise objects within the namespace. Each enterprise objects has its own namespace.

By default, there are seven FNS enterprise objects and namespaces:

- *Organization* (`orgunit`). Entities such as departments, centers, and divisions. Sites, hosts, users, and services can be named relative to an organization. The XFN term for organization is *organizational unit*. When used in an initial context the identifier `org` can be used as an alias for `orgunit`.
- *Site* (`site`). Physical locations, such as buildings, machines in buildings, and conference rooms within buildings. Sites can have files and services associated with them.
- *Host* (`host`). Machines. Hosts can have files and services associated with them.
- *User* (`user`). Human users. Users can have files and services associated with them.
- *File* (`fs`). Files within a file system.
- *Service* (`service`). Services such as printers, faxes, mail, and electronic calendars.
- *Printer* (`service/printer`). The printer namespace is subordinate to the service namespace.

Figure 21-1 shows how these enterprise namespaces are arranged.

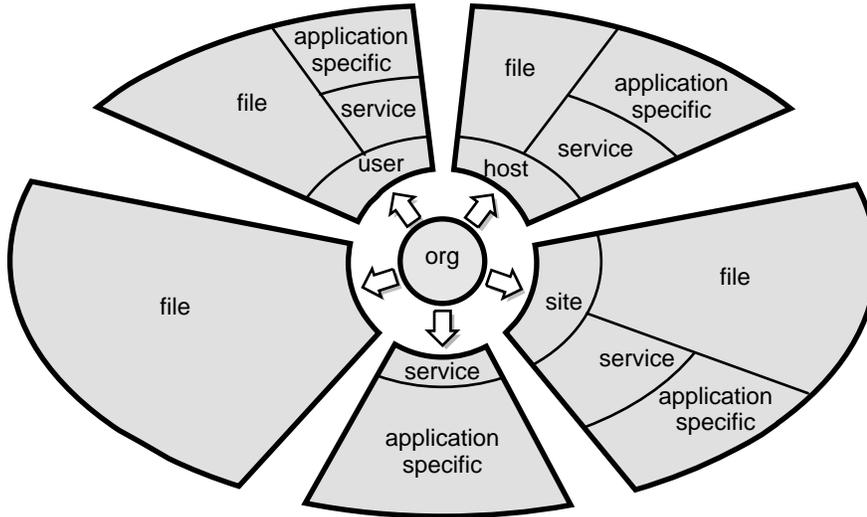


Figure 21-1 What FNS Policies Arrange

Some of these namespaces, such as users and hosts, can appear more than once in a federated namespace.

The policies that apply to these namespaces are summarized in Table 21-2.

Enterprise Namespace Identifiers

Enterprise namespaces are referred to by their atomic names in the federated enterprise namespace.

XFN uses leading underscore (“_”) characters to indicate an enterprise namespace identifier. For example, `_site`. FNS also supports the use of these identifiers without the leading underscore (“_”) character. These names without the underscore are extensions to the XFN policies. The `site` and `printer` contexts are also extensions to the XFN policies. These atomic names are listed in Table 21-1.

TABLE 21-1 Enterprise Namespace Identifiers in the Enterprise

Namespace	XFN Identifiers	FNS Identifiers	Resolves to
Organization	<code>_orgunit</code>	<code>orgunit</code> or <code>org</code>	Context for naming organizational units
Site	<code>_site</code>	<code>site</code>	Context for naming sites
Host	<code>_host</code>	<code>host</code>	Context for naming hosts
User	<code>_user</code>	<code>user</code>	Context for naming users

TABLE 21-1 Enterprise Namespace Identifiers in the Enterprise (continued)

Namespace	XFN Identifiers	FNS Identifiers	Resolves to
File system	_fs	fs	Context for naming files
Service	_service	service	Context for naming services
Printer		printer	Context for naming printers, (subordinate to service namespace)

Note - In XFN terminology, the names with the leading underscore are the *canonical* namespace identifiers. The names without the underscore are namespace identifiers that have been *customized* for the Solaris environment. These customized namespace identifiers, with the addition of `printer`, might not be recognized in non-Solaris environments. The canonical namespace identifiers are always recognized and so are portable to other environments.

Component Separators

The XFN component separator (/) delimits namespace identifiers. For example, composing the namespace identifier `orgunit` with the organizational unit name `west.sales` gives the composite name, `orgunit/west.sales`.

Default FNS Namespaces

There are seven namespaces supplied with FNS:

- *Organization*. (See “Organizational Unit Namespace” on page 377)
- *Site*. (See “Site Namespace” on page 379)
- *Host*. (See “Host Namespace” on page 379)
- *User*. (See “User Namespace” on page 380)
- *File*. (See “File Namespace” on page 380)
- *Service*. (See “Service Namespace” on page 380)
- *Printer*. (See “Service Namespace” on page 380)

Organizational Unit Namespace

The organizational unit namespace provides a hierarchical namespace for naming subunits of an enterprise. Each organizational unit name is bound to an *organizational*

unit context that represents the organizational unit. Organization unit names are identified by the prefixes `org/`, `orgunit/`, or `_orgunit/`. (The shorthand alias `org/` is only used in the initial context, never in the middle of a compound name. See “Initial Context Bindings for Naming Within the Enterprise” on page 391 and “Composite Name Examples” on page 382.)

NIS+ Environment

In an NIS+ environment, organizational units correspond to NIS+ domains and subdomains.

Under NIS+, organization units must map to domains and subdomains. You must have an organizational unit for each NIS+ domain and subdomain. You cannot have “logical” organization units within a domain or subdomain. In other words, you cannot divide an NIS+ domain or subdomain into smaller organization units. Thus, if you have a NIS+ domain `doc.com.` and two subdomains `sales.doc.com.` and `manf.doc.com.`, you must have three FNS organizational units corresponding to those three domains.

Organizational units are named using dot-separated right-to-left compound names, where each atomic element names an organizational unit within a larger unit. For example, the name `org/sales.doc.com.` names an organizational unit `sales` within a larger unit named `doc.com.` In this example, `sales` is an NIS+ subdomain of `doc.com.`

Organizational unit names can be either fully qualified NIS+ domain names or relatively named NIS+ names. Fully qualified names have a terminal dot; relative names do not. Thus, if a terminal dot is present in the organization name, the name is treated as a fully qualified NIS+ domain name. If there is no terminal dot, the organization name is resolved relative to the top of the organizational hierarchy. For example, `orgunit/west.sales.doc.com.` is a fully qualified name identifying the `west` organization unit, and `_orgunit/west.sales` is a relatively qualified name identifying the same subdomain.

NIS Environment

In a NIS environment there is only one organization unit per enterprise which corresponds to the NIS domain. This `orgunit` is named `orgunit/domainname` where *domainname* is the name of the NIS domain. For example, if the NIS domain name is `doc.com`, the organizational unit is `org/doc.com`.

In an NIS environment, you can use an empty string as a shorthand for the organizational unit. Thus, `org//` is equivalent to `org/domainname`.

Files-Based Environment

There is only one FNS organization unit and no subunits when your primary enterprise-level name service is files-based. The only permitted organization unit under files-based naming is `org//`.

Site Namespace

The site namespace provides a geographic namespace for naming objects that are naturally identified with their physical locations. These objects can be, for example, buildings on a campus, machines and printers on a floor, conference rooms in a building and their schedules, and users in contiguous offices. Site names are identified by the prefixes `site/` or `_site/`.

In the Solaris environment, sites are named using compound names, where each atomic part names a site within a larger site. The syntax of site names is dot-separated right-to-left, with components arranged from the most general to the most specific location description. For example, `_site/pine.bldg5` names the Pine conference room in building 5, while `site/bldg7.alameda` identifies building 7 of the Alameda location of some enterprise.

Host Namespace

The host namespace provides a namespace for naming computers. Host names are identified by the prefixes `host/` or `_host/`. For example, `host/deneb` identifies a machine named `deneb`.

Hosts are named in *hostname* contexts. The host context has a flat namespace and contains bindings of host names to *host contexts*. A host context allows you to name objects relative to a machine, such as files and printers found at that host.

In the Solaris environment, host names correspond to Solaris host names. Alias names for a single machine share the same context. For example, if the name `mail_server` is an alias for the machines `deneb` and `altair`, both `deneb` and `altair` will share the contexts created for `mail_server`.

Network resources should only be named relative to hosts as appropriate. In most cases, it is more intuitive to name resources relative to entities such as organizations, users, or sites. Dependence on host names forces the user to remember information that is often obscure and sometimes not very stable. For example, a user's files might move from one host to another because of hardware changes, file space usage, network reconfigurations, and so on. And users often share the same file server, which might lead to confusion if files were named relative to hosts. Yet if the files were named relative to the user, such changes do not affect how the files are named.

There might be a few cases in which the use of host names is appropriate. For example, if a resource is available only on a particular machine and is tied to the existence of that machine, and there is no other logical way to name the resource

relative to other entities, then it might make sense to name the resource relative to the host. Or, in the case of a file system, if the files are being shared by many users it might make sense to name them relative to the machine they are stored on.

User Namespace

The user namespace provides a namespace for naming human users in a computing environment. User names are identified by the prefixes `user/` or `_user/`.

Users are named in *user contexts*. The user context has a single-level namespace and contains bindings of user names to *user contexts*. A user context allows you to name objects relative to a user, such as files, services, or resources associated with the user.

In the Solaris environment, user names correspond to Solaris login IDs. For example, `_user/inga` identifies a user whose login ID is `inga`.

File Namespace

A file namespace (or file system) provides a namespace for naming files. File names are identified by the prefixes `fs/` or `_fs/`. For example the name `fs/etc/motd` identifies the file `motd` which is stored in the `/etc` directory.

The file namespace is described in more detail in “FNS File Naming” on page 370. and file contexts are discussed in “File Contexts Administration” on page 439

Service Namespace

The service namespace provides a namespace for services used by or associated with objects within an enterprise. Examples of such services are electronic calendars, faxes, mail, and printing. Service names are identified by the prefixes `service/` or `_service/`.

In the Solaris environment, the service namespace is hierarchical. Service names are slash-separated (/) left-to-right compound names. An application that uses the service namespace can make use of this hierarchical property to reserve a subtree for that application. For example, the printer service reserves the subtree `printer` in the service namespace.

FNS does not specify how service names or reference types are chosen. These are determined by service providers that share the service namespace. For example, the calendar service uses the name `_service/calendar` in the service context to name the calendar service and what is bound to the name `calendar` is determined by the calendar service.

Service Name and Reference Registration

SunSoft, Inc., maintains a registry of the names bound in the first level of the service namespace. To register a name, send an email request to `fns-register@sun.com`, or write to:

FNS Registration Sun Microsystems, Inc. 2550 Garcia Avenue Mountain View, CA 94043

Please include a brief description of the intended use of the name and a description of the format of the reference that can be bound to that name.

Printer Namespace

The printer namespace provides a namespace for naming printers. The printer namespace is associated with (subordinate to) the service namespace. In other words, printer service and the printer namespace is one of the services in the service namespace. Printer names are identified by the prefixes `service/printer` or `_service/printer`. For example, `service/printer/laser1` identifies the printer named `laser1`.

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, in the name, `org/east.sales/service/printer` the slash between `org` and `east.sales` is a component delimiter as described above and the slash that trails after the last element in the organization name (`sales/`) separates the service namespace from the organizational unit namespace. Thus, `org/west.sales/service/printer` names the printer service of the `west.sales` organization unit.

FNS Reserved Names

FNS reserves for its own use all the atomic names listed in Table 21-1 as namespace identifiers. For example: `_orgunit`, `org`, `_site`, `site`, and so forth. This limitation applies to contexts in which the namespace identifiers can appear, as defined by the arrangement of namespaces in “Structure of the Enterprise Namespace” on page 383. FNS does not otherwise restrict the use of these atomic names in other contexts.

For example, the atomic name `service` is used as a namespace identifier relative to a user name, as in `user/fatima/service/calendar`, to mean the root of user `fatima`'s service namespace. This does not preclude a system from using the name `service` as a user name, as in `user/service`, because FNS specifies that the

context to which the name `user/` is bound is for user names and not for namespace identifiers. In this case, `service` is unambiguously interpreted as a user name. On the other hand, you should not create a directory named `user` because `/user/mikhail` would cause confusion between the user `mikhail` and the file (or subdirectory) `/user/mikhail`.

Composite Name Examples

This section shows examples of names that follow FNS policies. (See Table 21-2 for a summary of these policies.)

The specific choices of organization names, site names, user names, host names, file names, and service names (such as `calendar` and `printer`) are illustrative only; these names are not specified by FNS policy.

Composing Names Relative to Organizations

The namespaces that can be associated with the organization namespace (`_orgunit`, `orgunit`, or `org`) are: `user`, `host`, `service`, `fs`, and `site`.

For example:

- `orgunit/doc.com/site/videoconf.bldg-5` names a conference room `videoconf` located in Building 5 of the site associated with the organization `doc.com`. (You can also use the alias `org` for `orgunit` to form, `org/doc.com/site/videoconf.bldg-5`.)
- `orgunit/doc.com/user/mjones` names a user `mjones` in the organization `doc.com`.
- `orgunit/doc.com/host/smptserver1` names a machine `smptserver1` belonging to the organization `doc.com`.
- `orgunit/doc.com/fs/staff/agenda9604/` names a file `staff/agenda9604` belonging to the organization `doc.com`.
- `orgunit/doc.com/service/calendar` names the `calendar` service for the organization `doc.com`. This might manage the meeting schedules for the organization.

Composing Names Relative to Users

The namespaces that can be associated with the `user` namespace are `service` and `fs`.

- `user/helga/service/calendar` names the `calendar` service of a user named `helga`.

- `user/helga/fs/tasklist` names the file `tasklist` under the home directory of the user `helga`.

Composing Names Relative to Hosts

The namespaces that can be associated with the `hosts` namespace are `service` and `fs`.

- `host/mailhop/service/mailbox` names the mailbox service associated with the machine `mailhop`.
- `host/mailhop/fs/pub/saf/archives.96` names the directory `pub/saf/archives.96` found under the root directory of the file system exported by the machine `mailhop`.

Composing Names Relative to Sites

The namespaces that can be associated with the `sites` namespace are `service` and `fs`.

- `site/bldg-7.alameda/service/printer/speedy` names a printer `speedy` in the `bldg-7.alameda` site.
- `site/alameda/fs/usr/dist` names a file directory `usr/dist` available in the `alameda` site.

Composing Names Relative to Services and Files

No other namespaces can be associated with either the files (`fs`) or services (`service`) namespaces. For example, you cannot compose a name such as `/services/calendar/orgunit/doc.com`. In other words, you cannot compose a compound name relative to either the files or the service namespace. You can, of course, compose a file or service name relative to some other namespace such as `/user/esperanza/service/calendar`.

Structure of the Enterprise Namespace

FNS policies define the structure of the enterprise namespace. The purpose of this structure is to allow easy and uniform composition of names. This enterprise namespace structure has two main rules:

- Objects with narrower scopes are named relative to objects with wider scopes.
- Namespace identifiers are used to denote the transition from one namespace to the next.

Table 21-2 is a summary of FNS policies for arranging the enterprise namespace. Figure 21-2 shows an example of a namespace layout that follows these FNS policies.

TABLE 21-2 Policies for the Federated Enterprise Namespace

Namespace Identifiers	Subordinate Namespaces	Parent Context	Namespace Organization	Syntax
orgunit _orgunit org	Site, user, host, file system, service	Enterprise root	Hierarchical	Dot-separated right-to-left
site _site	Service, file system	Enterprise root, organizational unit	Hierarchical	Dot-separated right-to-left
user _user	Service, file system	Enterprise root, organizational unit	Flat	Solaris login name
host _host	Service, file system	Enterprise root, organizational unit	Flat	Solaris host name
service _service	Application specific	Enterprise root, organizational unit, site, user, host	Hierarchical	/ separated left-to-right
fs _fs	None	Enterprise root, organizational unit, site, user, host	Hierarchical	/ separated, left-to-right
printer	None	Service	Hierarchical	/ separated left-to-right

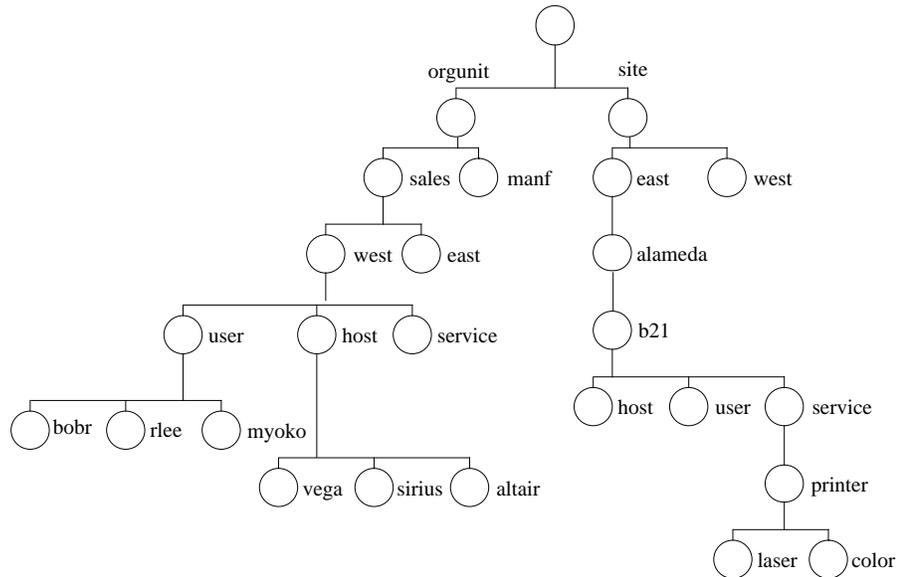


Figure 21-2 Example of an Enterprise Namespace

The namespace of an enterprise is structured around a hierarchy of organizational units. Names of sites, hosts, users, files, and services can be named relative to names of organizational units by composing the organizational unit name with the appropriate namespace identifier and object name.

In Figure 21-2, a user `myoko` in the west division of the sales organization of an enterprise is named using the name `orgunit/west.sales/user/myoko`.

Note the use of the namespace identifier `user` to denote the transition from the `orgunit` namespace to the `user` namespace. In a similar fashion (with the use of appropriate namespace identifiers), names of files and services can also be named relative to names of sites, users, or hosts. Names of sites can be named relative to organizational unit names.

The goal of easy and uniform composability of names is met using this structure. For example, once you know the name for an organizational unit within an enterprise (for example, `orgunit/west`), you can name a user relative to it by composing it with the `user` namespace identifier and the user's login name to yield a name such as `orgunit/west/user/josepha`.

To name a file in this user's file system, you can use a name like `orgunit/west/user/josepha/fs/notes`.

Enterprise Root

The root context of an enterprise, is a context for naming objects found at the root level of the enterprise namespace. Enterprise roots are bound in the global namespace.

There are two ways of naming the enterprise root:

- `.../rootdomain`.
- `org//`.

Using Three Dots to Identify the Enterprise Root

You can use `.../rootdomain/` to identify an enterprise root where:

- The initial three dots (`...`) are an atomic name indicating the global context (see “Policies for the Global Namespace” on page 405 for a description of the global context).
- `rootdomain/` is the enterprise root domain. For example, `doc.com/`.

Thus, `.../doc.com/` identifies the enterprise root of a company whose root domain is `doc.com`. In this example, the context for naming sites associated with the enterprise root is `.../doc.com/site/` such as `.../doc.com/site/alameda` or `.../doc.com/site/alameda.bldg5`.

Note - You can only use the `.../rootdomain` format if you have set up the global binding in DNS.

Using `org//` to Identify the Enterprise Root

You can use `org//` to identify an enterprise root. In essence, `org//` is an alias or functional equivalent for `.../domainname/`. When using `org//`, the double slashes identifies the root enterprise context and namespaces associated with it.

For example, `org//site/alameda` names the Alameda site associated with the enterprise root.

In contrast, `org/` or `orgunit/` (with a single slash) points to an organizational context which is not necessarily named relative to the enterprise root. For example, `org/sales/site/alameda`.

Enterprise Root Subordinate Contexts

The following objects can be named relative to the enterprise root:

- Organizational units in that enterprise
- Sites in the top organizational unit of the enterprise (an extension to XFN policies)
- Users in the top organizational unit of the enterprise
- Hosts in the top organizational unit of the enterprise
- Services for the top organizational unit of the enterprise
- File service for the top organizational unit of the enterprise

These objects are named by composing the namespace identifier of the target object's namespace with the name of the target object.

Enterprise Root and Organizational Subunits

Organizational subunits can be named relative to the enterprise root.

Given an organization root name, you can compose names for its subordinate organizational unit contexts by using one of the namespace identifiers, `orgunit` or `_orgunit`.

For example, if `.../doc.com` is the name of an enterprise, the root of the context for naming organizational units is `.../doc.com/orgunit/`, and organizational unit names look like `.../doc.com/orgunit/sales` and `.../doc.com/orgunit/west.sales`. Or, you could achieve the same result with `org//orgunit/sales`.

The following objects can be named relative to an organizational unit name:

- Sites for that organizational unit (an extension to the XFN policies)
- Hosts in that organizational unit
- Users in that organizational unit
- Services for that organization unit
- File service for that organizational unit

For example, the name `...doc.com/orgunit/sales/service/calendar`, identifies the calendar service of the `sales` organizational unit. (See “Organizational Unit Namespace” on page 377 and “Composing Names Relative to Organizations ” on page 382 for a more detailed description of naming objects relative to organization units.)

Enterprise Root and Sites

Sites are an extension to the XFN policies.

Sites can be named relative to

- The enterprise root
- An organizational unit

Sites named relative to the enterprise root are the same as sites named relative to the top organizational unit. Given an organization name, you can compose a name for its site context by using one of the namespace identifiers, `site` or `_site`. For example, if the enterprise root is `../doc.com` the context for naming sites relative to the enterprise root is `../doc.com/site`. Sites would have names like `../doc.com/site/alameda`.

The following objects can be named relative to a site name:

- Services at the site, such as the site schedule or calendar, printers, and faxes
- The file service available at the site

These objects are named by composing the site name with the namespace identifier of the target object's namespace and the name of the target object. For example, the name `site/Clark.bldg-5/service/calendar` names the calendar service of the conference room `Clark.bldg-5` and is obtained by composing the site name `site/Clark.bldg-5` with the service name `service/calendar`. (See "Composing Names Relative to Sites" on page 388 for a more detailed description of naming objects relative to sites.)

Enterprise Root and Users

Users can be named relative to

- An organizational unit
- The enterprise root

Users named relative to the enterprise root are the same as users named relative to the top organizational unit. Given an organization name, you can compose a name for its `username` context by using one of the namespace identifiers, `user` or `_user`. Thus, if `orgunit/east.sales` names an organization, then `orgunit/east.sales/user/hirokani` names a user `hirokani` in the `east.sales` organizational unit.

The following objects can be named relative to a user name:

- Services associated with the user
- The user's files

These objects are named by composing the user's name with the namespace identifier of the target object's namespace and the name of the target object. For example, the name `user/sophia/service/calendar` names the calendar for the user `sophia`. (See "User Namespace" on page 380 and "Enterprise Root and Users" on page 388 for more information on the user namespace and naming objects relative to users.)

Enterprise Root and Hosts

Hosts can be named relative to

- An organizational unit
- The enterprise root

Hosts named relative to the enterprise root are the same as hosts named relative to the top organizational unit. Given an organization name, you can compose a name for its `hostname` context by appending one of the namespace identifiers, `host` or `_host`. Thus if `orgunit/west.sales` names an organization, the name `org/west.sales/host/altair` names a machine `altair` in the `west.sales` organizational unit.

The following objects can be named relative to a host name:

- Services associated with the host
- Files exported by the host

These objects are named by composing the host name with the namespace identifier of the target object's namespace and the name of the target object. For example, the name `host/sirius/fs/release` names the file directory `release` being exported by the machine `sirius`. (See "Host Namespace" on page 379 and "Composing Names Relative to Hosts" on page 383 for more information on the host namespace and naming objects relative to hosts.)

Enterprise Root and Services

A service can be named relative to

- An organizational unit
- The enterprise root
- A user
- A host
- A site

Services named relative to the enterprise root are the same as services named relative to the top organizational unit.

A service context is named by using the namespace identifiers `service` or `_service`, relative to the organization, site, user, or host with which it is associated. For example, if `orgunit/corp.finance` names an organizational unit, then `orgunit/corp.finance/service/calendar` names the `calendar` service of the organizational unit `corp.finance`. (See "Service Namespace" on page 380 and "Composing Names Relative to Services and Files" on page 383 for more information on the user namespace and naming objects relative to users.)

FNS does not restrict the types of bindings in the service namespace. Applications can create contexts of a type other than service contexts and bind them in the service namespace.

FNS supports the creation of *generic* contexts in the service context. A generic context is similar to a service context except that a generic context has an application-determined reference type. All other properties of a generic context are the same as a service context.

For example, a company named World Intrinsic Designs Corp (WIDC), reserves the name `extcomm` in the service namespace to refer to a generic context for adding bindings related to its external communications line of products. The context bound to `extcomm` is a generic context, with reference type `WIDC_comm`. The only difference between this context and a service context is that this context has a different reference type.

Service names should be registered with SunSoft, Inc., as directed in “Service Name and Reference Registration” on page 381.

Enterprise Root and Files

A file namespace can be named relative to

- The enterprise root
- An organizational unit
- A user
- A host
- A site

Files named relative to the enterprise root are the same as files named relative to the top organizational unit. A file context is named by using the namespace identifiers `fs` or `_fs`, relative to the organization, site, user, or host with which it is associated. For example, if `orgunit/accountspayable.finance` names an organizational unit, then the name `user/jsmith/fs/report96.doc` names her file `report96.doc`. The file service of the user defaults to her home directory, as specified in the NIS+ `passwd` table. (See “File Namespace” on page 380 for more information on the user namespace.)

There can be no other type of context subordinate to a file system.

Enterprise Root and Printers

The printer context is an extension of XFN policies.

A printer namespace can be named in the service context. A printer context is named by using the namespace identifier, `printer`, in the service context relative to

- An organizational unit
- A user
- A host

- A site

For example, if `org/east.sales` names an organizational unit, then `org/eastsales/service/printer` names the printer service of the organizational unit `east.sales`. Thus, an individual printer named `lp1` would be identified as: `org/east.sales/service/printer/lp1`.

There can be no other type of context subordinate to a printer.

Initial Context Bindings for Naming Within the Enterprise

An initial context is the starting place from which client users, hosts, and applications can (eventually) name any object in the enterprise namespace.

Figure 21-3 shows the same naming system as the one shown in Figure 21-2, except that the initial context bindings are shaded and shown in italics. These initial contexts are shown from the point of view of the user, host, or application asking for a name to be resolved.

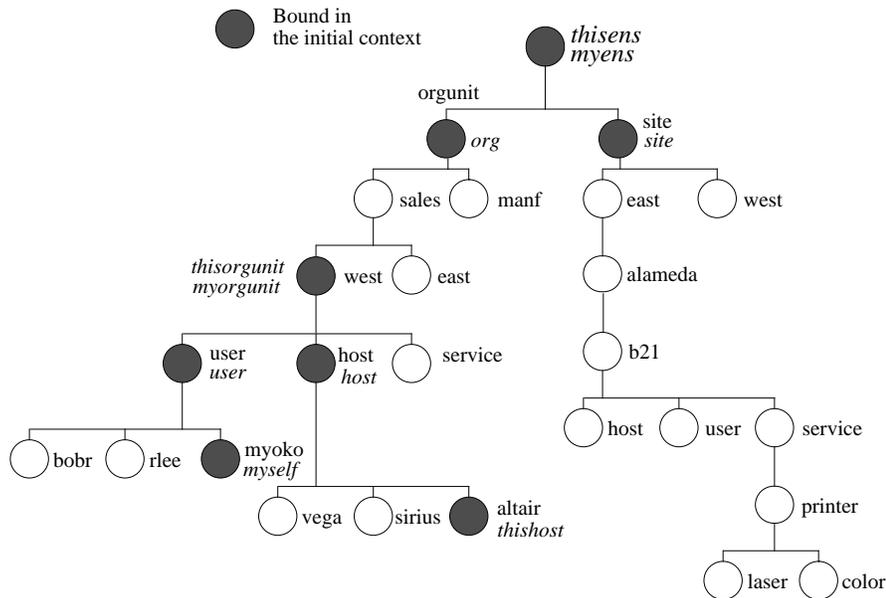


Figure 21-3 Example of Enterprise Bindings in the Initial Context

XFN provides an *initial context function*, `fn_ctx_handle_from_initial()`, that allows a client to obtain an initial context object as a starting point for name resolution. The initial context has a flat namespace for namespace identifiers. The bindings of these initial context identifiers are summarized in Table 21-3 and are described in more detail in subsequent sections. Not all of these names need to

appear in all initial contexts. For example, when a program is invoked by the superuser, only the host and client-related bindings appears in the initial context, none of the user-related bindings appear.

TABLE 21-3 Initial Context Bindings for Naming Within the Enterprise

Namespace	
Identifier	Binding
myself, _myself, thisuser	The context of the user who is trying to resolve a name.
myens, _myens	The enterprise root of the user who is trying to resolve a name.
myorgunit, _myorgunit	The user's primary organizational unit context. For example, in an NIS+ environment, the primary organizational unit is the user's NIS+ home domain.
thishost, _thishost	The context of the host that is trying to resolve a name.
thisens, _thisens	The enterprise root of the host that is trying to resolve a name.
thisorgunit, _thisorgunit	The host's primary organizational unit context. For example, in an NIS+ environment, the primary organizational unit is the host's NIS+ home domain
user, _user	The context in which users in the same organizational unit as the host are named
host, _host	The context in which hosts in the same organizational unit as the host are named
org, orgunit, _orgunit	The root context of the organizational unit namespace in the host's enterprise. For example, in an NIS+ environment, this corresponds to the NIS+ root domain
site, _site	The root context of the site namespace at the top organizational unit if the site namespace has been configured

In XFN terminology, names with a leading underscore prefix are called the *canonical* namespace identifiers. The names without the leading underscore, with the additions of `org` and `thisuser`, are Solaris customizations. Solaris customized namespace

identifiers are not guaranteed to be recognized in other, non-Solaris environments. The canonical namespace identifiers are always recognized and therefore portable to other environments.

Note - The current implementations of FNS does not support the addition or modification of names and bindings in the initial context.

Initial context bindings fall into three basic categories:

- User-related bindings (see “User-related Bindings” on page 393)
- Host-related bindings (see “Host-related Bindings” on page 394)
- “Shorthand” bindings (see ““Shorthand” Bindings” on page 395)

User-related Bindings

FNS assumes that there is a user associated with a process when the XFN initial context function is invoked. This association is based on the effective user ID (*euuid*) of the process. Although the association of user to process can change during the life of the process, the original context handle does not change.

FNS defines the following bindings in the initial context that are related to the user requesting name resolution.

myself

The namespace identifier *myself* (or either synonym *_myself* or *thisuser*) in the initial context resolves to the user context of whomever is making the request. For example, if a process owned by the user *jsmith* requests name resolution, the name:

- *myself* resolves in the initial context to *jsmith*'s user context
- *myself/fs/.cshrc* names the file *.cshrc* of *jsmith*

myorgunit

FNS assumes that each user is affiliated with an organizational unit of an enterprise. A user can be affiliated with multiple organizational units, but there must be one organizational unit that is primary, perhaps by its position in the organizational namespace or by the user's role in the organization.

- *NIS+*. In an *NIS+* namespace, this organizational unit corresponds to the user's home domain (which could be a subdomain).
- *NIS*. In a *NIS* namespace, there is only one enterprise-level organizational unit which corresponds to the user's domain.
- *Files*. In a files-based namespace, there is only one organizational unit, *org//* which maps to *myorgunit*.

The namespace identifier `myorgunit` (or `_myorgunit`) resolves in the initial context to the context of the primary organizational unit of the user making the request. For example, if the user making the request is `jsmith`, and `jsmith`'s home domain is `east.sales`, then `myorgunit` resolves in the initial context to the organizational unit context for `east.sales`, and the name `myorgunit/service/calendar` resolves to the calendar service of `east.sales`.

myens

FNS assumes that there is an association of a user to an enterprise. This corresponds to the namespace that holds `myorgunit`.

The namespace identifier `myens` (and `_myens`) resolves in the initial context to the enterprise root of the enterprise to which the user making the request belongs. For example, if `jsmith` is making the request, and `jsmith`'s NIS+ home domain is `east.sales`, which in turn is in the NIS+ hierarchy with the root domain name of `doc.com`. The name `myens/orgunit/` resolves to the top organizational unit of `doc.com`.

Note - Be careful about set-user-ID programs when using user-related composite names, such as `myorgunit` or `myself/service`, because these bindings depend on the effective user ID of a process. For programs that set-user-ID to root to access system resources on behalf of the caller, it is usually a good idea to call `seteuid(getuid())` before calling `fn_ctx_handle_from_initial()`.

Host-related Bindings

A process is running on a particular host when the XFN initial context function is invoked. FNS defines the following bindings in the initial context that are related to the host the process is running on.

thishost

The namespace identifier `thishost` (or `_thishost`) is bound to the host context of the host running the process. For example, if the process is running on the machine `cygnus`, `thishost` is bound to the host context of `cygnus`, and the name `thishost/service/calendar` refers to the calendar service of the machine `cygnus`.

thisorgunit

FNS assumes that a host is associated with an organizational unit. A host can be associated with multiple organizational units, but there must be one that is primary. In an NIS+ namespace, this organizational unit corresponds to the host's home domain.

The namespace identifier `thisorgunit` (or `_thisorgunit`) resolves to the primary organizational unit of the host running the process. For example, if that host is the machine `cygnus`, and `cygnus`'s NIS+ home domain is `west.sales`, then `thisorgunit` resolves to the organizational unit context for `west.sales` and the name `thisorgunit/service/fax` refers to the fax service of the organizational unit `west.sales`.

thisens

FNS assumes that there is an association of a host to an enterprise. This corresponds to the namespace that holds `thisorgunit`.

The namespace identifier `thisens` (or `_thisens`) resolves to the enterprise root of the host running the process. For example, under NIS+, if the host's home domain is `sales.doc.com`, then the name `thisens/site/` resolves to the root of the site namespace of `doc.com`.

“Shorthand” Bindings

FNS defines the following “shorthand” bindings in the initial context to enable the use of shorter names to refer to objects in certain commonly referenced namespaces.

user

The namespace identifier `user` (or `_user`) is bound in the initial context to the username context in organizational unit of the host running the process. This allows other users in the same organizational unit to be named from this context.

From the initial context, the names `user` and `thisorgunit/user` resolve to the same context. For example, if the host running the process is a machine `altair` and `altair` is in the `east.sales` organizational unit, the name `user/medici` names the user `medici` in `east.sales`.

host

The namespace identifier `host` (or `_host`) is bound in the initial context to the hostname context organizational unit of the host running the process. This allows other hosts in the same organizational unit to be named from this context.

From the initial context, the names `host` and `thisorgunit/host` resolve to the same context. For example, if the host running the process is a machine named `sirius` and it is in the `east.sales` organizational unit, the name `host/sirius` names the machine `sirius` in the organizational unit `east.sales`.

org

The namespace identifier `org` (or `orgunit`, `_orgunit`) is bound in the initial context to the root context of the organization of the enterprise to which the host running the process belongs.

From the initial context, the names `org` and `thisens/orgunit` resolve to the same context. For example, if the host running the process is the machine `aldebaran` and `aldebaran` is in the enterprise `doc.com.`, the name `org/east.sales` names the organizational unit `east.sales` in `doc.com.`

site

The namespace identifier `site` (or `_site`) is bound in the initial context to the root of the site naming system of the top organizational unit of the enterprise to which the host running the process belongs.

From the initial context, the names `site` and `thisens/site` resolve to the same context. For example, if the host running the process is the machine `aldebaran` and `aldebaran` is in the enterprise `doc.com.`, the name `site/pine.bldg-5` names a conference room, `pine` in building 5 of `doc.com.`

FNS and Enterprise Level Naming

FNS provides a method for federating multiple naming services under a single, simple interface for basic naming operations. FNS is designed to work with three enterprise-level name services:

- *NIS+*. See “How FNS Policies Relate to NIS+” on page 397, below and “FNS and NIS+ Naming” on page 367
- *NIS*. See “How FNS Policies Relate to NIS” on page 399 and “FNS and NIS Naming” on page 367
- *Files*. “How FNS Policies Relate to Files-Based Naming” on page 400 and “FNS and Files-Based Naming” on page 368

FNS is also designed to work with applications such as printer and calendar service as described in “Target Client Applications of FNS Policies” on page 400.

How FNS Policies Relate to NIS+

See “FNS and NIS+ Naming” on page 367 for overview and background information relating to FNS and NIS+. If you are not familiar with NIS+ and its terminology, refer to Part 1 and *Glossary* of this guide. You will find it helpful to be familiar with the structure of a typical NIS+ environment.

FNS stores bindings for enterprise objects in FNS tables which are located in domain-level `org_dir` NIS+ directories on NIS+ servers. FNS tables are similar to NIS+ tables. These FNS tables store bindings for the following enterprise namespaces:

- *Organization* namespaces as described in “NIS+ Domains and FNS Organizational Units” on page 397.
- *Hosts* namespaces as described in “NIS+ Hosts and FNS Hosts ” on page 398.
 - *Users* namespace as described in “NIS+ Users and FNS Users” on page 398
- *Sites* namespace which allows you to name geographical sites relative to the organization, hosts, and users.
- *Services* namespace which allows you to name services such a printer service and calendar service relative to the organization, hosts, and users.

NIS+ Domains and FNS Organizational Units

FNS names organization, user, and host enterprise objects within NIS+ which is the preferred Solaris enterprise name service. An NIS+ domain is comprised of logical collections of users and machines and information about them, arranged to reflect some form of hierarchical organizational structure within an enterprise.

FNS is implemented on NIS+ by mapping NIS+ domains to FNS organizations. An organizational unit name corresponds to a NIS+ domain name and is identified using either the fully qualified form of its NIS+ domain name, or its NIS+ domain name relative to the NIS+ root. The top of the FNS organizational namespace is mapped to the NIS+ root domain and is accessed using the name `org/` from the initial context.

In NIS+, users and hosts have a notion of a *home domain*. A host or user’s home domain is the NIS+ domain that maintains information associated with them. A user or host’s home domain can be determined directly using its NIS+ *principal name*. An NIS+ principal name is composed of the atomic user (login) name or the atomic host name and the name of the NIS+ home domain. For example, the user `sekou` with home domain `doc.com`. has an NIS+ principal name `sekou.doc.com` and the machine name `vega` has an NIS+ principal name `vega.doc.com`.

A user’s NIS+ home domain corresponds to the user’s FNS organizational unit. Similarly, a host’s home domain corresponds to its FNS organizational unit.

Trailing Dot in Organization Names

The trailing dot in an organization name indicates that the name is a fully qualified NIS+ domain name. Without the trailing dot, the organization name is an NIS+ domain name to be resolved relative to the NIS+ root domain.

For example, if the NIS+ root domain is `doc.com.`, with a subdomain `sales.doc.com.`, the following pairs of names refer to the same organization:

TABLE 21-4 Example of Relative and Fully Qualified Organization Names Under NIS+

Relative Name	Fully Qualified Name
<code>org/</code>	<code>org/doc.com.</code>
<code>org/sales</code>	<code>org/sales.doc.com.</code>

The name `org/manf.` (with trailing dot) would not be found, because there is no NIS+ domain with just the `manf.` name.

NIS+ Hosts and FNS Hosts

Hosts in the NIS+ namespace are found in the `hosts.org_dir` table of the host's home domain. Hosts in an FNS organization correspond to the hosts in the `hosts.org_dir` table of the corresponding NIS+ domain. FNS provides a context for each host in the `hosts` table.

NIS+ Users and FNS Users

Users in the NIS+ namespace are listed in the `passwd.org_dir` table of the user's home domain. Users in an FNS organization correspond to the users in the `passwd.org_dir` table of the corresponding NIS+ domain. FNS provides a context for each user in the `passwd` table.

NIS+ Security and FNS

The FNS `fncreate` command creates FNS tables and directories in the NIS+ hierarchy associated with the domain of the host on which the command is run. In order to run `fncreate`, you must be an authenticated NIS+ principle with credentials authorizing you to Read, Create, Modify, and Destroy NIS+ objects in that domain. You will be the *owner* of the FNS tables created by `fncreate`. One way to obtain this authorization is to be a member of the NIS+ group that has administrator privileges in the domain.

The `NIS_GROUP` environment variable should be set to name of the NIS+ administration group for the domain prior to running `fncreate`. You can specify whether or not individual users can make changes to FNS data that relates to them. See Chapter 6, for a description of NIS+ security.

How FNS Policies Relate to NIS

See “FNS and NIS Naming” on page 367 for overview and background information relating to FNS and NIS.

FNS provides the XFN interface for performing basic naming and attributes operations using NIS as the naming service.

FNS stores bindings for enterprise objects in FNS maps which are located in a `/var/yp/domainname` directory on the NIS master server (and NIS slave servers, if any). FNS maps are similar in structure and function to NIS maps. These NIS maps store bindings for the following enterprise namespaces:

- *Organization* which provides a namespace for naming objects relative to an entire enterprise. When NIS is the underlying naming service, there is a single organizational unit context that corresponds to the NIS domain. This organization unit context is identified in FNS by the NIS domain name or an empty name which defaults to the machines NIS domain name.
- *Hosts* namespace which correspond to the `hosts.byname` map of the NIS domain. FNS provides a context for each host in the `hosts.byname` map.
- *Users* namespace which correspond to the `passwd.byname` map. FNS provides a context for each user in the `passwd.byname` map of the domain.
- *Sites* namespace which allows you to name geographical sites relative to the organization, hosts, and users.
- *Services* namespace which allows you to name services such as a printer service and calendar service relative to the organization, hosts, and users.

FNS provides contexts which allow other objects to be named relative to these five namespaces.

The FNS `fncreate` command creates the FNS maps in the `/var/yp/domainname` directory of a NIS master server. This can be the same machine that is master server for the NIS naming service, or it can be a different machine that functions as an FNS master server. (If there are slave servers, NIS pushes the FNS maps to them as part of its normal operation.) To run `fncreate`, you must be a privileged user on the server that will host the FNS maps. Individual users cannot make changes to FNS data.

How FNS Policies Relate to Files-Based Naming

See “FNS and Files-Based Naming” on page 368 for overview and background information relating to FNS and files.

FNS provides the XFN interface for performing basic naming and attribute operations using local files as the naming service.

FNS stores bindings for enterprise objects in files which are located in a `/var/fn` directory which is normally NFS mounted on each machine. These FNS files store bindings for the following enterprise namespaces:

- *Organization* which provides a namespace for naming objects relative to the entire enterprise. When local files are the underlying naming service, there is a single organizational unit context that represents the entire system. This organization unit context is always identified in FNS as `org//`.
- *Hosts* namespace which correspond to the `/etc/hosts` file. FNS provides a context for each host in the `/etc/hosts` file.
- *Users* namespace which correspond to the `/etc/passwd` file. FNS provides a context for each user in the `/etc/passwd` file.
- *Sites* namespace which allows you to name geographical sites relative to the organization, hosts, and users.
- *Services* namespace which allows you to name services such as a printer service and calendar service relative to the organization, hosts, and users.

FNS provides contexts which allow other objects to be named relative to these five namespaces.

The FNS `fncreate` command creates the FNS files in the `/var/fn` directory of the machine on which the command is run. To run `fncreate`, you must have super-user privileges on that machine. Based on UNIX user IDs, individual users are allowed to modify their own contexts, bindings, and attributes using FNS commands.

Target Client Applications of FNS Policies

One goal of the FNS policies is to maintain coherence across the most commonly used tools, including the file system, the DeskSet tools, such as Calendar Manager, Print Tool, File Manager, and Mail Tool, and services that support these tools, such as RPC, email, and print subsystems.

Note - Some of these examples are not currently implemented in the Solaris environment. They are listed here to illustrate how FNS can be used.

- *Calendars*. Instead of using names of the form `username@hostname` to access someone’s calendar, in most cases you simply supply a site, organization, or user’s name. You should also be able to use composite names to name calendars. For

example, when federated under FNS, names of the following form are acceptable to calendar manager:

- bernadette
- user/bernadette
- site/pine.bldg-5 (calendar for Pine conference room)
- org/sales (calendar for the sales organization)

- *Printing.* Instead of naming a specific printer by its name, you should be able to name a printer relative to a user, site, or organization. For example:
 - ilych (ilych's default printer)
 - org/sales (an organization's default printer)
 - site/pine.bldg-5 (printer in the Pine conference room)

- *File access.* You should be able to use composite names to name file systems and files. The automounter should use FNS to make resolution of composite names possible. For example, you should be able to use a file name like `/xfn/user/baruch/fs/.cshrc` to reference the `.cshrc` file for user `baruch`.

- *RPC.* Instead of addressing services by their host name, program, and version numbers, you should be able to name the service using a composite name. For example, you should be able to name an RPC service relative to a user or an organization such as: `user/hatori/service/rpc`.

- *Mail* – Similarly, composite names can be used to name mail destinations. You should be able to use names such as the following:
 - angus
 - user/angus
 - org/mlist (an organization's mailing list)
 - site/pine.bldg-5 (mailbox of the conference room coordinator)

- *Other desktop applications* – You should be able to pass composite names to other desktop applications such as spreadsheets, document preparation tools, fax tools, and so on. Some of these applications attach their own namespace to the service namespace, thus becoming part of the FNS federation.

Example Applications: Calendar Service

This is a description of how one application, a calendar service, could be modified to use FNS policies. This example illustrates how FNS composite names might be presented to and accepted from users.

The DeskSet's calendar service is typical of client-server applications. Calendar servers run on some set of machines and maintain the users' calendars. Calendar

Manager (`cm`) runs on the desktop and contacts the appropriate server to obtain the calendars of interest.

The calendar service could benefit from FNS using a simple registry/lookup model as follows:

- *Binding* – Upon startup, the server registers the calendars that it manages by binding a reference containing its own ONC+ RPC address (*host, program, version*) to the composite name for each calendar it manages, such as `user/jsmith/service/calendar`.
- *Lookup* – When using `cm`, the user specifies another user's calendar simply by entering the user's name (for example, `hirokani`) or selecting it from a list of names previously entered. Given the user name `hirokani`, `cm` composes the composite name `user/hirokani/service/calendar` and uses this to look up the RPC address that it needs to communicate with the server that manages that calendar.

In the previous example, we used the name “calendar” to denote a calendar binding. The developers of the calendar service should register the name “calendar” with the FNS administrator, much as RPC programs are registered with the RPC administrator. Refer to “Service Name and Reference Registration” on page 381.

Note - The name “calendar” used here is an example. FNS policy does not specify the names of specific services.

The calendar service could take further advantage of FNS policy by allowing calendars to be associated with sites, organizations, and hosts, while still naming them in a uniform way. For example, by allowing calendars to be associated with a conference room (a site), the service can be used to “multibrowse” the conference room's calendar as well as a set of user calendars to find an available time for a meeting in that room. Similarly, calendars can be associated with organizations for group meetings and hosts for keeping maintenance schedules.

The `cm` calendar manager could simplify what the user needs to specify by following some simple steps.

1. `cm` uses a tool for accepting composite names from the user and constructing the name of the object whose calendar is of interest.

The object is the name of a user, a site, a host, or an organization. For example, the user might enter the name `kuanda` and the calendar manager generates the composite name `user/kuanda`. This tool could be shared amongst a group of DeskSet applications.

2. `cm` uses the XFN interface to compose this name with the suffix `/service/calendar` to obtain the name of the calendar.
3. This calendar name is then resolved relative to the process's initial context.

Continuing with the example, this results in the resolution of the name `user/kuanda/service/calendar`. Similarly, if the user enters the name of a

site, pine.bldg-5, cm generates the name
site/pine.bldg-5/service/calendar for resolution.

Other services such as printing and mail could take advantage of the FNS policies in a similar way.

FNS File System Namespace

Files may be named relative to users, hosts, organizations, and sites in FNS by appending the `fs` enterprise namespace identifier to the name of the object, and following this with the name of the file. For example, the file `draft96` in the sales division's `budget` directory might be named `org/sales/fs/budget/draft96`.

The initial context is located under `/xfn` in the file system's root directory. Thus a user might view the file by typing

```
% more /xfn/org/sales/fs/budget/draft96
```

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 21-4.

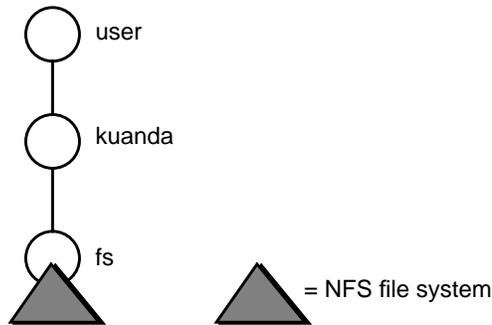


Figure 21-4 NFS File System—Simple Case

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 21-5 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.

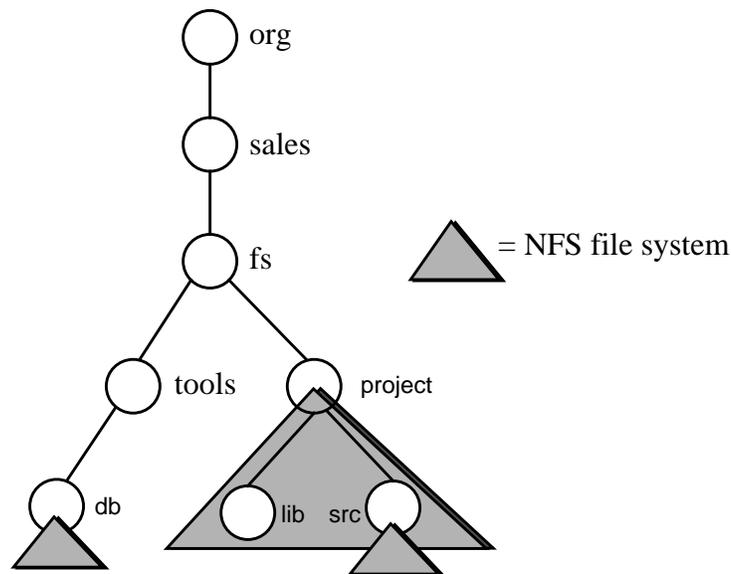


Figure 21-5 NFS File System—Multiple Servers

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, suppose the file system of the sales organization is composed of multiple NFS file systems. The following `ls` command shows only two file systems that have been visited recently and are currently mounted:

```
% ls /xfn/org/sales/fs
customers products
```

To see the entire listing, use the `fnlist` command.:

```
% fnlist org/sales/fs
Listing 'org/sales/fs':
products
goals
customers
incentives
```

The FNS Printer Namespace

The `printer` context is not part of the XFN policies. It is provided in FNS in order to store printer bindings.

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 without client side administration.

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/doc.com./service/printer
```

The name of a printer for a host, `deneb`, with a printer context might look like this:

```
host/deneb/service/printer/laser
```

Policies for the Global Namespace

Global name services have worldwide scope. This section describes the policies for naming objects that use global naming systems.

In regard to naming, an enterprise links to the federated global namespace by binding the root of the enterprise in the global namespace. This enables applications and users outside the enterprise to name objects within that enterprise. For example,

a user within an enterprise can give out the global name of a file to a colleague in another enterprise to use.

DNS and X.500 contexts provide global-level name service for naming enterprises. FNS provides support for both DNS and X.500 contexts. Without FNS, DNS and X.500 allow outside access to only limited portions of the enterprise namespace. FNS enables outside access to the entire enterprise namespace including services such as calendar manager.

Initial Context Bindings for Global Naming

The atomic name “. . .” (three dots) appears in the initial context of every FNS client. The atomic name “/ . . .” is bound to a context from which global names can be resolved.

TABLE 21-5 Initial Context Bindings for Global Naming

Atomic Name	Binding
. . .	Global context for resolving DNS or X.500 names
/ . . .	Synonym for three dots

Global names can be either fully qualified Internet domain names, or X.500 distinguished names.

- Internet domain names appear in the syntax specified by Internet RFC 1035. For example, . . . / doc . com. (See “Federating DNS” on page 406.)
- X.500 names appear in the syntax determined by the X/Open DCE Directory. For example, . . . / c=us/o=doc. (See “Federating X.500/LDAP” on page 407.)

The names “. . .” and “/ . . .” are equivalent when resolved in the initial context. For example, the names / . . . / c=us/o=doc and . . . / c=us/o=doc resolve in the initial context to the same object.

Federating DNS

Any fully qualified DNS name can be used in the global context. When a DNS name is encountered in the global namespace, it is resolved using the resolver library. The resolver library is the DNS name-resolution mechanism. A DNS name typically resolves to an Internet host address or to DNS domain records. When the global

context detects a DNS name, the name is passed to the DNS resolver for resolution. The result is converted into an XFN reference structure and returned to the requester.

The contents of DNS domains can be listed. However, the listing operations might be limited by practical considerations such as connectivity and security on the Internet. For example, listing the global root of the DNS domain is generally not supported by the root DNS servers. Most entities below the root, however, do support the list operation.

DNS hosts and domains are distinguished from each other by the presence or absence of name service (NS) resource records associated with DNS resource names.

- *DNS domain names.* If an NS record exists for a resource name, then that name is considered to be the name of a domain, and the returned reference is of type `inet_domain`.
- *DNS host names.* If no NS record exists for a resource name, then that name is considered to be the name of a host, and the returned reference is of type `inet_host`.

DNS can be used to federate other naming systems by functioning as a non-terminal naming system.

For example, an enterprise naming system can be bound to `doc.com` in DNS such that the FNS name `.../doc.com/` refers to the root of that enterprise's FNS namespace.

The enterprise naming system is bound to a DNS domain by adding the appropriate text (TXT) records to the DNS map for that domain. When the FNS name for that domain includes a trailing slash (/), the TXT resource records are used to construct a reference to the enterprise naming system.

For general information about DNS, see the `in.named` Man page or the DNS chapters in *Solaris Naming Setup and Configuration Guide*.

Federating X.500/LDAP

X.500 is a global directory service. It stores information and provides the capability to look up information by name as well as to browse and search for information.

X.500 information is held in a directory information base (DIB). Entries in the DIB are arranged in a tree structure. Each entry is a named object and comprises a defined set of attributes. Each attribute has a defined attribute type and one or more values.

An entry is unambiguously identified by a *distinguished name* that is the concatenation of selected attributes from each entry in the tree along a path leading from the root down to the named entry. For example, using the DIB shown in Figure 21-6,

```
c=us/o=doc
```

is a distinguished name of the `doc` organization in the U.S. Users of the X.500 directory can interrogate and modify the entries and attributes in the DIB.

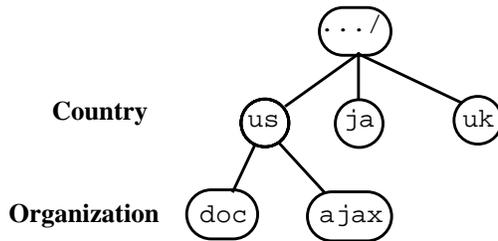


Figure 21-6 Example of an X.500 Directory Information Base

FNS federates X.500 by supplying the necessary support to permit namespaces to appear to be seamlessly attached below the global X.500 namespace.

For example, FNS facilitates linking the enterprise naming system for the `doc` organization below X.500. Starting from the initial context, an FNS name to identify the `sales` organizational unit of the `doc` organization might be

```
.../c=us/o=doc/orgunit/sales
```

The name within the enterprise is simply concatenated onto the global X.500 name. (Note that FNS names use the name `'...'` in the initial context to indicate that a global name follows.)

Name resolution of FNS names takes place as follows. When an X.500 name is encountered in the global namespace, it is resolved using the X.500 name-resolution mechanism. One of three outcomes is possible:

- The full name resolves to an X.500 entry. This indicates that the entry is held in X.500. The requested FNS operation is then performed on that entry.
- A prefix of the full name resolves to an X.500 entry. This indicates that the remainder of the name belongs to a subordinate naming system.

The next naming system pointer (NNSP) to the subordinate naming system is examined to return the XFN reference. Name resolution then continues in the subordinate naming system.

- An error is reported.

X.500 entries can be examined and modified using FNS operations (subject to access controls). However, it is not currently possible to list the subordinate entries under the root of the X.500 namespace by using FNS.

FNS and Enterprise Name Services

This chapter discusses administration matters relating to FNS and enterprise level naming services.

- “FNS and Enterprise-Level Naming Services” on page 409
- “Choosing an Enterprise-Level Name Service” on page 410
- “FNS and Naming Service Consistency” on page 410
- “Selecting a Naming Service” on page 412
- “Default Naming Service” on page 412
- “When NIS+ and NIS Coexist” on page 413
- “Advanced FNS and NIS+ Issues” on page 413
- “Advanced FNS and NIS Issues” on page 415
- “Advanced FNS and File-Based Naming Issues ” on page 418

FNS and Enterprise-Level Naming Services

Enterprise-level naming services are used to name objects within an enterprise. FNS currently supports three enterprise-level naming services:

- NIS+. (See also “FNS and NIS+ Naming” on page 367 and “How FNS Policies Relate to NIS+” on page 397.
- NIS. See “FNS and NIS Naming” on page 367 and “How FNS Policies Relate to NIS” on page 399.

- Local files. See “FNS and Files-Based Naming” on page 368 and “How FNS Policies Relate to Files-Based Naming” on page 400.

Choosing an Enterprise-Level Name Service

When you initially set up and configure your FNS namespace with the `fncreate` command as described in *Solaris Naming Setup and Configuration Guide*, the correct default name service is automatically selected for each machine.

If you later change a machine’s primary enterprise-level name service, you should run the `fnselect` command on that machine. See “Selecting a Naming Service” on page 412 for details.

FNS and Naming Service Consistency

As a system administrator one of your tasks is to maintain consistency between FNS and the underlying naming service by ensuring that the contents of FNS contexts and the files, maps, or tables of the underlying naming service correspond.

When you initially set up and configure your FNS namespace with the `fncreate` command as described in *Solaris Naming Setup and Configuration Guide*, `fncreate` ensures that FNS contexts are correctly created and are consistent with the underlying naming service data. After the FNS contexts have been set up, this correspondence needs to be maintained as users, hosts, printers, and so forth are added to and removed from the system. The following sections describe how to maintain FNS and name service consistency.

FNS and Solstice AdminSuite

If you have the Solstice AdminSuite product, you can use it to add, change, or delete user and host information in the underlying name service. This is a recommended method because the AdminSuite tools update the corresponding FNS namespace automatically.

Checking Naming Inconsistencies

When updates to FNS or the primary name service are made independent of the Solstice AdminSuite product, the resulting inconsistencies are resolved by the use of the FNS tool, `fncheck`. The `fncheck` command checks for inconsistencies between the FNS `hostname` and `user` contexts, and:

- *NIS+*. The `NIS+ hosts.org_dir` and `passwd.org_dir` system tables.
- *NIS*. The `NIS hosts.byname` and `passwd.byname` maps.
- *Files*. The `etc/hosts` and `etc/passwd` files.

The `fncheck` command lists those host and user names that are in the FNS namespace but not in the name service data, and those host and user names that are in the name service data but not in the FNS namespace.

The command syntax is:

```
fncheck [-r][-s][-u][-t hostname|username][domain_name]
```

TABLE 22-1 `fncheck` Command Options

Option	Description
<i>domain</i>	Apply the command to an NIS+ domain other than the one in which you are running the command.
<code>-t</code>	Specifies the type of context to check. Allowed types are <code>hostname</code> or <code>username</code> .
<code>-s</code>	Lists host or user names from the namespace dataset that are not in the FNS namespace
<code>-r</code>	Lists host or user names from the FNS namespace that do not have entries in the corresponding namespace dataset
<code>-u</code>	Updates the FNS namespace based on information in the relevant namespace dataset

The `-t` option is used to specify the contexts to check (host or user). If you omit the `-t` option, both the `hostname` and `username` contexts are checked.

When the `-r` option is used with the `-u` option, items that appear only in the FNS context are removed from the FNS context. When the `-s` option is used with the `-u` option, items that appear only in the namespace dataset 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 namespace data.

Selecting a Naming Service

When FNS constructs the bindings in the initial context for a machine, it does so on the basis of a particular naming service.

You can choose which name service FNS is to use with the `fselect` command. The name service setting you specify with `fselect` affects the entire machine, all applications running on that machine, and all users logged in to that machine.

Only root can run `fselect`. The command syntax is:

```
fselect [-D] [namesvc]
```

TABLE 22-2 `fselect` Command Options

Option	Description
<code>namesvc</code>	The naming service you want to select. Must be one of: <code>default</code> , <code>nisplus</code> , <code>nis</code> , or <code>files</code> .
<code>-D</code>	Display the naming service used to generate the FNS initial context.

For example, to select NIS+ as a machine's name service:

```
#fselect nisplus
```

For example, to select the default as a machine's name service and print the name of the service used to generate the FNS initial context:

```
#fselect -D default
```

Default Naming Service

If you do not designate a naming service with `fselect`, FNS uses the default naming service. The default naming service is determined by FNS based on the name service that the machine is using. If the machine is an NIS+ client, FNS uses NIS+ as the name service. If the machine is a NIS client, FNS uses NIS. If the machine is neither an NIS+ nor a NIS client, FNS uses `/etc` files as the machine's default name service.

When NIS+ and NIS Coexist

In rare cases you may need to access both NIS+ and NIS-based contexts. For example, you might have a NIS server running that is itself an NIS+ client. In this situation, you use the `fnselect` command to select the enterprise-level naming service that you want to work with.

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.

Migrating to NIS+ From NIS or Files-Based Naming

See:

- “Migrating From NIS to NIS+” on page 417.
- “Migrating From Files-Based Naming to NIS or NIS+” on page 419.

Mapping FNS Contexts to NIS+ Objects

FNS contexts are stored as NIS+ objects. All contexts associated with an organization are stored under the FNS `ctx_dir` directory of the associated NIS+ domain. The `ctx_dir` directory resides at the same level as the `org_dir` directory of the same domain. In other words, when running in conjunction with FNS, for every NIS+ domain or subdomain, there are corresponding `org_dir`, `groups_dir` and `ctx_dir` directory objects.

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, `nislsls`, can be used to list the NIS+ objects used by FNS. For example, the following commands list the contents of the NIS+ domain directory and its `ctx_dir` subdirectory.

```
# nisl doc.com.
doc.com.:
manf
sales
groups_dir
org_dir
ctx_dir
```

```
# nisl ctx_dir.doc.com.
ctx_dir.DOC.COM.:
fns
fns_user
fns_host
fns_host_alto
fns_host_mladd
fns_host_elvira
fns_user_jjones
fns_user_jsmith
fns_user_aw
```

Use the `niscat` command to list the contents of the `fns_hosts` table.

```
# niscat fns_host.ctx_dir
altair *BINARY* *BINARY*
cygnus *BINARY* *BINARY*
centauri *BINARY* *BINARY*
```

Checking Access Control

Use `niscat -o` to see the access control of a context. 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 fns_host.ctx_dir
Object Name      : fns_host
Owner            : alto.doc.com.
Group            :          admin.doc.com.
Domain           :          ctx_dir.doc.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         :
```

(continued)

```

[0] Name      : atomicname
   Attributes : (SEARCHABLE, TEXTUAL DATA, CASE INSENSITIVE)
   Access Rights : r-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-

```

```

# niscat -o "[atomicname=altair],fns_host.ctx_dir"
Object Name : fns_host
Owner      : altair.doc.com.
Group      :      admin.doc.com.
Domain     : ctx_dir.doc.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

```

(See “The `niscat` Command ” on page 226 for additional information on the `niscat` command.)

To change the access control or ownership of a particular context, use the commands:

- `nischown`
- `nischmod`
- `nischgrp`

Give either the binding entry or the bindings table as an argument, depending on the object the operation is to affect.

Advanced FNS and NIS Issues

This section provides specific information on the relationship between NIS and FNS.

NIS and FNS Maps and Makefiles

FNS uses six new maps which are stored in `/var/yp/domainname` directories on the NIS master and slave servers:

- `fns_host.ctx` which stores host attributes and subcontext data. When this is first created, it derives its information from the `hosts.byname` map.
- `fns_host.ctx` which stores user attributes and subcontext data. When this is first created, it derives its information from the `passwd.byname` map.
- `fns_org.ctx` which stores organization attributes and subcontext data.
- `fns_host.attr` which stores host attributes for attribute based searches.
- `fns_user.attr` which stores user attributes for attribute based searches.
- `fns_org.attr` which stores organization attributes for attribute based searches.

Service and file context information for hosts, users, and the organization are stored in the respective `fns_host.ctx`, `fns_user.ctx`, and `fns_org.ctx` maps. Printer context information is stored in the same maps as other service context information. However, the older `printers.conf.byname` map is still supported.

Sites are subcontexts of the organization and site context information is stored in the `fns_org.ctx` map.

Note - These FNS maps should not be edited directly. You modify or work with these maps by running the appropriate FNS commands such as `fncreate`, `fndestroy`, `fnbind`, `fnunbind`, `fnrename`, `fnattr`, `fnlookup`, and `fnlist`. These commands must be run on the NIS master server. You cannot run them on slave servers or client machines.

The FNS map files are placed in the `/var/yp/domainname` directory. The NIS Makefile in `/var/yp` is modified to be aware of the FNS Makefile in `/etc/fn/domainname`.

Large FNS Contexts

NIS has a 64k limit on the number of entries a NIS map can contain. If only service and printer contexts are created for each object (host or user), that limit will be reached when the number of users or hosts exceeds 7k. If additional contexts are created for hosts or users, as is usually the case, the upper 64k limit will be reached with far fewer hosts or users.

FNS solves this problem by automatically creating new maps once an old map has reached its maximum size. Each new map is identified by adding a numeric suffix to the map's name. For example, when a second `fns_user.ctx` map is created it is given the name `fns_user_0.ctx`. If a third map became necessary it would be given the name `fns_user_1.ctx`. As additional maps are created, the number is incremented each time.

Printer Backward Compatibility

In Solaris release 2.5, FNS support for printer naming under NIS was provided for the organization context with a map named `printers.conf.byname`. In the current Solaris release, organization context printer support is maintained in the `fns_org.ctx` map. That is, the `fncreate_printer` command now modifies the `fns_org.ctx` map and not the `printers.conf.byname` map.

Migrating From NIS to NIS+

The `fncopy` command handles the FNS-related aspects of changing your underlying enterprise-level naming service from NIS to NIS+. This command copies and converts NIS-based FNS contexts to NIS+ based contexts.

The command syntax is:

```
fncopy [-i oldsvc -o newsvc] [-f filename] oldctx newctx
```

TABLE 22-3 `fncopy` Command Options

Option	Description
<code>-i oldsvc</code>	Source naming service. Only <code>nis</code> or <code>files</code> may be specified.
<code>-o newsvc</code>	Target naming service. Only <code>nisplus</code> or <code>nis</code> may be specified.
<code>-f filename</code>	Name of file listing the FNS contexts to be copied
<code>oldctx</code>	Old FNS context to be copied
<code>newctx</code>	Target new FNS context

For example, to copy the contexts listed in the file `/etc/sales_users` from the `doc.com` domain of a NIS-based naming service to the `sales.doc.com` domain of an NIS+ naming service, you would enter:

```
fncopy -i nis -o nisplus -f /etc/sales_users org/sales.doc.com/user
```

Advanced FNS and File-Based Naming Issues

This section provides specific information on the relationship between files-based naming and FNS.

FNS Files

FNS uses new files which are stored in `/var/fn` directories on each machine. (While a `/var/fn` directory is normally stored on each machine, you can mount and export a central `/var/fn` directory via NFS.)

The new FNS files are:

- `fns_host.ctx` which stores host attributes and subcontext data. When this is first created, it derives its information from the `/etc/hosts` file.
- `fns_user.ctx` which stores user attributes and subcontext data. When this is first created, it derives its information from the `/etc/passwd` file.
- `fns_org.ctx` which stores organization attributes and subcontext data.
- `fns_host.attr` which stores host attributes for attribute based searches.
- `fns_user.attr` which stores user attributes for attribute based searches.
- `fns_org.attr` which stores organization attributes for attribute based searches.
- Users' sub-context and attribute information is stored in separate `/var/fn` files that are owned by each user. This allows users to modify their own data with FNS commands. These user-specific files are named `fns_user_username.ctx` where *username* is the login ID of the individual user.

Service and file context information for hosts, users, and the organization are stored in the respective `fns_host.ctx`, `fns_user.ctx`, and `fns_org.ctx` files. Printer context information is stored in the same files as other service context information.

Sites are subcontexts of the organization and site context information is stored in the `fns_org.ctx` file.

Note - These FNS files should not be edited directly. You modify or work with these files by running the appropriate FNS commands such as `fncreate`, `fndestroy`, `fnbind`, `fnunbind`, `fnrename`, `fnattr`, `fnlookup`, and `fnlist`. When you run these commands as root, they affect the context that they are applied to such as hosts, site, and organization unit. When you run these commands as a user, they affect only your own user sub-contexts.

Migrating From Files-Based Naming to NIS or NIS+

The `fnccopy` command handles the FNS-related aspects of changing your underlying enterprise-level naming service from files to NIS or NIS+. This command copies and converts files-based FNS contexts to NIS or NIS+ based contexts.

The command syntax is:

```
fnccopy [-i oldsvc -o newsvc] [-f filename] oldctx newctx
```

For example, to copy the contexts listed in the file `/etc/host_list` to the `doc.com` domain of an NIS+ naming service, you would enter:

```
fnccopy -i files -o nisplus -f /etc/host_list //doc.com/host
```

Printer Backward Compatibility

In Solaris release 2.5, FNS support for printer naming for files was provided for the organization context with a file named `printers.conf.byname`. In the current Solaris release, organization context printer support is maintained in the `fns_org.ctx` map. That is, the `fncreate_printer` command now modifies the `fns_org.ctx` map and not the `printers.conf.byname` map.

Enterprise Level Contexts

This chapter describes how to individually create, and administer existing enterprise-level contexts.

- “Creating Enterprise Level Contexts” on page 422
- “All Hosts Context” on page 424
- “Single Host Context” on page 425
- “Host Aliases” on page 425
- “All-Users Context” on page 425
- “Single User Context” on page 426
- “Service Context” on page 426
- “Printer Context” on page 427
- “Generic Context” on page 427
- “Site Context” on page 428
- “File Context” on page 429
- “Namespace Identifier Context” on page 429
- “Administering Enterprise Level Contexts” on page 430
- “Displaying the Binding” on page 430
- “Listing the Context” on page 431
- “Binding a Composite Name to a Reference” on page 433
- “Removing a Composite Name” on page 436
- “Renaming an Existing Binding” on page 436
- “Destroying a Context” on page 437

Creating Enterprise Level Contexts

FNS contexts are created using the `fncreate` command. This section describes how to create FNS contexts individually rather than for the entire organization as described in *Solaris Naming Setup and Configuration Guide*. The `fncreate` command creates a context of the specified type and binds it to the given composite name. It also creates subcontexts for the context.

The `fncreate` command has the following syntax,

```
fncreate -t context_type [-f input_file] [-o][-r reference_type][-s][-v] [-D] composite_name
```

TABLE 23-1 `fncreate` Command Options

Option	Description
<code>--t <i>context</i></code>	Specifies the type of context to create. The <i>context</i> operator can be one of <code>org</code> , <code>hostname</code> , <code>username</code> , <code>host</code> , <code>user</code> , <code>service</code> , <code>site</code> , <code>nsid</code> , <code>generic</code> , or <code>fs</code> .
<code>-f</code>	Creates a context for every host or user listed in <i>input_file</i> . This option can only be used with the <code>-t username</code> or <code>-t 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.
<code>-o</code>	Creates only the context specified. Without the <code>-o</code> option, subcontexts are created according to the FNS policies.
<code>-r</code>	Specifies the <i>reference_type</i> of the generic context being created. It can only be used with the <code>-t generic</code> option.
<code>-s</code>	Creates new contexts for composite names already in use. Otherwise, no new contexts are created for names already bound.
<code>-D</code>	Displays information about the NIS+ object associated with a context each time a context is created. This option is useful for debugging.
<code>-v</code>	Displays information about the creation as each context is created.

Note - If you specify the `-o` option when creating an organization context, the associated `host`, `user`, and `service` contexts are still created but they are 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`.

Creating an Organization Context

Use the `org` type to create an organization context. The composite name must be one of the following depending on the primary naming service:

- *NIS+*. The name of an existing NIS+ domain (or subdomain). An NIS+ domain is an NIS+ directory object with an `org_dir` subdirectory. Populated host and `passwd` tables for the domain must exist in the domain's `org_dir` subdirectory.
- *NIS*. The name of the NIS domain. Associated host and `passwd` maps must also exist.
- */etc files*. There is only the `org//` organization context when using `/etc` files.

Organization Context NIS+ Example

Assume the root NIS+ domain is `doc.com` and there is a subdomain `sales.doc.com`. To create a `sales` organization context to correspond to the `sales` subdomain, you would enter the following command:

```
fncreate -t org org/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.doc.com`.

Because this example used only the `-t` option without the `-o` option, it 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. In effect, that is the same as running the following commands:

```
fncreate -t hostname org/sales/host/  
fncreate -t username org/sales/user/  
fncreate -t service org/sales/service/
```

If, instead, you ran `fncreate -o -t org` the `org` context is created and the `hostname`, `username`, and `service` contexts are also 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 subsequently manipulate `host` and `user` contexts, the `NIS_GROUP` environment variable must have been set accordingly at the time `fncreate` is executed. For example, assuming a C-Shell, to set `NIS_GROUP` to `fns_admins.doc.com`:

```
rootmaster# setenv NIS_GROUP fns_admins.doc.com
```

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 machine 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 host org/sales/host/hname
```

Where *hname* is the name of each machine 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 machine 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 single 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/antares/
```

creates a context for the host named `antares` and effectively runs the commands

```
fncreate -t service org/sales/host/antares/service/  
fncreate -t fs org/sales/host/antares/fs/
```

The `host` context and its subcontexts are owned by the machine. In the above example, the machine `antares`, with NIS+ principal name `antares.sales.doc.com`, owns the contexts:

- `org/sales/host/antares/`
- `org/sales/host/capsule/service/`
- `org/sales/host/capsule/fs.`

The `hostname` context (`org/sales/host` in the above example) to which the machine belongs must already exist. The machine 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 `hostname` context are bound to the reference of that host context.

All-Users Context

The `username` type creates a `username` context in which individual 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
```

Where *uname* represents the various user names that appear 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. Individual user contexts and their subcontexts are owned by the users 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 the `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.doc.com`.

The username context (`org/sales/user` in the above example) 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 the `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 telephone directory, 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. The service namespace can be partitioned 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 `plotter` context.

```
# fncreate -t service org/sales/service/plotter
```

Names such as `org/sales/service/plotter/speedy` and `org/sales/service/plotter/color` 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.

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 a 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 running 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/alameda
# fncreate -t site org/sales/site/alameda.bldg-5
```

create a site context `alameda` and a site subcontext `alameda.bldg-5` for it.

Note - Because these terminal atomic names are not namespace identifiers, no additional bindings 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 file context) for a user or a host. For example, the command

```
# fncreate -t fs org/sales/user/petrova/fs/
```

creates the `fs` context for user `petrova`. Because the terminal atomic name is a namespace identifier, `fncreate` also adds a binding for `org/sales/user/petrova/_fs/` that is bound to the reference of `org/sales/user/petrova/fs/`.

The `fs` context of a user is the user's home directory as it is stored in the NIS+ `passwd.org_dir` table. The `fs` 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 for users and hosts. See "File Contexts Administration" on page 439, 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/alameda.bldg-5/
```

creates the `nsid` context for the site `alameda.bldg-5` 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/alameda.bldg-5/service/
```

to create the `service` context for `alameda.bldg-5`.

The `nsid` context created is owned by the administrator who ran the `fncreate` command.

Administering Enterprise Level Contexts

A number of tools are provided for examining and managing FNS contexts. The commands and their syntax are shown in the sections that follow.

Displaying the Binding

`fnlookup` displays the binding of the given composite name.

```
fnlookup [-v][-L] composite_name
```

TABLE 23-2 `fnlookup` Command Options

Option	Description
<code>-v</code>	Displays the binding in more detail
<code>-L</code>	Displays the reference to which the XFN link is bound

For example, to show the binding for the user `darwin` in detail, you would enter:

```
# fnlookup -v user/darwin/  
Reference type: onc_fn_user  
Address type: onc_fn_nisplus  
length: 52  
context type: user  
representation: normal  
version: 0  
internal name: fns_user_darwin.ctx_dir.sales.doc.com.
```

Suppose `user/Charles.Darwin` is linked to `user/darwin`. The first command in the following example shows what `user/Charles.Darwin` is bound to (an XFN link). The second command follows the XFN link, `user/darwin`, and shows what `user/darwin` is bound to (the user context).

```
# fnlookup user/Charles.Darwin
Reference type: fn_link_ref
Address type: fn_link_addr
  Link name: user/darwin
# fnlookup -L user/Charles.Darwin
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.

```
fnlist [-lv] [name]
```

TABLE 23-3 `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, to display the bindings under the `user` context:

```
# fnlist user/
Listing 'user/':
jjones
julio
chaim
James.Jones
```

If no `name` is given, the command lists the contents of the initial context.

```
# fnlist
Listing '':
_myorgunit
...
_myself
thishost
myself
_organit
_x500
_host
```

(continued)

```

_thisens
myens
thisens
org
orgunit
_dns
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: julio
Reference type: onc_fn_user
Address type: onc_fn_nisplus
context type: user
name: chaim
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: julio
Reference type: onc_fn_user
Address type: onc_fn_nisplus
length: 52
context type: user
representation: normal
version: 0

```

```

internal name: fns_user_julio.ctx_dir.sales.doc.com.
name: chaim
Reference type: onc_fn_user
Address type: onc_fn_nisplus
  length: 52
  context type: user
  representation: normal
  version: 0
internal name: fns_user_chaim.ctx_dir.sales.doc.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.doc.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. (See below.)
- The second usage allows the user to bind a reference constructed using arguments in the command line to a name. (See “Constructing a Reference on the Command Line ” on page 434.)

Binding an Existing Name to a New Name

The syntax of `fnbind` for binding existing names to new names is:

```
fnbind [-s][-v][-L] oldname newname
```

TABLE 23-4 `fnbind` Command Options (Binding Names)

Option	Description
<i>oldname</i>	The existing composite name
<i>newname</i>	The new name to which you are binding the old name
<code>-s</code>	Supersedes any existing binding of the original composite name
<code>-v</code>	Prints out the reference used for the binding
<code>-L</code>	Creates an XFN link using <i>name</i> and binding it to <i>new_name</i>

For example, to bind the name `user/julio/service/printer` to the reference of `myorgunit/service/printer` you would enter:

```
# fnbind myorgunit/service/printer user/julio/service/printer
```

If the given *newname* is already bound, `fnbind -s` must be used or the operation will fail. In the above example, if `user/julio/service/printer` is already bound, the `-s` option must be used to overwrite the existing binding with that of `myorgunit/service/printer` as shown below:

```
# fnbind -s myorgunit/service/printer user/julio/service/printer
```

The `-v` option prints out the reference used for the binding.

```
# fnbind -v myorgunit/service/printer user/julio/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, to create a link from `user/julio/service/printer` to `myorgunit/service/printer` you would enter:

```
# fnbind -sL myorgunit/service/printer user/julio/service/printer
```

Constructing a Reference on the Command Line

The syntax of `fnbind` for building a reference on the command line is:

```
fnbind -r [-s] [-v] newname [-O | -U] reftype {[-O | -U] | addresstype [-c|-x] addresscontents}+
```

TABLE 23-5 `fnbind` Command Options (Reference Construction)

Option	Description
<i>newname</i>	The new name for which you are constructing a reference
<i>reftype</i>	The type of reference you are creating. Unless the <code>-O</code> or <code>-U</code> options are used, <code>FN_ID_STRING</code> is used as the identifier for <i>reftype</i> .
<i>addresstype</i>	The type of address you are creating. Unless the <code>-O</code> or <code>-U</code> options are used, <code>FN_ID_STRING</code> is used as the identifier for <i>addresstype</i> .
<i>addresscontents</i>	The address of the reference you are creating. Unless the <code>-c</code> or <code>-x</code> options are used, the address is stored as an XDR-encoded string.
<code>-s</code>	Supersedes any existing binding of the original composite name
<code>-v</code>	Prints out the reference used for the binding
<code>-c</code>	Stores address contents without XDR encoding
<code>-x</code>	Interprets address contents as a hexadecimal input string and store it as is
<code>-r</code>	Creates a reference with a specified type and binds the reference to a name specified on the command line
<code>-O</code>	Interprets and stores type string as ASN.1 dot-separated integer list
<code>-U</code>	Interprets and stores type string as a DCE UUID

For example, to bind the name `thisorgunit/service/calendar` to the address contents of `staff@cygnus` with a reference type of `onc_calendar` and an address type `onc_cal_str` you would enter:

```
# fnbind -r thisorgunit/service/calendar onc_calendar onc_cal_str staff@cygnus
```

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 normal, readable characters, 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.1 dot-separated integer list string. If the `-U` option is given, the identifier format is `FN_ID_DCE_UUID`, a DCE UUID in string form.

Note - For more information on ASN.1, see *ISO 8824: 1990, Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*. For more information on DCE UUID see *X/Open Preliminary Specification, October 1993, X/Open DCE: Remote Procedure Call (ISBN: 1-872630-95-2)*.

For example, to bind to the name `thisorgunit/service/nx` a reference with a hexadecimal string as the address contents and OIDs as reference and address types, you would enter:

```
# 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, to remove the binding associated with the name `user/jjones/service/printer/color`, you would enter:

```
# fnunbind user/jjones/service/printer/color
```

Renaming an Existing Binding

The `fnrename` command renames an existing binding.

For example, to rename the binding of `clndr` to `calendar`, in the context named by `user/jjones/service/` you would enter:

```
# fnunbind user/jjones/service/printer/color
```

Destroying a Context

`fndestroy` removes the given composite name from the namespace and destroys the context named by the composite name.

For example, to unbind the name `user/jones/` from the namespace and destroys the context named by `user/jjones/` you would enter:

```
# fndestroy user/jjones/
```

If the composite name identifies a context to be removed, the command fails if the context contains subcontexts.

Administering File Contexts

This chapter describes how to administer application specific contexts.

- “File Contexts Administration” on page 439
- “Creating a File Context With `fncreate_fs` ” on page 440
- “Advanced Input Formats” on page 443
- “Backward Compatibility Input Format ” on page 444

File Contexts Administration

File contexts may be:

- *Created* using the `fncreate_fs` command (see “Creating a File Context With `fncreate_fs` ” on page 440).
- *Inspected* using the `fnlist` command (see “Listing the Context” on page 431) or the `fnlookup` command (see “Displaying the Binding” on page 430).
- Pruned or destroyed using `fnunbind` command (see “Removing a Composite Name” on page 436) or the `fndestroy` command (see “Destroying a Context” on page 437).

Creating a File Context With `fncreate_fs`

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.

- **Input file.** File context bindings may be provided by an input file (See “Creating File Contexts With an Input File” on page 441)
- **Command line.** File context bindings may be created from the command line (See “Creating File Contexts With Command-line Input” on page 442).

The two methods of `fncreate_fs` have the following syntax:

```
fncreate_fs [-v] [-r] -f file composite_name
fncreate_fs [-v] [-r] composite_name [options] [location...]
```

TABLE 24-1 `fncreate_fs` Command Options

Option	Description
<i>composite_name</i>	The composite name of the file context.
<code>-f file</code>	Use an input file named <i>file</i> .
<i>options</i>	Mount options
<i>location</i>	Mount location.
<code>-v</code>	Sets verbose output, displaying information about the contexts being created and modified.
<code>-r</code>	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, XDR-encoded string.

Creating File Contexts With an 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 one or more entries with the form:

```
name [options] [location...]
```

Where:

- *name* is the reference 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*.
- *options* are mount options, if any. The *options* field begins with a hyphen (“-”). This is followed by a comma-separated list (with no spaces) 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.
- *location* is the mount location. 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.

For each entry a reference to the mount locations and the corresponding mount options is bound to the name *composite_name/name*.

If *options* and *location* are both omitted, then no reference is bound to *composite_name/name*. Any existing reference is unbound.

For example, suppose you want `kuanda`'s file system to be an NFS mount of the directory `/export/home/kuanda` from host `altair` as shown in Figure 21-4. The command would be run as follows:

```
% fncreate_fs -f infile user/kuanda/fs
```

With `infile` containing:

```
. altair:/export/home/kuanda
```

To set up a more complex file system distributed over more than one server as shown in Figure 21-5, run the command

```
% fcreate_fs -f infile org/sales/fs
```

with `infile` containing

```
tools/db altair:/export/db
project altair:/export/proj
project/lib altair:/export/lib
project/src deneb:/export/src
```

To change the NFS mounts for `project` and its subcontexts `src` and `lib` to be read-only, you can change `infile` as follows:

```
tools/db svr1:/export/db
project -ro svr1:/export/projproject/lib altair:/export/lib
project/src svr2:/export/src
```

The `-ro` is unnecessary in the third and fourth lines because `src` and `lib` are subcontexts of `project`, they will inherit the `-ro` mount option from above.

The following input file would make all of the mounts read-only except for `org/sales/fs/project/src`.

```
. -ro
tools/db svr1:/export/db
project svr1:/export/proj
project/lib altair:/export/lib
project/src -rw svr2:/export/src
```

Creating File Contexts With Command-line Input

The `fcreate_fs` command also allows the binding description to be provided on the command line:

```
fcreate_fs composite_name [mount_options] [mount_location ...]
```

This is equivalent to using the input file form of the command but entering the individual bindings from your keyboard. The previous example in which `kuanda`'s file system was set could be set from the command line as follows:

```
% fcreate_fs user/kuanda/fs altair:/export/home/kuanda
```

Similarly, the hierarchy illustrated in Figure 21-5 could have been set up by running the sequence of commands:

```
% fcreate_fs org/sales/fs/tools/db altair:/export/db
% fcreate_fs org/sales/fs/project altair:/export/proj
% fcreate_fs org/sales/fs/project/lib altair:/export/lib
% fcreate_fs org/sales/fs/project/src deneb:/export/src
```

To make all three of the mounts read-only, you would run this command:

```
% fcreate_fs org/sales/fs -ro
```

Advanced Input Formats

The following two sections apply to both input file and command-line input formats.

Multiple Mount Locations

Multiple *location* fields may be specified for NFS file systems that are exported from multiple, functionally equivalent locations:

```
% fcreate_fs org/sales/fs altair:/sales cygnus:/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 altair,cygnus:/sales
```

The hosts may be weighted, with the weighting factor appended to the host name as an integer in parentheses: the lower the number, the more desirable the server. The default weighting factor is zero (most desirable). Negative numbers are not allowed.

The following example illustrates one way to indicate that *cygnus* is the preferred server:

```
% fcreate_fs org/sales/fs altair(2),cygnus(1):/sales
```

Variable Substitution

Variable names, prefixed by \$, may be used in the *options* or *location* fields of *fcreate_fs*. For example, a mount location may be given as:

```
altair:/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 `fcreate_fs`:

```
name [mount_options] [mount_location ...] \  
/offset1 [mount_options1] mount_location1 ... \  
/offset2 [mount_options2] mount_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 [mount_options] [mount_location ...] \  
name/offset1 [mount_options1] mount_location1 ... \  
name/offset2 [mount_options2] mount_location2 ...
```

The first line is omitted if both *mount_options* and *mount_location* are omitted. This format is for compatibility only. It provides no additional functionality, and its use is discouraged.

FNS and Global Naming Systems

This chapter describes two global naming systems (DNS and X.500/LDAP) and how to federate them under FNS.

- “FNS and Global Naming Systems” on page 445
- “Obtaining the Root Reference” on page 446
- “Federating Under DNS” on page 448
- “Federating Under X.500/LDAP” on page 449

FNS and Global Naming Systems

See “Global Naming Services” on page 368 for overview and background information on the relationship between FNS and global naming services.

FNS supports federation of enterprise naming systems into the global naming systems, DNS and X.500/LDAP. 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

Note - You can only federate a global naming service if your enterprise-level name service is NIS+ or NIS. If you are using a files-based name service for your enterprise, you cannot federate either DNS or X.500/LDAP.

Obtaining the Root Reference

To federate an enterprise naming service under DNS or X.500/LDAP, information must be added to these respective naming systems to enable access to and from the enterprise and the global Internet outside of the enterprise. This information is the *root reference*, which consists of network address information describing how to reach the top of a particular enterprise namespace.

The root reference consists of a single address which contains a single, XDR-encoded string. The address type and content varies according to the enterprise level name service you are using: NIS+ or NIS.

NIS+ Root Reference

When your enterprise level name service is NIS+, the root reference address type is: `onc_fn_nisplus_root`. There are two required, and one optional, elements in a root reference network address. The elements are separated by white spaces:

```
root-domain server [server_IP_address]
```

TABLE 25-1 NIS+ Root Reference

Address Element	Description
<i>root_domain</i>	The fully qualified name of the NIS+ root domain (trailing dot required).
<i>server</i>	The host name of one of the NIS+ servers (master or replica) serving <i>nis+_root_domain</i> .
<i>server_IP_address</i>	The IP address of <i>nis+server</i> . This is optional if the address of <i>nis+server</i> is expected to be known. This means it should be available through one of the name services listed in the <code>/etc/nsswitch.conf</code> file.

For example, suppose that the NIS+ root domain is `doc.com.` (notice the trailing dot), and that it can be reached using the host `nismaster.doc.com`. The root reference would be:

```
doc.com. nismaster.doc.com
```

The IP address of the server is not given in the example above because it is expected to be available through other means. If for some reason that IP address was not available through other means, the root reference would look like:

```
doc.com. nismaster.doc.com 123.123.123.33
```

NIS Root Reference

When your enterprise level name service is NIS, the root reference address type is: `onc_fn_nis_root`. There are two required, and one optional, elements in a root reference network address. The elements are separated by white spaces:

```
root_domain server [server_IP_address]
```

TABLE 25-2 NIS Root Reference

Address Element	Description
<i>root_domain</i>	The fully qualified name of the NIS domain (trailing slash required).
<i>server</i>	The host name of one of the NIS servers (master or slave) serving <i>root_domain</i> .
<i>server_IP_address</i>	The IP address of <i>nis-server</i> . This is optional if the address of <i>nis-server</i> is expected to be known. This means it should be available through one of the name services listed in the <code>/etc/nsswitch.conf</code> file.

For example, suppose that the NIS domain is `doc.com`, and that it can be reached using the host `ypmaster.doc.com`. The root reference would be:

```
doc.com/ ypmaster.doc.com
```

The IP address of the server is not given in the example above because it is expected to be available through other means. If for some reason that IP address was not available through other means, the root reference would look like:

```
doc.com/ ypmaster.doc.com 123.123.123.37
```

Federating Under DNS

This section describes the steps required to add TXT (text) records for a subordinate enterprise naming system implemented with NIS+ or 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's root reference.

1. Obtain the root reference.

See "Obtaining the Root Reference" on page 446.

2. Add a root reference TXT record to the DNS loopback file.

By default, this manual uses the name `/etc/named.local` for this file (other common names for this file are `domain.127.0.0` or `db.127.0.0`).

The root reference TXT record has the following format:

For NIS+

```
TXT ``XFNNISPLUS rootdomain server [server_IP_address]``
```

For example:

```
TXT ``XFNNISPLUS doc.com. nismaster.doc.com``
```

For NIS

```
TXT ``XFNNIS rootdomain server [server_IP_address]``
```

For example:

```
TXT ``XFNNIS doc.com/ ypmaster.doc.com``
```

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 domain with reference information for NIS+ bound in it.

```
$ORIGIN doc.com
@ IN SOA foo bar.eng.doc.com
(
  100 ;; Serial
  3600 ;; Refresh
  3600 ;; Retry
  3600 ;; Expire
  3600 ;; Minimum
)
NS nshost
```

(continued)

```
TXT "XFNNISPLUS doc.com. nismaster 123.123.123.33"
nshost IN A 133.33.33.34
```

For more information about DNS files, see “DNS Boot and Data Files” on page 478.

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
```

Where *pid* is the process ID number of `in.named`.

For further information on how DNS TXT records are used for XFN references, see “DNS Text Record Format for XFN References ” on page 619.

Federating Under X.500/LDAP

In order to federate a subordinate naming system (either NIS+ or NIS) in X.500/LDAP:

- Root reference information must be added into X.500 describing how to reach the subordinate naming system.
- An X.500 client API must be specified.

Specifying an X.500 Root Reference

1. Obtain the NIS+ root reference for your NIS+ hierarchy.
See “Obtaining the Root Reference” on page 446.

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=doc` with the object classes `top`, `organization`, and `XFN-supplement` (1.2.840.113536.25). The `XFN-supplement` object class allows

the `c=us/o=doc` entry to store reference information for a subordinate naming system.

```
# fnattr -a ../c=us/o=doc 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 system to the entry.

After creating the X.500 entry, you can then add information about the subordinate system by binding the appropriate root reference to the named entry.

For example, if your subordinate naming system is NIS+, and the NIS+ server you want to use is `nismaster`, you would enter:

```
# fncbind -r ../c=us/o=doc/ onc_fn_enterprise onc_fn_nisplus_root \  
"doc.com. nismaster"
```

If your subordinate naming system is NIS, and the NIS server you want to use is `ypmaster`, you would enter:

```
# fncbind -r ../c=us/o=doc/ onc_fn_enterprise onc_fn_nis_root \  
"doc.com/ ypmaster"
```

These examples bind the reference for the NIS+ or NIS hierarchy with the root domain name `doc.com`, to the next naming system pointer (NNSP) of the X.500 entry `c=us/o=doc`, thus linking the X.500 namespace with the `doc.com` namespace.

The address format used is that of the root reference described in “Obtaining the Root Reference” on page 446. Note the use of the trailing slash in the name argument to `fncbind`, `../c=us/o=doc/`, to signify that the reference is being bound to the NNSP of the entry, rather than to the entry itself.

For further information on X.500 entries and XFN references, see “X.500 Attribute Syntax for XFN References ” on page 621.

Specifying an X.500 Client API

An X.500 client API is required in order to access X.500 using FNS. You can use one of two different clients:

- **XDS/XOM API.** The XDS/XOM API must be installed. It is exported from the `/opt/SUNWxds/lib/libxomxds.so` shared object. Consult “*Getting started with the SunLink X.500 Client Toolkit*” for details on SunSoft’s X.500 product.
- **LDAP (Lightweight Directory Access Protocol) API.** The LDAP API is automatically installed as part of Solaris Release 2.6.

The API that you use is specified in each machine’s `/etc/fn/x500.conf` file. This file contains configuration information for X.500 and LDAP. This file can be edited directly. The default `x500.conf` file contains two entries:

```
x500-access: xds ldap
ldap-servers: localhost ldap
```

Where *localhost* and *ldap* are the IP addresses or hostnames of one or more LDAP servers.

The first entry specifies the order in which X.500 accesses APIs. In the example above, X.500 will first try to use XDS/XOM. If XDS/XOM is not available, it will default to using LDAP. If the entry read: `x500-access: ldap xds`, X.500 would use LDAP and only fall back on XDS if LDAP were not available.

The second entry lists the IP addresses or hostnames of servers running LDAP. Each server is tried in turn until a successful LDAP connection is achieved. In the example above, the *localhost* is tried first. If LDAP is not available on that server, the next one is tried.

Administering FNS Attributes

This chapter describes FNS attributes and how to administer them. .

- “Attributes Overview” on page 453
- “Examining Attributes” on page 453
- “Updating Attributes ” on page 456

Attributes Overview

Attributes may be applied to named objects. Attributes are optional. A named object can have no attributes, one attribute, or multiple attributes.

Each attribute has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values.

XFN defines the base attribute interface for examining and modifying the values of attributes associated with existing named objects. These objects can be contexts or any other type of object. Associated with a context are syntax attributes that describe how the context parses compound names.

The extended attribute interface contains operations that search for specific attributes and that create objects and their associated attributes.

Examining Attributes

Search for attributes with the `fnsearch` command.

The syntax of the `fnsearch` command is

```
fnsearch [-ALLv] [-n max] [-s scope] name [-a ident]... [-O|-U] filter_expr [filter_arg]
```

TABLE 26-1 `fnsearch` Command Options

Option	Description
<code>-n <i>max</i></code>	Display only <i>max</i> number of objects.
<code>-s <i>scope</i></code>	Set the scope of the search
<code>-a <i>ident</i></code>	Display only those attributes that match <i>ident</i> .
<i>name</i>	Composite name
<i>filter_expr</i>	Boolean, logical, grouping, relational, and comparison operators (see Table 26-2)
<i>filter_arg</i>	Arguments for filter expressions (see Table 26-2)
<code>-A</code>	Consult only the authoritative source.
<code>-L</code>	Follow XFN links.
<code>-l</code>	Display the object references for the matching objects.
<code>-v</code>	Verbose. Display detailed object references for the matching objects.
<code>-O</code>	Use an OSI OID as the identifier
<code>-U</code>	Use a DCE UUID as the identifier

Searching for Objects Associated With an Attribute

With the `fnsearch` command, you can search for objects that are associated with the attributes you choose.

For example, to find all the objects that are associated with the attribute `for_sale` in `orgunit/sales/site/`, you would enter the following command:

```
% fnsearch orgunit/sales/site/ for_sale
```

Customizing Attribute Searches

You can also use all the following in filter expressions in your search patterns.

TABLE 26-2 `fnsearch` Filter Expression Operators

Filter Expression Operator	Symbols and Forms
Logical operators	or, and, not
Parentheses for grouping	()
Relational operators: Compare an attribute to a supplied value	<code>==</code> True if at least one attribute value is equal to the supplied value. <code>!=</code> True if none of the attribute values are equal to the supplied value. <code><</code> True if at least one attribute value is less than the supplied value. <code><=</code> True if at least one attribute value is less than or equal to the supplied value. <code>></code> True if at least one attribute value is greater than the supplied value. <code>>=</code> True if at least one attribute value is greater than or equal to the supplied value. <code>~=</code> True if at least one attribute value matches the supplied value according to some context-specific approximate matching criterion. This criterion must subsume strict equality.
Example:	<pre>% fnsearch name "not (make == 'olds' and year == 1983)"</pre>
Substitution tokens: Helpful when writing shell scripts; allow the use of OSI OIDs and DCE UUIDs when used with the <code>-O</code> and <code>-U</code> options	<code>%a</code> for attribute <code>%s</code> for string <code>%i</code> for identifier <code>%v</code> for attribute value (only <code>fn_attr_syntax_ascii</code> is currently supported)
Example:	The following three examples are equivalent. <pre>% fnsearch name "color == 'red'" % fnsearch name "%a == 'red'" color % fnsearch name "%a == %s" color red</pre>

TABLE 26-2 `fnsearch` Filter Expression Operators (continued)

Filter Expression Operator	Symbols and Forms
Wild card strings	<code>*</code> , <code>*string</code> , <code>string*</code> , <code>str*ing</code> , <code>%s*</code>
Extended operators	<code>'name'</code> (<i>wildcarded_string</i>), <code>'reftype'</code> (<i>identifier</i>), <code>'addrtype'</code> (<i>identifier</i>)
Example:	Search for objects with names starting with "Bill" and IQ attributes over 80. <code>% fnsearch name "'name'('bill'*) and IQ > 80"</code>

See the `fnsearch` man page for detailed information about creating search patterns.

Updating Attributes

The `fnattr` command lets you update and examine attributes associated with FNS named objects. You can perform four attribute operations with the `fnattr` command:

- Add an attribute:

```
fnattr -a [-s] name [-O|-U] identifier values
```

- Delete an attribute:

```
fnattr -d name [[-O|-U] identifier [values]]
```

- Modify an attribute:

```
fnattr -m name [-O|-U] identifier oldvalue newvalue
```

- List an attribute:

```
fnattr -l name [[-O|-U] identifier
```

TABLE 26-3 `fnattr` Command Options

Option	Description
<i>name</i>	Composite name.
<i>identifier</i>	Attribute name.
<i>values</i>	One or more attributes values
<i>oldvalue</i>	The attribute value that you want to change.
<i>newvalue</i>	The new attribute value.
-a	Add (create) a new attribute
-d	Delete an attribute
-m	Change (modify) an attribute
-l	List attribute values
-s	Add in "supersede" mode. Removes any existing values for the <i>identifier</i> attribute and creates new attribute values.
-L	List attributes and values.
-O	Use an OSI OID as the identifier
-U	Use a DCE UUID as the identifier

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. Specify the composite name the attribute is associated with, the attribute identifier, and the values to add.

```
fnattr -a [-s] name [-O | -U] identifier value1 [value2+]
```

The following example adds the attribute identifier `model` and the value `hplaser` to `thisorgunit/service/printer`.

```
# fnattr -a thisorgunit/service/printer model hplaser
```

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 them with the values added. If this option is omitted, the resulting values for the specified attribute includes the existing values and the new values added.

```
# fnattr -as thisorgunit/service/printer model hplaser
```

The example above will first remove any existing values associated with `model` and add `hplaser` as the value.

Deleting an Attribute

To delete an attribute associated with an FNS named object, use the `-d` option.

```
fnattr -d name [[-O | -U] identifier value1 [value2+]]
```

You can control what to delete:

- *Name* only. If only the composite name is specified and no attribute identifier is specified, all the attributes associated with the named object are removed.
- *Name* and *identifier* only. If only the composite name and an attribute identifier is specified, but no attribute values are specified, the entire attribute identified by *identifier* is removed.
- *Name*, *identifier*, and *values*. If the composite name, an attribute identifier, and one or more attribute values are specified, then only those values are removed from the attribute. (Removal of the last remaining value of an attribute is the same as removing the attribute itself.)

For example, to delete 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.

```
fnattr -l name [[-O | -U] identifier]
```

For example to list 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.

```
fnattr -m name [-O | -U] identifier old_value new_value
```

For example, to replace the value `postscript` with `laser` you would enter:

```
# fnattr -m thisorgunit/service/printer model postscript laser
```

Only the specified values are affected. Other attributes and values associated with the name are not affected.

Other Options

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).

PART VI Administering DNS

This part describes the *Domain Name System (DNS)* and how to administer it.

- Chapter 27
- Chapter 28

Introduction to DNS

This chapter describes the structure and provides an overview of the Domain Name System (DNS).

- “DNS Basics” on page 464
- “Introducing the DNS Namespace” on page 468
- “Zones” on page 473
- “DNS Servers” on page 474
- “How DNS Affects Mail Delivery” on page 477
- “DNS Boot and Data Files” on page 478
- “Names of DNS Data Files” on page 478
- “Data File Resource Record Format” on page 481
- “Standard Resource Record Format” on page 481
- “Special Resource Record Characters” on page 482
- “Control Entries” on page 483
- “Resource Record Types” on page 484
- “Solaris DNS BIND Implementation” on page 491

See *Solaris Naming Setup and Configuration Guide* for information on initially setting up and configuring DNS service.

Note - One of the most common, and important, uses of DNS is connecting your network to the global Internet. In order to connect to the Internet, your network IP address must be registered with whomever is administering your parent domain. Who that administrator is varies according to your geographic location and type of parent domain.

DNS Basics

The Domain Name System (DNS) is an application-layer protocol that is part of the standard TCP/IP protocol suite. This protocol implements the DNS name service, which is the name service used on the Internet.

This section introduces the basic DNS concepts. It assumes that you have some familiarity with network administration, particularly TCP/IP, and some exposure to other name services, such as NIS+ and NIS.

Refer to *Solaris Naming Setup and Configuration Guide* for information regarding initial setup and configuration of DNS.

Note - DNS, NIS+, NIS, and FNS provide similar functionality and sometimes use the same terms to define different entities. Thus, this chapter takes care to define terms like domain and name server according to their DNS functionality, a very different functionality than NIS+ and NIS domains and servers.

Name-to-Address Resolution

Though it supports the complex, world-wide hierarchy of computers on the Internet, the basic function of DNS is actually very simple: providing *name-to-address resolution* for TCP/IP-based networks. Name-to-address resolution, also referred to as “mapping,” is the process of finding the IP address of a computer in a database by using its host name as an index.

Name-to-address mapping occurs when a program running on your local machine needs to contact a remote computer. The program most likely will know the host name of the remote computer but may not know how to locate it, particularly if the remote machine is in another company, miles from your site. To get the remote machine’s address, the program requests assistance from the DNS software running on your local machine, which is considered a *DNS client*.

Your machine sends a request to a *DNS name server*, which maintains the distributed DNS database. The files in the DNS database bear little resemblance to the NIS+ Host Table or even the local `/etc/hosts` file, though they maintain similar information: the host names, IP addresses, and other information about a particular group of computers. The name server uses the host name your machine sent as part of its request to find or “resolve” the IP address of the remote machine. It then returns this IP address to your local machine IF the host name is in its DNS database.

Figure 27-1 shows name-to-address mapping as it occurs between a DNS client and a name server, probably on the client’s local network.

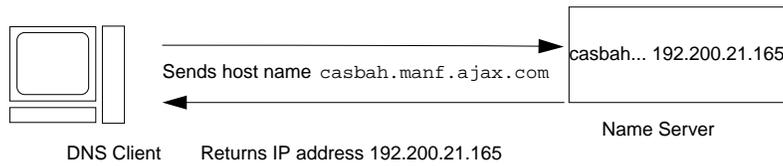


Figure 27-1 Name to Address Resolution

If the host name is not in that name server’s DNS database, this indicates that the machine is outside of its authority, or, to use DNS terminology, outside the *local administrative domain*. Thus, each name server is spoken of as being “authoritative” for its local administrative domain.

Fortunately, the local name server maintains a list of host names and IP addresses of *root domain name servers*, to which it will forward the request from your machine. These root name servers are authoritative for huge organizational domains, as explained fully in “DNS Hierarchy and the Internet” on page 469. These hierarchies resemble UNIX file systems, in that they are organized into an upside-down tree structure.

Each root name server maintains the host names and IP address of top level domain name servers for a company, a university, or other large organizations. The root name server sends your request to the top-level name servers that it knows about. If one of these servers has the IP address for the host you requested, it will return the information to your machine. If the top-level servers do not know about the host you requested, they pass the request to second-level name servers for which they maintain information. Your request is then passed on down through the vast organizational tree. Eventually, a name server that has information about your requested host in its database will return the IP address back to your machine.

Figure 27-2 shows name-to-address resolution outside the local domain.

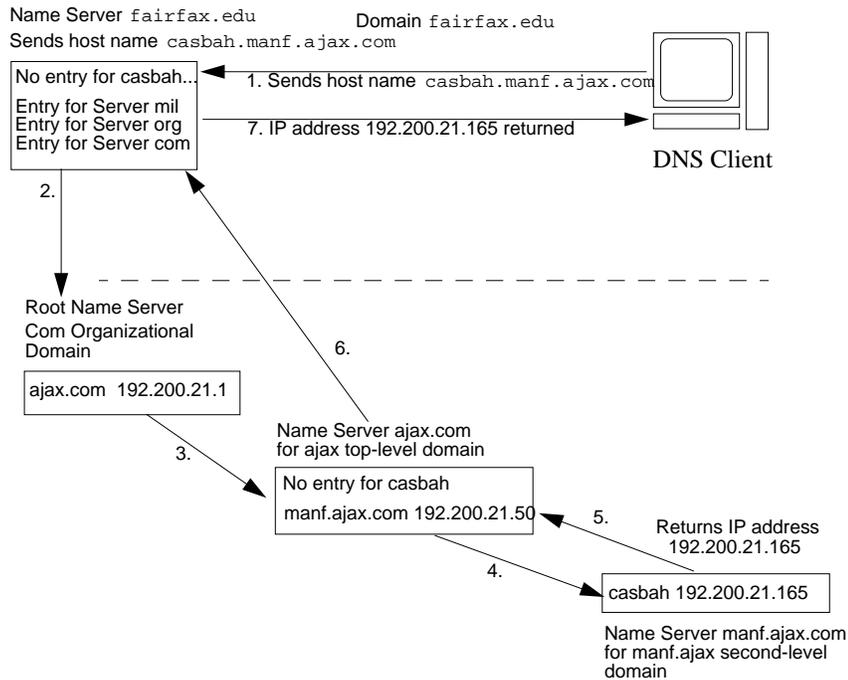


Figure 27-2 Name to Address Resolution for a Remote Host

DNS Administrative Domains

From a DNS perspective, an *administrative domain* is a group of machines that are administered as a unit. Information about this domain is maintained by at least two name servers; they are “authoritative” for the domain. The DNS domain is a purely logical grouping of machines. It could correspond to a physical grouping of machines, such as all machines attached to the Ethernet in a small business. But a local DNS domain just as likely could include all machines on a vast university network that belong to the computer science department or to university administration.

For example, suppose the Ajax company has two sites, one in San Francisco and one in Seattle. The `Retail.Sales.Ajax.com` domain might be in Seattle and the `Wholesale.Sales.Ajax.com` domain might be in San Francisco. One part of the `Sales.Ajax.com` domain would be in one city, the other part in the second city.

Each administrative domain must have its own unique subdomain name. Moreover, if you want your network to participate in the Internet, the network must be part of a registered administrative domain. The section “Joining the Internet” on page 470 has full details about domain names and domain registration.

in.named and DNS Name Servers

As mentioned previously, name servers in an administrative domain maintain the DNS database. They also run the `in.named` daemon, which implements DNS services, most significantly, name-to-address mapping. `in.named` is a public domain TCP/IP program and included with the Solaris 2.6 Solaris 2.6 release environment.

Note - The `in.named` daemon is also called the Berkeley Internet Name Domain service, or BIND, because it was developed at the University of California at Berkeley.

There are three types of DNS name servers:

- Primary server
- Secondary server
- Cache-only server

Each domain must have one primary server and should have at least one secondary server to provide backup. “Zones” on page 473 explains primary and secondary servers in detail.

DNS Clients and the Resolver

To be a DNS client, a machine must run the *resolver*. The resolver is neither a daemon nor a single program; rather, it is a set of dynamic library routines used by applications that need to know machine names. 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. Once the resolver is configured, a machine can request DNS service from a name server.

When a machine’s `/etc/nsswitch.conf` file specifies `hosts: dns` (or any other variant that includes `dns` in the `hosts` line), the resolver libraries are automatically used. If the `nsswitch.conf` file specifies some other name service before `dns`, that name service is consulted first for host information and only if that name service does not find the host in question are the resolver libraries used.

For example, if the `hosts` line in the `nsswitch.conf` file specifies `hosts: nisplus dns`, the NIS+ name service will first be searched for host information. If the information is not found in NIS+, then the DNS resolver is used. Since name services such as NIS+ and NIS only contain information about hosts in their own network, the effect of a `hosts:nisplus dns` line in a switch file is to specify the use of NIS+ for local host information and DNS for information on remote hosts out on the Internet.

There are two kinds of DNS clients:

- *Client-only*. A client-only DNS client does not run `in.named`; instead, it consults the resolver. The resolver knows about a list of name servers for the domain, to which queries are then directed.

- *Client-server.* A client-server uses the services provided by `in.named` to resolve queries forwarded to it by client-machine resolvers.

The Solaris 2.6 release environment includes the dynamic library routines that make up the resolver. *Solaris Naming Setup and Configuration Guide*, contains instructions for setting up a host as a DNS client.

Introducing the DNS Namespace

The entire collection of DNS administrative domains throughout the world are organized in a hierarchy called the *DNS namespace*. This section shows how the namespace organization affects both local domains and the Internet.

DNS Namespace Hierarchy

Like the UNIX file system, DNS domains are organized as a set of descending branches similar to the roots of a tree. Each branch is a domain, each subbranch is a *subdomain*. The terms *domain* and *subdomain* are relative. A given domain is a subdomain relative to those domains above it in the hierarchy, and a parent domain to the subdomains below it.

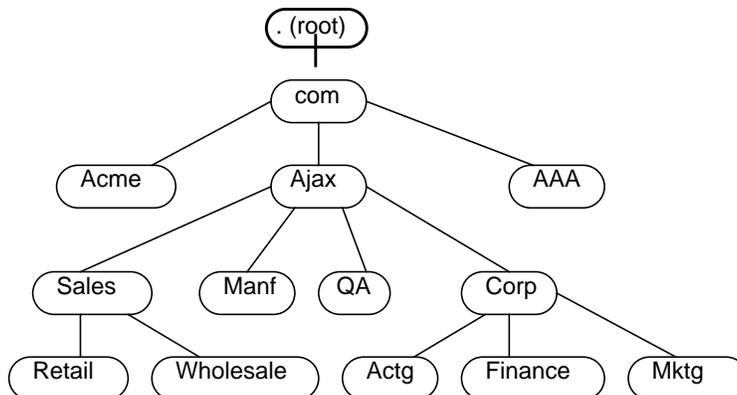


Figure 27-3 Domains and Subdomains

For example, in Figure 27-3, `com` is a parent domain to the `Acme`, `Ajax`, and `AAA` domains. Or you could just as easily say that those are subdomains relative to the `com` domain. In its turn, the `Ajax` domain is a parent to four subdomains (`Sales`, `Manf`, `QA`, and `Corp`).

A domain contains one parent (or top) domain plus the associated subdomains if any. Domains are named up the tree starting with the lowest (deepest) subdomain

and ending with the root domain. For example, `Mktg.Corp.Ajax.Com.` from Figure 27-3.

DNS Hierarchy in a Local Domain

If your company is large enough, it may support a number of domains, organized into a local namespace. Figure 27-4 shows a domain hierarchy that might be in place in a single company. The top-level, or “root” domain for the organization is `ajax.com`, which has three sub-domains, `sales.ajax.com`, `test.ajax.com`, and `manf.ajax.com`.

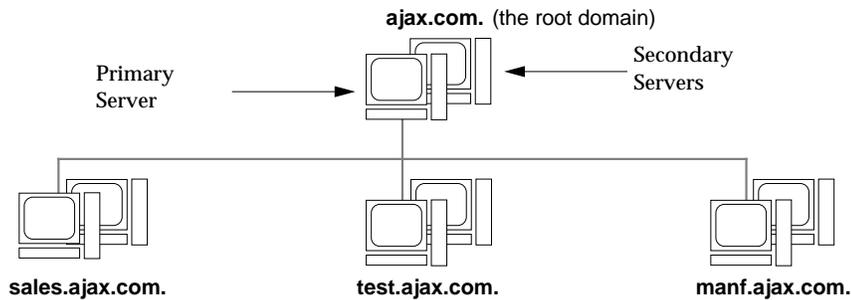


Figure 27-4 Hierarchy of DNS Domains in a Single Organization

DNS clients request service only from the servers that support their domain. If the domain’s server does not have the information the client needs, it forwards the request to its parent server, which is the server in the next-higher domain in the hierarchy. If the request reaches the top-level server, the top-level server determines whether the domain is valid. If it is *not* valid, the server returns a “not found” type message to the client. If the domain is valid, the server routes the request down to the server that supports that domain.

DNS Hierarchy and the Internet

The domain hierarchy shown in Figure 27-4 is, conceptually, a “leaf” of the huge DNS namespace supported on the global Internet.

The DNS namespace for the Internet is organized hierarchically as shown Figure 27-5. It consists of the root directory, represented as a dot (`.`) and two top level domain hierarchies, one organizational and one geographical. Note that the `com` domain introduced in Figure 27-3 is one of a number of top-level organizational domains in existence on the Internet.

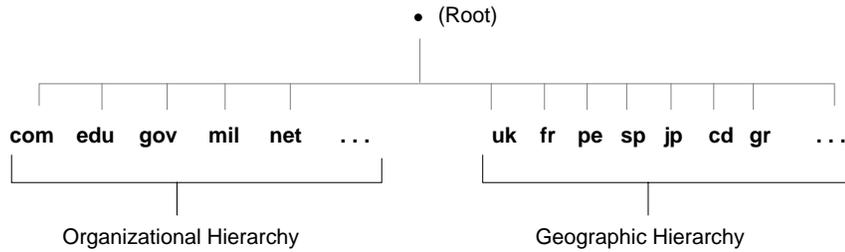


Figure 27-5 Hierarchy of Internet Domains

At the present time, the organizational hierarchy divides its namespace into the top-level domains listed shown in Table 27-1. It is probable that additional top-level organizational domains will be added in the future.

TABLE 27-1 Internet Organizational Domains

Domain	Purpose
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
mil	Military groups
net	Major network support centers
org	Nonprofit organizations and others
int	International organizations

The geographic hierarchy assigns each country in the world a two- or three-digit identifier and provides official names for the geographic regions within each country. For example, domains in Britain are subdomains of the `uk` top-level domain, Japanese domains are subdomains of `jp`, and so on.

Joining the Internet

The Internet root domain, top-level domains (organizational and geographical) are maintained by the various Internet governing bodies. People with networks of any size can “join” the Internet by registering their domain name in either the organizational or the geographical hierarchy.

Every DNS domain must have a domain name. If your site wants to use DNS for name service *without* connecting to the Internet, you can use any name your organization wants for its your domains and subdomains, if applicable. However, if your site plans wants to join the Internet, it *must* register its domain name with the Internet governing bodies.

To join the Internet, you have to:

- Register your DNS domain name with the an appropriate Internet governing body.
- Obtain a network IP address from that governing body.

There are two ways to accomplish this:

- You can communicate directly with the appropriate Internet governing body or their agent. In the United States, InterNIC is the company that currently handles network address and domain registration matters.
- You can contract with an Internet Service Provider (ISP) to assist you. ISPs provide a wide range of services from consulting to actually hosting your Internet presence.

Domain Names

Domain names indicate a domain's position in the overall DNS namespace, much as path names indicate a file's position in the UNIX file system. After your local domain is registered, its name is prepended to the name of the Internet hierarchy to which it belongs. For example, the `ajax` domain shown in Figure 27-4 has been registered as part of the Internet `com` hierarchy. Therefore, its Internet domain name becomes `ajax.com`.

Figure 27-6 shows the position of the `ajax.com` domain in the DNS namespace on the Internet.

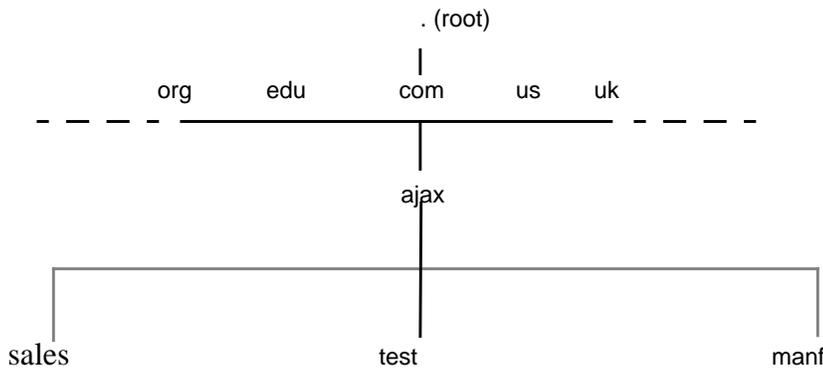


Figure 27-6 Ajax Domain's Position in the DNS Namespace

The `ajax.com` subdomains now have the following names.

```
sales.ajax.com
test.ajax.com
manf.ajax.com
```

DNS does not require domain names to be capitalized, though they may be. Here are some examples of machines and domain names:

```
Boss.manf.ajax.com
quota.Sales.ajax.com
```

The Internet organization regulates administration of its domains by granting each domain authority over the names of its hosts and by expecting each domain to delegate authority to the levels below it. Thus, the com domain has authority over the names of the hosts in its domain. It also authorizes the formation of the Ajax.com domain and delegates authority over the names in that domain. The Ajax.com domain, in turn, assigns names to the hosts in its domain and approves the formation of the Sales.Ajax.com, Test.Ajax.com, and Manf.Ajax.com domains.

Fully-Qualified Domain Names

A domain name is said to be *fully-qualified* when it includes the names of every DNS domain from the local domain on up to “.”, the DNS root domain. Conceptually, the fully-qualified domain name indicates the path to the root, as does the absolute path name of a UNIX file. However, fully-qualified domain names are read from lowest, on the left, to highest, on the right. Therefore, a fully-qualified domain name has the syntax:

<local_domain_name>.<Internet_Org_name> .

root domain 

The fully qualified domain names for the ajax domain and its subdomains are:

```
ajax.com.
sales.ajax.com
test.ajax.com.
manf.ajax.com
```

Note the dot at the furthest right position of the name.

Zones

DNS service for a domain is managed on the set of name servers first introduced “in.named and DNS Name Servers” on page 467. Name servers can manage a single domain, or multiple domains, or domains and some or all of their corresponding subdomains. The part of the namespace that a given name server controls is called a *zone*; thus, the name server is said to be authoritative for the zone. If you are responsible for a particular name server, you may be given the title zone administrator.

The data in a name server’s database are called *zone files*. One type of zone file stores IP addresses and host names. When someone attempts to connect to a remote host using a host name by a utility like `ftp` or `telnet`, DNS performs name-to-address mapping, by looking up the host name in the zone file and converting it into its IP address.

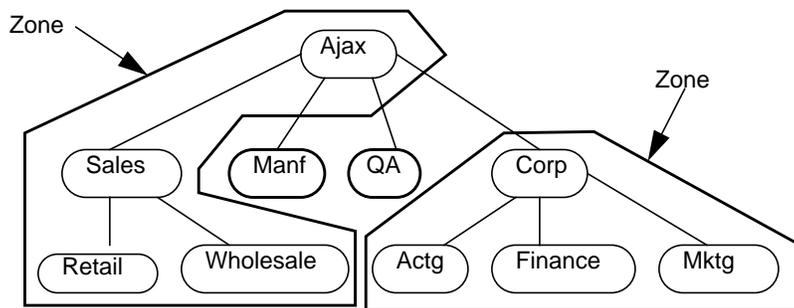


Figure 27-7 Domains and Zones

For example, the `Ajax` domain shown in Figure 27-7 contains a top domain (`Ajax`), four subdomains, and five sub-subdomains. It is divided into four zones shown by the thick lines. Thus, the `Ajax` name server administers a zone composed of the `Ajax`, `Sales`, `Retail`, and `Wholesale` domains. The `Manf` and `QA` domains are zones unto themselves served by their own name servers, and the `Corp` name server manages a zone composed of the `Corp`, `Actg`, `Finance`, and `Mktg` domains.

Reverse Mapping

The DNS database also include zone files that use the IP address as a key to find the host name of the machine, enabling IP address to host name resolution. This process is called *reverse resolution* or more commonly, reverse mapping. Reverse mapping is used primarily to verify the identity of the machine that sent a message or to authorize remote operations on a local host.

The `in-addr.arpa` Domain

The `in-addr.arpa` domain is a conceptual part of the DNS namespace that uses IP addresses for its leaves, rather than domain names. It is the part of your zone that enables address to name mapping.

Just as DNS domain names are read with the lowest level subdomain occupying the furthest left position and the root at the far right, `in-addr.arpa` domain IP addresses are read from lowest level to the root. Thus, the IP addresses are read backward. For example, suppose a host has the IP address `192.200.21.165`. In the `in-addr.arpa` zone files, its address is listed as `165.21.200.192.in-addr.arpa.` with the dot at the end indicating the root of the `in-addr.arpa` domain.

DNS Servers

DNS servers perform one or more functions:

- *Zone Master Servers.* A master name server maintains all the data corresponding to the zone, making it the authority for that zone. Master servers are commonly called *authoritative* name servers. (See “Master Servers” on page 475.)

There are two types of master server:

- *Zone primary master server.* Each zone has one server that is designated as the *primary* master server for that zone. (See “Primary Master Server” on page 475.)
- *Zone secondary master server.* A zone may have one or more *secondary* master servers. Secondary master servers obtain their DNS data from the zone’s primary master server. (See “Primary Master Server” on page 475.)
- *Cache-only Server.* All servers are caching servers in the sense that they maintain a cache of DNS data. A cache-only server is a server that is not a master server for any zone other than the `in-addr.arpa.` domain. (See “Caching and Cache-only Servers” on page 475.)
- *Root Domain servers.* A root domain server is the authoritative server for the top of your DNS domain hierarchy. If your network is connected to the Internet, your root domain servers are out on the Internet itself. If your network is not connected to the Internet, you must set up your own root domain server. (See “Root Domain Name Server” on page 476.)

These different server functions can be performed by the same machine. For example, a machine can be a primary master server for one zone and a secondary master server for another zone. When this manual refers to a primary or secondary or cache-only server, it is not referring to a particular machine, but the role that machine plays for a given zone.

Master Servers

The master name servers maintain all the data corresponding to the zone, making them the authority for that zone. These are commonly called *authoritative* name servers. The data corresponding to any given zone should be available on at least two authoritative servers. You should designate one name server as the *primary* master server and at least one more as a *secondary* master server, to act as a backup if the primary is unavailable or overloaded.

A server may function as a master for multiple zones: as a primary for some zones, and as a secondary for others.

Primary Master Server

The *primary* master server is the DNS name server that loads the master copy of its data from disk when it starts `in.named`. A zone's primary master server is where you make changes for the zone. The primary master is the source for DNS information regarding its zone. The primary server may also delegate authority to secondary servers in its zone as well as to servers outside its zone.

Secondary Master Server

A *secondary* master server maintains a copy of the data for the zone. The primary server sends its data and delegates authority to the secondary server. Clients can query a secondary server for DNS information. By using secondary servers, you can improve response time and reduce network overhead by spreading the load over multiple machines. Secondary servers also provide redundancy in case the primary server is not available.

When the secondary server starts `in.named`, it requests all the data for the given zone from the primary. The secondary server then periodically checks with the primary to see if it needs to update its database. The process of sending the most recent zone database from the primary to the secondary is called a *zone transfer*. Thus, you do not modify data files on a secondary server, you modify the data files on the zone's primary server and the secondary servers update their files from the primary.

Caching and Cache-only Servers

All name servers are *caching servers*. This means that the name server caches received information until the data expires. The expiration process is regulated by the time-to-live (TTL) field that may be attached to the data.

Additionally, you can set up a *cache-only server* that is not authoritative for any zone. A cache-only server is a server that is not a master server for any zone other than the `in-addr.arpa` domain. A cache-only server handles the same kind of queries

from clients that authoritative name servers perform. But the cache-only server does not maintain any authoritative data itself.

A cache-only server requires less memory than an authoritative server, but cannot function by itself if no primary or secondary servers are available.

Root Domain Name Server

A DNS name space must have one or more *root domain name servers* that are authoritative for the root domain.

- If your network is connected to the Internet, your root domain server exists at the root domain Internet site and all you have to do is provide that site's Internet IP addresses in your cache file as explained in "Internet Root Domain Server" on page 476.
- If your network is not connected to the Internet, you must set up primary and secondary name servers in the root-level domain on your local network as explained in "Non-Internet Root Domain Server" on page 477. This is so that all domains in your network have a consistent authoritative server to which to refer; otherwise, machines may not be able to resolve queries.

The information that identifies the root domain name servers is stored in a cache file. This manual and most Solaris sites call this file `named.ca`. (Other common names for this file are: `root.cache`, `named.root`, or `db.cache`.) Each server's boot file contains a record identifying the file that holds the root domain name server information.

Internet Root Domain Server

If your site is connected to the Internet, your DNS name server's boot files must point to a common cache file (usually called `named.ca`) that identifies the root domain name servers. A template for this file may be obtained from InterNIC registration services via:

- **Anonymous FTP.** The FTP site is: `ftp.rs.internic.net`. The file name is: `/domain/named.root`.
- **Gopher.** The Gopher site is: `rs.internic.net`. The file is: `named.root` which can be found under the InterNIC Registration Services menu, InterNIC Registration Archives submenu.

If you are naming your DNS files according to the conventions in this manual, you need to move this file to `/var/named/named.ca`.

Non-Internet Root Domain Server

If your site is not connected to the Internet, you must set up one or more of your servers to perform as root domain name servers. The boot files of all DNS name servers on your network must point to a common cache file (usually called `named.ca`) that identifies the root domain name servers. You then create a cache file that identifies your root name servers.

Since a single machine can be the primary domain name server for more than one machine, the easiest way to create a root domain name server is to have the server for your highest level domain also be the server for the logical “.” domain.

For example, suppose you have given your network the domain name `solo`. The DNS master name server is `dnsmaster.solo.` (with a trailing dot). In this case, you would make `dnsmaster` the root master server for the “.” domain.

If your network has more than one top-level domain, the root domain server name should be the primary name server for all top-level domains. For example, if your network is divided into two separate, non-hierarchical domains named `solo` and `private`, the same server must be root master server for both of them. Following the example above that would mean that `dnsmaster.solo.` is root domain master for both the `solo` and the `private` domains.

How DNS Affects Mail Delivery

DNS provides two principal services, it performs name to address mapping (and also maps addresses to host names), as discussed in “Name-to-Address Resolution” on page 464. It also helps mail delivery agents, such as `sendmail` and POP, deliver mail along the Internet.

To deliver mail across the Internet, DNS uses *mail exchange records* (MX records). Most organizations don’t allow direct delivery of mail that comes across the Internet for hosts within the organization. Instead, they use a central mail host (or a set of mail hosts) to intercept incoming mail messages and route them to their recipients.

The mail exchange record identifies the mail host that services each machine in a domain. Therefore, a mail exchange record lists the DNS domain names of remote organizations and either the IP address or the host name of its corresponding mail host.

DNS Boot and Data Files

In addition to the `in.named` daemon, DNS on a name server consists of a boot file called `named.boot`, a resolver file named `resolv.conf`, and four types of zone data files.

Names of DNS Data Files

So long as you are internally consistent, you can name the zone data files anything you want. This flexibility may lead to some confusion when working at different sites or referring to different DNS manuals and books.

For example, the file names used in Sun manuals and at most many Solaris sites vary from those used in the book *DNS and BIND* by Albitz and Liu, O'Reilly & Associates, 1992, and both of those nomenclatures have some differences from that used in the public-domain *Name Server Operations Guide for BIND*, University of California.

In addition, this manual and other DNS documentation uses generic names that identify a file's main purpose, and specific example names for that file in code record samples. For example, *Solaris Naming* manuals use the generic name `hosts` when describing the function and role of that file, and the example names `db.doc` and `db.sales.doc` in code samples.

For reference purposes, Table 27-2 compares BIND file names from these three sources:

TABLE 27-2 BIND File Name Examples

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
<code>/etc/named.boot</code>	<code>/etc/named.boot</code>	<code>/etc/named.boot</code>	The boot file specifies the type of server it is running on and the zones over which it has control. It contains a list of domain names and the names of the data files.
<code>/etc/resolv.conf</code>	<code>/etc/resolv.conf</code>	<code>/etc/resolv.conf</code>	This file resides on every DNS client (including DNS servers) and designates the servers which the client queries for DNS information.

TABLE 27-2 BIND File Name Examples (continued)

Solaris Names	O'Reilly Names or other names	U.C. Berkeley Names	Content and Purpose of File
named.ca	db.cache db.root	root.cache	This file establishes the names of root servers and lists their addresses.
Generic: hosts Examples: db.doc db.sales	Generic: db.domain Examples: db.movie db.fx	Generic: hosts Example: ucbhosts	This file contains all the data about the machines in the local zone that the server serves.
Generic: hosts.rev Examples: doc.rev	Generic: db.ADDR Examples: db.192.249.249 db.192.249.253	hosts.rev	This file specifies a zone in the in-addr.arpa. domain, a special domain that allows reverse (address-to-name) mapping.
named.local	Generic: db.cache Example: db.127.0.0	named.local	This file specifies the address for the local loopback interface, or localhost
\$INCLUDE files	\$INCLUDE files	\$INCLUDE files	Any file identified by an \$INCLUDE () statement in a data file.

The named.boot File

The boot file (`/etc/named.boot`) establishes the server as a primary, secondary, or cache-only name server. It also specifies the zones over which the server has authority and which data files it should read to get its initial data.

The boot file is read by `in.named` when the daemon is started by the server's start up script, `/etc/init.d/inetsvc`. The boot file directs `in.named` either to other servers or to local data files for a specified domain.)

The named.ca File

The `named.ca` file establishes the names of root servers and lists their addresses. If your network is connected to the Internet, `named.ca` lists the Internet name servers;

otherwise, it lists the root domain name servers for your local network. The `in.named` daemon cycles through the list of servers until it contacts one of them. It then obtains from that server the current list of root servers, which it uses to update `named.ca`.

The `hosts` File

The `hosts` file contains all the data about the machines in the local zone. The name of this file is specified in the boot file. To avoid confusion with `/etc/hosts`, name the file something other than `hosts`, for example, you could name these files using the pattern `db.domain`. Using that nomenclature, the host files for the `doc.com` and `sales.doc.com` domains might be `db.doc` and `db.sales`.

The `hosts.rev` File

The `hosts.rev` file specifies a zone in the `in-addr.arpa` domain, the special domain that allows reverse (address-to-name) mapping. The name of this file is specified in the boot file.

The `named.local` File

The `named.local` file specifies the address for the local loopback interface, or `localhost`, with the network address `127.0.0.1`. The name of this file is specified in the boot file. Like other files, you can give it a name other than the name used in this manual.

`$INCLUDE` Files

An include file is any file named in an `$INCLUDE()` statement in a DNS data file. `$INCLUDE` files can be used to separate different types of data into multiple files for your convenience.

For example, suppose a data file contained following line:

```
$INCLUDE /etc/named/data/mailboxes
```

This line causes the `/etc/named/data/mailboxes` file to be loaded at that point. In this instance, `/etc/named/data/mailboxes` is an `$INCLUDE` file. Use of `$INCLUDE` files is optional. You can use as many as you wish, or none at all.

Data File Resource Record Format

All the data files used by the DNS daemon `in.named` are written in standard resource record format. Each DNS data file must contain certain resource records. This section describes the DNS data files and the resource records each file should contain.

Standard Resource Record Format

In the standard resource record format, each line of a data file is called a *resource record* (RR), which contains the following fields separated by white space:

<i>namettl</i>	<i>class</i>	<i>record-type</i>	<i>record-specific-data</i>
----------------	--------------	--------------------	-----------------------------

The order of the fields is always the same; however, the first two are optional (as indicated by the brackets), and the contents of the last vary according to the *record-type* field.

The *name* Field

The first field is the name of the domain that applies to the record. If this field is left blank in a given RR, it defaults to the name of the previous RR.

A domain name in a zone file can be either a fully qualified name, terminated with a dot, or a relative name, in which case the current domain is appended to it.

The *ttl* Field

The second field is an optional time-to-live field. This specifies how long (in seconds) this data will be cached in the database before it is disregarded and new information is requested from a server. By leaving this field blank, the *ttl* defaults to the minimum time specified in the Start-Of-Authority (SOA) resource record.

If the *ttl* value is set too low, the server will incur a lot of repeat requests for data refreshment; if, on the other hand, the *ttl* value is set too high, changes in the information will not be timely distributed.

Most *ttl* values should be initially set to between a day (86400) and a week (604800). Then, depending on the frequency of actual change of the information, you can change the appropriate *ttl* values to reflect that frequency. Also, if you have some *ttl* values that have very high numbers because you know they relate to data that rarely changes. When you know that the data is now about to change, reset the *ttl* to a low

value (3600 to 86400) until the change takes place. Then change it back to the original high value.

All RR's with the same name, class, and type should have the same *tll* value.

The *class* Field

The third field is the record *class*. Only one *class* is currently in use: IN for the TCP/IP protocol family.

The *record-type* Field

The fourth field states the resource record *type*. There are many types of RR's; the most commonly used types are discussed in "Resource Record Types" on page 484.

The *record-specific-data* Field

The contents of the *record-specific-data* field depend on the type of the particular resource record.

Although case is preserved in names and data fields when loaded into the name server, all comparisons and lookups in the name server database are case insensitive. However, this situation may change in the future; thus, you should be consistent in your use of lower and uppercase.

Special Resource Record Characters

The following characters have special meanings:

TABLE 27-3 Special Resource Record Characters

Character	Definition
.	A free-standing dot in the name field refers to the current domain.
@	A free-standing @ in the name field denotes the current origin.
..	Two free-standing dots represent the null domain name of the root when used in the name field.
\X	Where X is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, you can use \. to place a dot character in a label.

TABLE 27-3 Special Resource Record Characters (continued)

Character	Definition
<code>\DDD</code>	Where each <i>D</i> is a digit, this is the octet corresponding to the decimal number described by <i>DDD</i> . The resulting octet is assumed to be text and is not checked for special meaning.
<code>()</code>	Use parentheses to group data that crosses a line. In effect, line terminations are not recognized within parentheses.
<code>;</code>	A semicolon starts a comment; the remainder of the line is ignored.
<code>*</code>	An asterisk signifies a wildcard.

Most resource records have the current origin appended to names if they are not terminated by a dot (.). This is useful for appending the current domain name to the data, such as machine names, but may cause problems when you do not want this to happen. You should use a fully qualified name ending in a period if the name is not in the domain for which you are creating the data file.

Control Entries

The only lines that do not conform to the standard RR format in a data file are control-entry lines. There are two kinds of control entries: `$INCLUDE()` and `$ORIGIN()`.

\$INCLUDE

An include line begins with `$INCLUDE` in column 1, and is followed by a file name (known as the `$INCLUDE` file). This feature is particularly useful for separating different types of data into multiple files as in this example:

```
$INCLUDE /etc/named/data/mailboxes
```

The line is interpreted as a request to load the `/etc/named/data/mailboxes` file at that point. The `$INCLUDE` command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

Use of `$INCLUDE` statements and files is optional. You can use as many as you wish, or none at all.

\$ORIGIN()

The \$ORIGIN command is a way of changing the origin in a data file. The line starts in column 1, and is followed by a domain name. It resets the current origin for relative domain names (for example, not fully qualified names) to the stated name. This is useful for putting more than one domain in a data file.

Note - You cannot use \$ORIGIN() for putting more than one zone in a data file.

Use of \$ORIGIN commands in a data file is optional. If there is no \$ORIGIN() statement the default origin for DNS data files is the domain named in the second field of the `primary` or `secondary` line of the `named.boot` file.

Resource Record Types

The most commonly used types of resource records are listed in Table 27-4. They are usually entered in the order shown in Table 27-4, but that is not a requirement.

TABLE 27-4 Commonly Used Resource Record Types

Type	Description
SOA	Start of authority
NS	Name server
A	Internet address (name to address)
PTR	Pointer (address to name)
CNAME	Canonical name (nickname)
TXT	Text information
WKS	Well-known services
HINFO	Host information
MX	Mail exchanger

SOA— Start of Authority

Code Example 27-1 shows the syntax of a start-of-authority (SOA) resource record.

CODE EXAMPLE 27-1 SOA Record Format

```
name class SOA origin person-in-charge ( serial number  
  refresh  
  retry  
  expire  
ttl)
```

The Start-Of-Authority record designates the start of a zone. The zone ends at the next SOA record. The SOA record fields are described below.

name

This field indicates the name of the zone. Note that the zone name must end with a trailing dot. For example: `doc.com.` is correct, while `doc.com` is wrong.

class

This field is the address class. For example, `IN` for Internet (the most commonly used class).

SOA

This field is the type of this resource record.

origin

This field is the name of the host where this data file resides. Note that this host name must end in a trailing dot. For example, `dnsmaster.doc.com.` is correct, but `dnsmaster.doc.com` is wrong.

person-in-charge

This field is the email address of the person responsible for the name server. For example, `kjd.nismaster.doc.com.` Again, this name must end with a trailing dot.

serial

This field is the version number of this data file. You must increment this number whenever you make a change to the data: secondary servers use the `serial` field to

detect whether the data file has been changed since the last time they copied the file from the master server.

refresh

This field indicates how often, in seconds, a secondary name server should check with the primary name server to see if an update is needed. For example, 7200 indicates a period of two hours.

retry

This field indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

expire

This field is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh.

ttl

This field is the default number of seconds to be used for the time-to-live field on resource records that don't have a *ttl* specified elsewhere.

There should only be one SOA record per zone. Code Example 27-2 is a sample SOA resource record.

CODE EXAMPLE 27-2 Sample SOA Resource Record

```
;name class SOA origin person-in-charge
doc.com. IN SOA dnsmaster.doc.com. root.nismaster.doc.com. (
    101 ;Serial
    7200 ;Refresh
    3600 ;Retry
    432000 ;Expire
    86400) ;Minimum )
```

NS—Name Server

Code Example 27-3 shows the syntax of a name-server (NS) resource record:

CODE EXAMPLE 27-3 NS Record Format

```
domainname [optional TTL] class NS name-server-name
```

The name-server record lists by name a server responsible for a given domain. The *name* field lists the domain that is serviced by the listed name server. If no *name* field is listed, then it defaults to the last name listed. One NS record should exist for each primary and secondary master server for the domain. Code Example 27-4 is a sample NS resource record.

CODE EXAMPLE 27-4 Sample NS Resource Record

```
;domainname [TTL] class NS nameserver  
doc.com 90000 IN NS sirius.doc.com.
```

A—Address

Code Example 27-5 shows the syntax of an address (A) resource record:

CODE EXAMPLE 27-5 Address Record Format

```
machinename [optional TTL] class A address
```

The address (A) record lists the address for a given machine. The *name* field is the host name, and the *address* is the IP address. One A record should exist for each address of the machine (in other words, routers, or gateways require at least two entries, a separate entry including the IP address assigned to each network interface).

CODE EXAMPLE 27-6 Sample Address Record

```
;machinename [TTL] class A address  
sirius IN A 123.45.6.1
```

HINFO—Host Information

Code Example 27-7 shows the syntax of a host-information (HINFO) resource record:

CODE EXAMPLE 27-7 HINFO Record Format

```
[optional name] [optional TTL] class HINFO hardware OS
```

The host-information resource record (HINFO) contains host-specific data. It lists the hardware and operating system that are running at the listed host. If you want to

include a space in the machine name or in the entry in the *hardware* field, you must surround the entry with quotes. The *name* field specifies the name of the host. If no name is specified, it defaults to the last *in.named* host. One HINFO record should exist for each host. Code Example 27-8 is a sample HINFO resource record.

CODE EXAMPLE 27-8 Sample HINFO Resource Record

```
;[name] [TTL] class HINFO hardware OS
IN HINFO Sparc-10 UNIX
```



Caution - Because the HINFO field provides information about the machines on your network, many sites consider it a security risk and no longer use it.

WKS—Well-Known Services

Code Example 27-9 shows the syntax of a well-known services (WKS) resource record:

CODE EXAMPLE 27-9 WKS Record Format

```
[Optional name] [TTL] class WKS address protocol-list-of-services
```

The Well-Known Services (WKS) record describes the well-known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in the *services* database. Only one WKS record should exist per protocol per address. Code Example 27-10 is an example of a WKS resource record.

CODE EXAMPLE 27-10 Sample WKS Resource Record

```
;[name] [TTL] class WKS address protocol-list-of-services
altair IN WKS 123.45.6.1 TCP ( smtp discard rpc
                                sftp uucp-
path systat daytime
                                netstat gotd nntp doc.com )
```



Caution - The WKS record is optional. For security reasons, most sites no longer provide this information.

CNAME—Canonical Name

Code Example 27-11 shows the syntax of a canonical-name (CNAME) resource record.

CODE EXAMPLE 27-11 CNAME Record Format

```
nickname [optional TTL] class CNAME canonical-name
```

The Canonical-Name Resource record (CNAME) specifies a nickname or alias for a canonical name. A nickname should be unique. All other resource records should be associated with the canonical name and not with the nickname. Do not create a nickname and then use it in other resource records. Nicknames are particularly useful during a transition period, when a machine's name has changed but you want to permit people using the old name to reach the machine. Nicknames can also be used to identify machines that serve some specific purpose such as a mail server. Code Example 27-12 is a sample CNAME resource record.

CODE EXAMPLE 27-12 Sample CNAME Resource Record

```
;nickname      [TTL] class CNAME canonical-name  
mailhost      IN CNAME antares.doc.com
```

PTR—Pointer Record

Code Example 27-13 shows the syntax for a pointer (PTR) resource record.

CODE EXAMPLE 27-13 PTR Record Format

```
special-name [optional TTL] class PTR-real-name
```

A pointer record allows special names to point to some other location in the domain. In the example, PTR's are used mainly in the `in-addr.arpa.` records for the translation of an address (the special name) to a real name. When translating an address, if the domain is fully qualified only the machine identification number need be specified. PTR names should be unique to the zone. The PTR records Code Example 27-14 sets up reverse pointers for the special `in-addr.arpa` domain.

CODE EXAMPLE 27-14 Sample PTR Resource Record

```
;special name [TTL] class PTR-real-name  
1 IN PTR sirius.doc.com.
```

MX—Mail Exchanger

Code Example 27-15 shows the syntax for a mail-exchanger (MX) resource record.

CODE EXAMPLE 27-15 MX Record Format

```
name [optional TTL] class MX preference-value mailer-exchanger
```

The mail-exchanger resource records are used to specify a machine that knows how to deliver mail to a domain or specific machines in a domain. There may be more than one MX resource record for a given name. In Code Example 27-16, `Seismo.CSS.GOV.` (note the fully qualified domain name) is a mail gateway that knows how to deliver mail to `Munnari.OZ.AU.` Other machines on the network cannot deliver mail directly to `Munnari.Seismo` and `Munnari` may have a private connection or use a different transport medium. The *preference-value* field indicates the order a mailer should follow when there is more than one way to deliver mail to a single machine. The value 0 (zero) indicates the highest preference. If there is more than one MX resource record for the same name, records may or may not have the same *preference* value.

You can use names with the wildcard asterisk (*) for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. In Code Example 27-16, all mail to hosts in domain `foo.com` is routed through `RELAY.CS.NET.` You do this by creating a wildcard resource record, which states that the mail exchanger for `*.foo.com` is `RELAY.CS.NET.` The asterisk will match any host or subdomain of `foo.com`, but it will not match `foo.com` itself.

CODE EXAMPLE 27-16 Sample MX Resource Record

```
;name [TTL] class MX preference mailer-exchanger  
Munnari.OZ.AU. IN MX 0 Seismo.CSS.GOV.  
foo.com. IN MX 10 RELAY.CS.NET.  
*.foo.com. IN MX 20 RELAY.CS.NET.
```

Solaris DNS BIND Implementation

For your convenience, the Solaris 2.6 release supplies a compiled version of Berkeley Internet Name Domain (BIND) version 4.9.4, Patch-Level 1. In compiling this software, options and choices were made to meet the needs of the greatest number of sites. If this pre-compiled version of BIND does not meet your requirements, you can recompile your own version of BIND from the publicly available source code.

In compiling the BIND version supplied with the Solaris 2.6 release, the following choices were made:

- *RFC1535*. Not implemented since because doing so would remove implicit search lists.
- *Inverse Queries*. Enabled because SunOS 4.x `nslookup` will not work without them.
- *Bogus Name Logging*. Logging of bogus name servers is not implemented because it produces too many unimportant messages.
- *Default Domain Name*. If the DNS domain name is not set in `/etc/resolv.conf`, or via the `LOCALDOMAIN` environment variable, `libresolv` derives it from the NIS or NIS+ domain name provided that the `/etc/nsswitch.conf` file contains `nisplus` or `nis` as the first element in the `hosts` line.
- *Utility Scripts*. The BIND utility scripts are not included in this Solaris release.
- *Test Programs*. The BIND test programs `dig`, `dnsquery`, and `host` are not included in this Solaris release because their purpose is similar to that of `nslookup` and `nstest`.

Administering DNS

This chapter describes how to administer the Domain Name System (DNS). For more detailed information, see *DNS and Bind* by Cricket Liu and Paul Albitz, (O'Reilly, 1992) and "Name Server Operations Guide for BIND", University of California, Berkeley.

- "Trailing Dots in Domain Names" on page 493
- "Modifying DNS Data Files" on page 494
- "Adding and Deleting Machines" on page 495
- "Adding Additional DNS Servers" on page 496
- "Creating DNS Subdomains" on page 497
- "DNS Error Messages and Problem Solving" on page 501

Trailing Dots in Domain Names

When working with DNS-related files, follow these rules regarding the trailing dot in domain names:

- Use a trailing dot in domain names in `hosts`, `hosts.rev`, `named.ca`, and `named.local` data files. For example, `sales.doc.com.` is correct.
- Do not use a trailing dot in domain names in `named.boot` or `resolv.conf` files. For example, `sales.doc.com` is correct.

Modifying DNS Data Files

Whenever you add or delete a host or make some other change in one of the DNS data files in the master DNS server or otherwise modify DNS data files, you must also:

- Change the serial number in the SOA resource record so the secondary servers modify their data accordingly (see “Changing the SOA Serial Number” on page 494).
- Inform `in.named` on the master server that it should reread the data files and update its internal database (see).

Changing the SOA Serial Number

Every DNS database file begins with a Start of Authority (SOA) resource record. Whenever you alter any data in a DNS database file, you must increment the SOA serial number by one integer.

For example, if the current SOA Serial Number in a data file is 101, and you make a change to the file’s data, you must change 101 to 102. If you fail to change the SOA serial number, the domain’s secondary servers will not update their copy of the database files with the new information and the primary and secondary servers will become out of synch.

A typical SOA record of a sample `hosts` file looks like this:

```
; sample hosts file
@ IN      SOA  nismaster.doc.com. root.nismaster.doc.com. (
    109 ; Serial
    10800 ; Refresh
                                1800 ; Retry
    3600000 ; Expire
    86400 ) ; Minimum
```

Thus, if you made a change to this `hosts` file, you would change 109 to 110. The next time you change the file, you would change 110 to 111.

Forcing `in.named` to Reload DNS Data

When `in.named` successfully starts, the daemon writes its process ID to the file `/etc/named.pid`. To have `in.named` reread `named.boot` and reload the database, enter:

```
# kill -HUP `cat /etc/named.pid`
```

This will eliminate all previously cache, and the caching process will start over again.



Caution - Do not attempt to run `in.named` from `inetd`. This will continuously restart the name server and defeat the purpose of having a cache.

Adding and Deleting Machines

When you add or delete a machine, always make your changes in the data files stored on your primary DNS server. Do not make changes or edit the files on your secondary servers because those will be automatically updated from the primary server based on your changing the SOA serial number.

Adding a Machine

To add a machine to a DNS domain, you set the new machine up as a DNS client and then add records for the new machine to the appropriate `hosts` and `hosts.rev` files.

For example, to add the host `rigel` to the `doc.com` domain:

1. **Create a `/etc/resolv.conf` file on `rigel`.**
2. **Add `dns` to the `hosts` line of `rigel`'s `/etc/nsswitch.conf` file**
(See “DNS and Internet Access” on page 20.)
3. **Add an address (A) record for `rigel` to the primary server's `hosts` file.**

For example:

```
rigel IN A 123.45.6.112
```

4. **Add any additional optional records for `rigel` to the primary server's `hosts` file.**

Optional records could include:

- Alias (CNAME)
- Mail exchange (MX)
- Well known services (WKS)

- Host information (HINFO)
5. **Add a PTR record for `rigel` to the `hosts.rev` file.**
 6. **Increment the SOA serial number in the primary server's `hosts` and `hosts.rev` files.**
 7. **Reload the server's data.**
Either reboot the server or enter:

```
# kill -HUP `cat /etc/named.pid`
```

These steps are explained in more detail in *Solaris Naming Setup and Configuration Guide*.

Removing a Machine

To remove a machine from a DNS domain:

1. **Remove `dns` from the `hosts` line of the machine's `nsswitch.conf` file.**
2. **Remove the machine's `/etc/resolv.conf` file.**
3. **Delete the records for that machine from the primary server's `hosts` and `hosts.rev` files.**
4. **If the machine has CNAME records pointing to it, those CNAME records must also be deleted from the `hosts` file.**
5. **Set up replacements for services supported by the removed machine.**
If the machine is a primary server, mail host, or host for any other necessary process or service, you must take whatever steps are necessary to set up some other machine to perform those services.

Adding Additional DNS Servers

You can add primary and secondary servers to your network. To add a DNS server:

1. **Set the server up as a DNS client.**
See “Adding a Machine” on page 495.

2. **Set up the server's boot file.**
3. **Set up the server's `named.ca` file.**
4. **Set up the server's `hosts` file.**
5. **Set up the server's `hosts.rev` file.**
6. **Set up the server's `named.local` file.**
7. **Initialize the server.**
8. **Test the server.**

These steps are explained in more detail in *Solaris Naming Setup and Configuration Guide*.

Creating DNS Subdomains

As your network grows you may find it convenient to divide it into one or more DNS subdomains. (See “Introducing the DNS Namespace” on page 468 for a discussion of DNS domain hierarchy and structure.)

When you divide your network into a parent domain and one or more subdomains, you reduce the load on individual DNS servers by distributing responsibility across multiple domains. In this way you can improve network performance. For example, suppose there are 900 machines on your network and all of them are in one domain. In this case, one set of DNS servers composed of a primary and additional secondary and caching-only servers have to support 900 machines. If you divide this network into a parent domain and two subdomain, each with 300 machines, then you have three sets of primary and secondary servers each responsible for only 300 machines.

By dividing your network into domains that match either your geographic or organizational structure (or both), the DNS domain names indicate where a given machine or email address fits into your structure. For example, `rigel@alameda.doc.com` implies that the machine `rigel` is located at your Alameda site, and the email address `barnum@sales.doc.com` implies that the user `barnum` is part of your Sales organization.

Dividing your network into multiple domains requires more set up work than keeping everything in one domain, and you have to maintain the delegation data that ties your domains together. On the other hand, when you have multiple

domains, you can distribute domain maintenance tasks among different administrators or teams, one for each domain.

Planning Your Subdomains

Here are some points to consider before dividing your network into a parent and one or more subdomains:

- *How many subdomains?* The more subdomains you create, the more initial set up work you have to do and the more ongoing coordination work for the administrators in the parent domain. The more subdomains, the more delegation work for the servers in the parent domain. On the other hand, fewer domains mean larger domains, and the larger a domain is the more server speed and memory is required to support it.
- *How to divide your network?* You can divide your network into multiple domains any way you want. The three most common methods are by organizational structure where you have separate subdomains for each department or division (sales, research, manufacturing, etc.); by geography where you have separate subdomains for each site; or by network structure where you have separate subdomains for each major network component. The most important rule to remember is that administration and use will be easier if your domain structure follows a consistent, logical, and self-evident pattern.
- *Consider the future.* The most confusing domain structures are those that grow over time with subdomains added haphazardly as new sites and departments are created. To the degree possible, try to take future growth into account when designing your domain hierarchy. Also take into account stability. It is best to base your subdomains on what is most stable. For example, if your geographic sites are relatively stable but your departments and divisions are frequently reorganized, it is probably better to base your subdomains on geography rather than organizational function. On the other hand, if your structure is relatively stable but you frequently add or change sites, it is probably better to base your subdomains on your organizational hierarchy.
- *Wide area network links.* When a network spans multiple sites connected via modems or leased lines, performance will be better and reliability greater if your domains do not span such Wide Area Network (WAN) links. In most cases, WAN links are slower than contiguous network connections and more prone to failure. When servers have to support machines that can only be reached over a WAN link, you increase the network traffic funneling through the slower link, and if there is a power failure or other problem at one site, it could affect the machines at the other sites. (The same performance and reliability considerations apply to DNS zones. As a general rule of thumb, it is best if zones do not span WAN links.)
- *NIS+ name service.* If your enterprise-level name service is NIS+, administration will be easier if your DNS and NIS+ domain and subdomain structures match.

- *Subdomain names.* To the degree possible, it is best to establish and follow a consistent policy for naming your subdomains. When domain names are consistent, it is much easier for users to remember and correctly specify them. Keep in mind that domain names are an important element in all of your DNS data files and that changing a subdomain name requires editing every file in which the old name appears. Thus, it is best to choose subdomain names that are stable and unlikely to need changing. You can use either full words, such as `manufacturing`, or abbreviations, such as `manf`, as subdomain names, but it will confuse users if some subdomains are named with abbreviations and others with full names. If you decide to use abbreviations, use enough letters to clearly identify the name because short cryptic names are hard to use and remember. Do not use reserved top-level Internet domain names as subdomain names. This means that names like `org`, `net`, `com`, `gov`, `edu`, and any of the two-letter country codes such as `jp`, `uk`, `ca`, and `it` should never be used as a subdomain name.

Setting Up a Subdomain

In most cases, new subdomains are usually created from the start with a new network and machines, or split off from an existing domain. The process is essentially similar in both cases.

Once you have planned your new subdomain, follow these steps to set it up:

- 1. Make sure all of the machines in the new subdomain are properly set up as DNS clients.**

If you are carving a new subdomain out of an existing domain, most of the machines are probably already set up of DNS clients. If you are building a new subdomain from scratch (or adding new machines to an existing network) you must install properly configured `resolv.conf` and `nsswitch.conf` files on each machine as described in *Solaris Naming Setup and Configuration Guide*.

- 2. Install properly configured boot and DNS data files on the subdomain's primary master server.**

Install the following files on each server (see *Solaris Naming Setup and Configuration Guide* for details):

- `/etc/named.boot`.
- `/var/named/named.ca`.
- `/var/named/hosts`.
- `/var/named/hosts.rev`.
- `/var/named/named.local`.

Note that the server host files must have an Address (A) record, any necessary CNAME records for each machine in the subdomain and the server `hosts.rev` files must have a pointer (PTR) record for each machine in the subdomain. Optional HINFO and WKS records can also be added.

- 3. If you are splitting an existing domain, remove the records for the machines in the new subdomain from the parent domain's master server `hosts` and `hosts.rev` files.**

This requires deleting the A records for the machines that are now in the new subdomain from the `hosts` files of the old domain's servers, and also deleting the PTR records for those machines from the old domain's `hosts.rev` files. Any optional HINFO and WKS records for the moved machines should also be deleted.

- 4. If you are splitting an existing domain, add the new subdomain name to CNAME records in the parent domain's master server `hosts` and file.**

For example, suppose you are using the machine `aldebaran` as a fax server and it had the following CNAME record in the `hosts` file of the parent domain's servers:

```
faxserver IN CNAME aldebaran
```

In addition to creating a new `faxserver` CNAME record for `aldebaran` in the `hosts` file of the new subdomain's master server, you would also have to change this CNAME record in the parent domain's `hosts` file to include `aldebaran`'s subdomain as shown below:

```
faxserver IN CNAME aldebaran.manf.doc.com
```

- 5. Add NS records for the new subdomain's servers to the parent domain's `hosts` file.**

For example, suppose that your parent domain is `doc.com` and you are creating a new `manf.doc.com` subdomain with the machine `rigel` as `manf`'s primary master server and `aldebaran` as the secondary master server. You would add the following records to the `hosts` file of `doc.com`'s primary master server:

```
manf.doc.com 99999 IN NS rigel.manf.doc.com
              99999 IN NS aldebaran.manf.doc.com
```

- 6. Add A records for the new subdomain's servers to the parent domain's `hosts` file.**

Continuing with the above example, you would add the following records to the `hosts` file of `doc.com`'s primary master server:

rigel.manf.doc.com	99999	IN	A	1.22.333.121
aldebaran.manf.doc.com	99999	IN	A	1.22.333.136

7. Start up named on the subdomain's servers.

```
# /usr/sbin/in.named
```

Instead of running `in.named` from the command line, you can reboot. See *Solaris Naming Setup and Configuration Guide* for details.

DNS Error Messages and Problem Solving

See Appendix A, and Appendix B, for DNS problem solving and error message information.

PART **VII** Appendices

This part of the manual provides reference material.

- Appendix A
- Appendix B
- Appendix C
- Appendix D

Problems and Solutions

This appendix describes some of the problems you may encounter while administering Solaris 2.6 release namespaces and how to correct them.

- “Troubleshooting NIS+” on page 506
- “NIS+ De-Bugging Options” on page 506
- “NIS+ Administration Problems” on page 507
- “NIS+ Database Problems” on page 511
- “NIS+ and NIS Compatibility Problems” on page 512
- “NIS+ Object Not Found Problems” on page 514
- “NIS+ Ownership and Permission Problems” on page 517
- “NIS+ Security Problems” on page 519
- “NIS+ Performance and System Hang Problems” on page 528
- “NIS+ System Resource Problems” on page 532
- “NIS+ User Problems” on page 533
- “Other NIS+ Problems” on page 535
- “NIS Problems and Solutions” on page 537
- “DNS Problems and Solutions” on page 544
- “FNS Problems and Solutions” on page 549

Troubleshooting NIS+

In this appendix, 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 first. If the problem is simple, or specific to a single error message, its solution is usually described in Appendix B.

NIS+ De-Bugging Options

The `NIS_OPTIONS` environment variable can be set to control various NIS+ debugging options.

Options are specified after the `NIS_OPTIONS` command separated by spaces with the option set enclosed in double quotes. Each option has the format `name=value`. Values can be integers, character strings, or filenames depending on the particular option. If a value is not specified for an integer value option, the value defaults to 1.

`NIS_OPTIONS` recognizes the following options:

TABLE A-1 `NIS_OPTIONS` Options and Values

Option	Values	Actions
<code>debug_file</code>	<i>filename</i>	Directs debug output to specified file. If this option is not specified, debug output goes to <code>stdout</code> .
<code>debug_bind</code>	<i>Number</i>	Displays information about the server selection process.
<code>debug_rpc</code>	<i>1 or 2</i>	If the value is 1, displays RPC calls made to the NIS+ server and the RPC result code. If the value is 2, displays both the RPC calls and the contents of the RPC and arguments and results.
<code>debug_calls</code>	<i>Number</i>	Displays calls to the NIS+ API and the results that are returned to the application.

TABLE A-1 NIS_OPTIONS Options and Values (continued)

Option	Values	Actions
pref_srvr	String	Specifies preferred servers in the same format as that generated by the nisprefadm command (see Table 14-1). This will over-ride the preferred server list specified in nis_cachemgr.
pref_type	String	Not currently implemented.

For example, (assuming that you are using a C-Shell):

- To display many debugging messages you would enter:

```
setenv NIS_OPTIONS ``debug_calls=2 debug_bind debug_rpc``
```

- To obtain a simple list of API calls and store them in the file /tmp/CALLS you would enter:

```
setenv NIS_OPTIONS ``debug_calls debug_file=/tmp/CALLS``
```

NIS+ Administration Problems

This section describes problems that may be encountered in the course of routine NIS+ namespace administration work. Common symptoms include:

- ``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

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.
- You attempted to link a table entry.
- 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 as described in "Removing a Directory" on page 193. Keep in mind that you must first remove the directory's `org_dir` and `groups_dir` subdirectories before you remove the directory itself.

Lack of Disk Space

Lack of sufficient disk space will cause a variety of different error messages. (See "Insufficient Disk Space" on page 532 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 /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 `altair` might be a client of the `subdoc.doc.com.` domain. If the master server of the `subdoc.doc.com.` subdomain is the machine

sirius, then `sirius` is a client of the `doc.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 `altair` machine is `alladin.subdoc.doc.com.` The fully qualified name of the `sirius` machine is `sirius.doc.com.` The name `sirius.subdoc.doc.com.` is wrong and will cause an error because `sirius` is a client of `doc.com.`, not `subdoc.doc.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.

Removal or Disassociation of NIS+ Directory from Replica Fails

When removing or disassociating a directory from a replica server you must first remove the directory's `org_dir` and `groups_dir` subdirectories before removing the directory itself. After each subdirectory is removed, you must run `nisping` on the parent directory of the directory you intend to remove. (See "Removing a Directory " on page 193.)

If you fail to perform the `nisping` operation, the directory will not be completely removed or disassociated.

If this occurs, you need to perform the following steps to correct the problem:

1. Remove `/var/nis/rep/org_dir` on the replica.
2. Make sure that `org_dir.domain` does not appear in `/var/nis/rep/serving_list` on the replica.
3. Perform a `nisping` on *domain*.
4. From the master server, run `nisrmdir -f replica_directory`.

If the replica server you are trying to dissociate is down or out of communication, the `nisrmdir -s` command will return a `Cannot remove replica name: attempt to remove a non-empty table` error message.

In such cases, you can run `nisrmdir -f -s replicaname` on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -s` to dissociate an

out-of-communication replica, you *must* run `nisrmdir -f -s` *again* as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s replicaname` when the replica is back in service, the old out-of-date information left on the replica could cause problems.

NIS+ Database Problems

This section covers problems related to the namespace database and tables. Common symptoms include:

- 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`
- `rpc.nisd` fails.

See also “NIS+ Ownership and Permission Problems” on page 517.

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 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.

Note - If you started `rpc.nisd` with the `-B` option, you must also kill the `rpc.nisd_resolv` daemon.

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.

rpc.nisd Fails

If an NIS+ table is too large, `rpc.nisd` may fail.

Diagnosis:

Use `nislsls` to check your NIS+ table sizes. Tables larger than 7k may cause `rpc.nisd` to fail.

Solution

Reduce the size of large NIS+ tables. Keep in mind that as a naming service NIS+ is designed to store references to objects, not the objects themselves.

NIS+ and NIS Compatibility Problems

This section describes problems having to do with NIS compatibility with NIS+ and earlier systems and the switch configuration file. Common symptoms include:

- 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 2.6 release 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.

NIS+ Object Not Found Problems

This section describes problem in which NIS+ was unable to find some object or principal. Common symptoms include:

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 -lin` in the appropriate domain to check that the object exists.

Lagging or Out-of-Sync Replica

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 Release 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`. Starting with Solaris Release 2.5, the two files were 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 Release 2.5, the *content* of the files were also changed and they are not backward compatible with Solaris Release 2.4 or earlier. Thus, if you rename either the directories or the files to match the Solaris Release 2.4 patterns, the files will not work with *either* the Solaris Release 2.4 or the Solaris Release 2.5 or later versions 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 `nisls` 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 `nisl`s 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.)

Links To of From Table Entries Do Not Work

You cannot use the `nisl`n command (or any other command) to create links between entries in tables. NIS+ commands do not follow links at the entry level.

NIS+ Ownership and Permission Problems

This section describes problems related to user ownership and permissions. Common symptoms include:

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 9,) or add you to a group that does have the required permissions (as described in Chapter 11).

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 524 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 `passwd` 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 domainname.`

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 `saladin` is granted namespace read-only permissions. Then a machine named `saladin` is added to the domain. The user `saladin` will no longer be able to perform any namespace operations requiring any sort of permission, and the root user of the machine `saladin` will only have read-only permission in the namespace.

Symptoms:

- The user or machine gets "permission denied" type 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 524.

NIS+ Security Problems

This section describes common password, credential, encryption, and other security-related problems.

Security Problem Symptoms

Error messages with operative clauses such as:

- Authentication error
- Authentication denied
- Cannot get public key
- Chkey failed
- Insufficient permission to
- Login incorrect
- Keyserver fails to encrypt
- No public key
- Permission denied
- "Password [problems]"

User or root unable to perform any namespace operations or tasks. (See also “NIS+ Ownership and Permission Problems” on page 517.)

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 also 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 160 and “Unlocking a Password ” on page 161.
- The password has been locked because the user has exceeded an inactivity maximum See “Specifying Maximum Number of Inactive Days” on page 167.
- The password has expired. See “Password Privilege Expiration” on page 165.

See Chapter 10 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 10 for general information on passwords.

- See “Setting a Password Age Limit ” on page 162.
- See “Password Privilege Expiration” on page 165.

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-2 lists all the objects, tables, and files that store credential-related information and how to reset it.

TABLE A-2 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.
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 7, 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. 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 `doc.com.` root domain, and is made the master server of the `sales.doc.com.` domain.

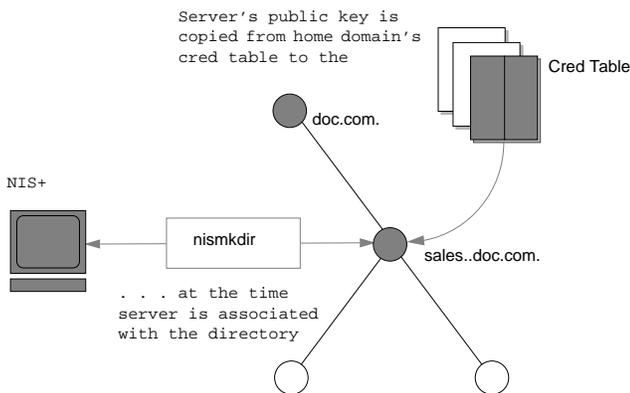


Figure A-1 Public Key is Propagated to Directory Objects

Its public key is copied from the `cred.org_dir.doc.com.` domain and placed in the `sales.doc.com.` directory object. This can be verified with the `niscat -o sales.doc.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-3 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>	"The <code>nisping</code> Command " on page 201
Directory object of domain supported by server	<code>nisupdkeys</code>	"The <code>nisupdkeys</code> Command" on page 112
<code>NIS_COLDSTART</code> file of clients	<code>nisinit -c</code>	"The <code>nisinit</code> Command " on page 197
<code>NIS_SHARED_DIRCACHE</code> file of clients	<code>nis_cachemgr</code>	"The <code>nis_cachemgr</code> Command " on page 199

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 `nislsls`. 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 `nislsls` 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 or superuser.* Run `keylogout -f` followed by `keylogin -r`.

Keyserver Failure

The `keyserver` 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 99.
- The `keyserver` daemon is not running. Use the `ps` command to see if `keyserver` is running. If it is not running, restart it and then do a `keylogin`.
- While `keyserver` 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+ and Login Passwords 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.

Note - 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 106.)

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 an NIS+ Principal” on page 107.

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 -pto` 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 `nisls -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:

Note - For security reasons, you should not have User ID 0 listed 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.

NIS+ Performance and System Hang Problems

This section describes common slow performance and system hang problems.

Performance Problem 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 57 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.

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.)

NIS+ System Resource Problems

This section describes problems having to do with lack of system resources such as memory, disk space, and so forth.

Resource Problem Symptoms

Error messages with operative clauses such as:

- No memory
- Out of disk space
- “Cannot [do something] with log” type messages
- 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 `logfile`s. `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.

NIS+ User Problems

This section describes NIS+ problems that a typical user might encounter.

User Problem 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 `passwd` 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 “1” the same as the numeral “1.”

- “*Login incorrect*” type message. For causes other than simply mistyping the password, see “Login Incorrect Message ” on page 520.
- The user’s password privileges have expired (see “Password Privilege Expiration” on page 165).
- An inactivity maximum has been set for this user, and the user has passed it (see “Specifying Maximum Number of Inactive Days” on page 167).
- 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 149 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 534).

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 “Setting Minimum Password Life ” on page 163.
- The password is locked or expired. See “Login Incorrect Message ” on page 520 and “Password Locked, Expired, or Terminated” on page 520.

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, 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 resyncing

(These messages are generated by, or in conjunction with, the NIS+ error code constant: `NIS_DUMPLATER` one replica is already resyncing.)

These messages indicate that there was some other problem:

- `replica_update`: error result was ...
- `replica_update`: nis dump result `nis_perror` error string
- `rootreplica_update`: update failednis dump result `nis_perror` 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.

NIS Problems and Solutions

This section explains how to resolve problems encountered on networks running NIS. It covers problems seen on an NIS client and those seen on an NIS server.

Before trying to debug an NIS server or client, review Chapter 17, which explains the NIS environment. Then look for the subheading in this section that best describes your problem.

Symptoms:

Common symptoms of NIS binding problems include:

- Messages saying that `ypbind` can't find or communicate with a server.
- Messages saying server not responding.
- Messages saying NIS is unavailable
- Commands on a client limp along in background mode or function much slower than normal.
- Commands on a client hang. Sometimes commands hang even though the system as a whole seems fine and you can run new commands.
- Commands on a client crash with obscure messages, or no message at all.

NIS Problems Affecting One Client

If only one or two clients are experiencing symptoms that indicate NIS binding difficulty, the problems probably are on those clients. If many NIS clients are failing to bind properly, the problem probably exists on one or more of the NIS servers (see "NIS Problems Affecting Many Clients" on page 541).

`ypbind` Not Running on Client

One client has problems, but other clients on the same subnet are operating normally. On the problem client, run `ls -l` on a directory such as `/usr` that contains files owned by many users, including some not in the client `/etc/passwd` file. If the resulting display lists file owners who are not in the local `/etc/passwd` as numbers, rather than names, this indicates that NIS service is not working on the client.

These symptoms usually mean that the client `ypbind` process is not running. Run `ps -e` and check for `ypbind`. If you do not find it, log in as superuser and start `ypbind` by typing:

```
client# /usr/lib/netsvc/yp/ypstart
```

Missing or Incorrect Domain Name

One client has problems, the other clients are operating normally, but `ypbind` is running on the problem client. The client may have an incorrectly set domain.

On the client, run the `domainname` command to see which domain name is set.

```
Client#7 domainname neverland.com
```

Compare the output with the actual domain name in `/var/yp` on the NIS master server. The actual NIS domain is shown as a subdirectory in the `/var/yp` directory.

```
Client#7 ls /var/yp...
-rwxr-xr-x 1 root Makefile
drwxr-xr-x 2 root binding
drwx----- 2 root doc.com
...
```

If the domain name returned by running `domainname` on a machine is not the same as the server domain name listed as a directory in `/var/yp`, the domain name specified in the machine's `/etc/defaultdomain` file is incorrect. Log in as superuser and correct the client's domain name in the machine's `/etc/defaultdomain` file. This assures that the domain name is correct every time the machine boots. Now reboot the machine.

Note - The domain name is case-sensitive.

Client Not Bound to Server

If your domain name is set correctly, `ypbind` is running, and commands still hang, then make sure that the client is bound to a server by running the `ypwhich` command. If you have just started `ypbind`, then run `ypwhich` several times (typically, the first one reports that the domain is not bound and the second succeeds normally).

No Server Available

If your domain name is set correctly, `ypbind` is running, and you get messages indicating that the client cannot communicate with a server, this may indicate a number of different problems:

- Does the client have a `/var/yp/binding/domainname/ypservers` file containing a list of servers to bind to? If not, run `ypinit -c` and specify in order of preference the servers that this client should bind to.

- If the client does have a `/var/yp/binding/domainname/ypservers` file, are there enough servers listed in it if one or two become unavailable? If not, add additional servers to the list by running `ypinit -c`.
- If none of the servers listed in the client's `ypservers` file are available, the client searches for an operating server using broadcast mode. If there is a functioning server on the client's subnet, the client will find it (though performance may be slowed during the search). If there are no functioning servers on the client's subnet can solve the problem in several ways:
 - If the client has no server on the subnet and no route to one, you can install a new slave server on that subnet.
 - You can make sure your routers are configured to pass broadcast packets so that the client can use broadcast to find a server on another subnet. You can use the `netstat -r` command to verify the route.
 - If there should be a route, but it is not working, make sure that the route daemon `in.routed/in.rdisc` is running. If it is not running, start it.

Note - For reasons of security and administrative control it is preferable to specify the servers a client is to bind to in the client's `ypservers` file rather than have the client search for servers through broadcasting. Broadcasting ties up the network, slows the client, and prevents you from balancing server load by listing different servers for different clients.

- Do the servers listed in a clients `ypservers` file have entries in the `/etc/hosts` file? If not, add the servers to the NIS maps hosts input file and rebuild your maps by running `ypinit -c` or `ypinit -s` as described "Working With NIS Maps" on page 304.
- Is the `/etc/nsswitch.conf` file set up to consult the machine's local `hosts` file in addition to NIS? See Chapter 2 for more information on the switch.
- Is the `/etc/nsswitch.conf` file set up to consult files first for services and `rpc`?

ypwhich Displays Are Inconsistent

When you use `ypwhich` several times on the same client, the resulting display varies because the NIS server changes. This is normal. The binding of the NIS client to the NIS server changes over time when the network or the NIS servers are busy. Whenever possible, the network stabilizes at a point where all clients get acceptable response time from the NIS servers. As long as your client machine gets NIS service, it does not matter where the service comes from. For example, an NIS server machine may get its own NIS services from another NIS server on the network.

When Server Binding is Not Possible

In extreme cases where local server binding is not possible, use of the `ypset` command may temporarily allow binding to another server, if available, on another network or subnet. However, in order to use the `-ypset` option, `ypbind` must be started with either the `-ypset` or `-ypsetme` options.

Note - For security reasons, the use of the `-ypset` and `-ypsetme` options should be limited to debugging purposes under controlled circumstances. Use of the `-ypset` and `-ypsetme` options can result in serious security breaches because while they are operative anyone can then alter server bindings causing trouble for others and permitting unauthorized access to sensitive data. If you must start `ypbind` with these options, once you have fixed the problem you should kill `ypbind` and restart it again without those options.

`ypbind` Crashes

If `ypbind` crashes almost immediately each time it is started, look for a problem in some other part of the system. Check for the presence of the `rpcbind` daemon by typing:

```
% ps -ef | grep rpcbind
```

If `rpcbind` is not present or does not stay up or behaves strangely, consult your RPC documentation.

You may be able to communicate with `rpcbind` on the problematic client from a machine operating normally. From the functioning machine, type:

```
% rpcinfo client
```

If `rpcbind` on the problematic machine is fine, `rpcinfo` produces the following output:

```
program version netid address service owner
...
100007 2 udp 0.0.0.0.2.219 ypbind superuser
100007 1 udp 0.0.0.0.2.219 ypbind superuser
100007 1 tcp 0.0.0.0.2.220 ypbind superuser
100007 2 tcp 0.0.0.0.128.4 ypbind superuser
100007 2 ticotsord \000\000\020H ypbind superuser
100007 2 ticots \000\000\020K ypbind superuser
...
```

Your machine will have different addresses. If they are not displayed, `ypbind` has been unable to register its services. Reboot the machine and run `rpcinfo` again. If the `ypbind` processes are there and they change each time you try to restart

`/usr/lib/netsvc/yp/ypbind`, reboot the system, even if the `rpcbind` daemon is running.

NIS Problems Affecting Many Clients

If only one or two clients are experiencing symptoms that indicate NIS binding difficulty, the problems probably are on those clients (see “NIS Problems Affecting One Client” on page 537). If many NIS clients are failing to bind properly, the problem probably exists on one or more of the NIS servers.

Network or Servers are Overloaded

NIS can hang if the network or NIS servers are so overloaded that `ypserv` cannot get a response back to the client `ypbind` process within the time-out period.

Under these circumstances, every client on the network experiences the same or similar problems. In most cases, the condition is temporary. The messages usually go away when the NIS server reboots and restarts `ypserv`, or when the load on the NIS servers or network itself decreases.

Server Malfunction

Make sure the servers are up and running. If you are not physically near the servers, use the `ping` command.

NIS Daemons Not Running

If the servers are up and running, try to find a client machine behaving normally, and run the `ypwhich` command. If `ypwhich` does not respond, kill it. Then log in as `root` on the NIS server and check if the NIS `ypbind` process is running by entering:

```
# ps -e | grep yp
```

Note - Do not use the `-f` option with `ps` because this option attempts to translate user IDs to names which causes more name service lookups that may not succeed.

If either the `ypbind` or `ypserv` daemons are not running, kill them and then restart them by entering:

```
# /usr/lib/netsvc/yp/ypstop  
# /usr/lib/netsvc/yp/ypstart
```

If both the `ypserv` and `ypbind` processes are running on the NIS server, type:

```
# ypwhich
```

If `ypwhich` does not respond, `ypserv` has probably hung and should be restarted. While logged in as `root` on the server, kill `ypserv` and restart it by typing:

```
# /usr/lib/netsvc/yp/ypstop  
# /usr/lib/netsvc/yp/ypstart
```

Servers Have Different Versions of an NIS Map

Because NIS propagates maps among servers, occasionally you may find different versions of the same map on various NIS servers on the network. This version discrepancy is normal and acceptable if the differences do not last for more than a short time.

The most common cause of map discrepancy is that something is preventing normal map propagation. For example, an NIS server or router between NIS servers is down. When all NIS servers and the routers between them are running, `ypxfr` should succeed.

If the servers and routers are functioning properly, check the following:

- Log `ypxfr` output (see “Logging `ypxfr` Output” on page 542).
- Check the control files (see “Check the `crontab` File and `ypxfr` Shell Script” on page 543).
- Check the `ypservers` map on the master (see “Check the `ypservers` Map” on page 543).

Logging `ypxfr` Output

If a particular slave server has problems updating maps, log in to that server and run `ypxfr` interactively. If `ypxfr` fails, it tells you why it failed, and you can fix the problem. If `ypxfr` succeeds, but you suspect it has occasionally failed, create a log file to enable logging of messages. To create a log file, enter:

```
ypslave# cd /var/yp  
ypslave# touch ypxfr.log
```

This creates a `ypxfr.log` file that saves all output from `ypxfr`.

The output resembles the output `ypxfr` displays when run interactively, but each line in the log file is time stamped. (You may see unusual ordering in the time-stamps. That is okay—the time-stamp tells you when `ypxfr` started to run. If copies of `ypxfr` ran simultaneously but their work took differing amounts of time, they may

actually write their summary status line to the log files in an order different from that which they were invoked.) Any pattern of intermittent failure shows up in the log.

Note - When you have fixed the problem, turn off logging by removing the log file. If you forget to remove it, it continues to grow without limit.

Check the crontab File and ypxfr Shell Script

Inspect the root crontab file, and check the `ypxfr` shell script it invokes. Typographical errors in these files can cause propagation problems. Failures to refer to a shell script within the `/var/spool/cron/crontabs/root` file, or failures to refer to a map within any shell script can also cause errors.

Check the ypservers Map

Also, make sure that the NIS slave server is listed in the `ypservers` map on the master server for the domain. If it is not, the slave server still operates perfectly as a server, but `yppush` does not propagate map changes to the slave server.

Work Around

If the NIS slave server problem is not obvious, you can work around it while you debug using `rcp` or `ftp` to copy a recent version of the inconsistent map from any healthy NIS server. For instance, here is how you might transfer the problem map:

```
ypslave# rcp ypmaster:/var/yp/mydomain/map.* /var/yp/mydomain
```

Here the `*` character has been escaped in the command line, so that it will be expanded on `ypmaster`, instead of locally on `ypslave`.

ypserv Crashes

When the `ypserv` process crashes almost immediately, and does not stay up even with repeated activations, the debug process is virtually identical to that described in “`ypbind` Crashes” on page 540. Check for the existence of the `rpcbind` daemon as follows:

```
ypserver% ps -e | grep rpcbind
```

Reboot the server if you do not find the daemon. Otherwise, if the daemon is running, type the following and look for similar output:

```
% rpcinfo -p ypserver
program vers proto port service
100000 4 tcp 111 portmapper
100000 3 tcp 111 portmapper
100068 2 udp 32813 cmsd
...
100007 1 tcp 34900 ypbind
100004 2 udp 731 ypserv
100004 1 udp 731 ypserv
100004 1 tcp 732 ypserv
100004 2 tcp 32772 ypserv
```

Your machine may have different port numbers. The four entries representing the ypserv process are:

```
100004 2 udp 731 ypserv
100004 1 udp 731 ypserv
100004 1 tcp 732 ypserv
100004 2 tcp 32772 ypserv
```

If they are not present, and ypserv is unable to register its services with rpcbind, reboot the machine. If they are present, deregister the service from rpcbind before restarting ypserv. To deregister the service from rpcbind, on the server type:

```
# rpcinfo -d number 1
# rpcinfo -d number 2
```

Where *number* is the ID number reported by rpcinfo (100004, in the example above).

DNS Problems and Solutions

This section describes some common DNS problems and how to solve them.

Clients Can Find Machine by Name but Server Cannot

Symptoms:

DNS clients can find machines by either IP address or by host name, but the server can only find machines by their IP addresses.

Probable cause and solution:

This is most likely caused by omitting `dns` from the `hosts` line of the server's `nsswitch.conf` file. For example, a *bad* `hosts` line might look like this:
`hosts: files`

When using DNS you must include `dns` in the `hosts` record of every machine's `nsswitch.conf` file. For example:

```
hosts: dns nisplus [NOTFOUND=return] files
```

or

```
hosts: nisplus dns [NOTFOUND=return] files
```

Changes Do Not Take Effect or Are Erratic

Symptom:

You add or delete machines or servers but your changes are not recognized or do not take effect. Or in some instances the changes are recognized and at other times they are not in effect.

Probable cause:

The most likely cause is that you forgot to increment the SOA serial number on the primary master server after you made your change. Since there is no new SOA number, your secondary servers do not update their data to match that of the primary so they are working with the old, unchanged data files.

Another possible cause is that the SOA serial number in one or more of the primary data files was set to a value lower than the corresponding serial number on your secondary servers. This could happen, for example, if you deleted a file on the primary and then recreated it from scratch using an input file of some sort.

A third possible cause is that you forgot to send a HUP signal to the primary server after making changes to the primary's data files.

Diagnosis and solution:

First, check the SOA serial numbers in the data file that you changed and the corresponding file on the secondary server.

- If the SOA serial number in the primary file is equal to, or less than, the serial number in the secondary file, increase the serial number on the primary's file so that it is greater than the number in the secondary file. For example, if the SOA number in both files is 37, change the number in the primary's file to 38. The next time the secondary checks with the primary, it will load the new data. (There are utilities that can force a primary to immediately transfer data to the secondaries, if you have one of these utilities you can update the secondary without waiting for it to check the primary.)
- Review the `syslog` output for the most recent named `nnnn` restarted or named `nnn` reloading `nameserver` entry. If the timestamp for that entry is before the

time you finished making changes to the file, either reboot the server or force it to read the new data as explained in “Forcing `in.named` to Reload DNS Data” on page 494.

DNS Client Cannot Lookup “Short” Names

Symptoms:

Client can lookup fully qualified names but not short names.

Possible cause and solution:

Check the client’s `/etc/resolv.conf` file for spaces at the end of the domain name. No spaces or tabs are allowed at the end of the domain name.

Reverse Domain Data Not Correctly Transferred to Secondary

While zone domain-named data is properly transferred from the zone primary master server to a zone secondary server, the reverse domain data is not being transferred. In other words, the `host.rev` file on the secondary is not being properly updated from the primary.

Possible causes:

Syntax error in the secondary server’s boot file.

Diagnosis and Solution:

Check the secondary server’s boot file. Make sure that the primary server’s IP address is listed for the reverse zone entries just as it is for the hosts data.

For example, the following boot file is *incorrect* because the primary server’s IP address (129.146.168.119) is missing from the secondary `in-addr.arpa` record:

```
;  
; /etc/named.boot file for dnssecondary  
directory /var/named  
secondary doc.com 129.146.168.119 dnshosts.bakup  
secondary 168.146.129.in-addr.arpa doc.rev.bakup
```

This is what the correct file should look like:

```
;  
; /etc/named.boot file for dnssecondary  
directory /var/named  
secondary doc.com 129.146.168.119 dnshosts.bakup  
secondary 168.146.129.in-addr.arpa 129.146.168.119 doc.rev.bakup
```

Server Failed and Zone Expired Problems

When a secondary server cannot obtain updates from its master, it logs a `master unreachable` message. If the problem is not corrected, the secondary expires the zone and stops answering requests from clients. When that happens, users start seeing `server failed` messages.

Symptoms:

- Masters for secondary zone *domain* `unreachable` messages in `syslog`.
- Secondary zone *domain* `expired` messages in `syslog`.
- ***** *domain* Can't find *name*: server failed** messages to users.

Note that if the problem lies with a secondary server, some users could still be successfully obtaining DNS information from the master and thus operating without experiencing any difficulty.

Possible causes:

The two most likely causes for these problems are network failure and a wrong IP address for the master in the secondary's boot file.

Diagnosis and solution:

- Check that the secondary's boot file contains the correct IP address for the master. Check the line:

```
secondary domain IPaddress hostsfile
```

Make sure that the IP address of the master matches the master's actual IP address and the address for the master specified in the hosts file. If the IP address is wrong, correct it, and then reboot the secondary.

- If the master's IP address is correct, make sure the master is up and running correctly by pinging the master's IP address: For example, to ping the master at IP address `129.146.168.119`, you would enter:

```
% ping 129.146.168.119 -n 10
```

- If the master does not respond to the ping, make sure it is up and running properly.
- If the master is running okay, use `ps` to make sure it is running `named`. If it is not running `named`, reboot it.

- If the master is correctly running `named`, you most likely have a network problem.

rlogin, rsh, and ftp Problems

Symptoms:

- Users are asked for password when they try to `rlogin` to a machine in another domain over the Internet.
- Users are denied access when they try to `ftp` to a machine in another domain over the Internet.
- Users are denied access when they try to use `rlogin` or `rsh` to a machine on their own network.

Possible causes:

- The user is working at a machine that does not have a PTR record in the primary master server's `hosts.rev` file.
- A missing or incorrect delegation of a sub-domain in the `hosts.rev` file.

Diagnosis and solution:

Check the appropriate `hosts.rev` file and make sure there is a PTR record for the user's machine. For example, if the user is working at the machine `altair.doc.com` with an IP address of `129.146.168.46`, the `doc.com` primary master server's `doc.rev` file should have an entry like:

```
46 IN PTR altair.doc.com.
```

If the record is missing, add it to the `hosts.rev` file and then reboot the server or reload its data as explained in "Forcing `in.named` to Reload DNS Data" on page 494.

Check and correct the NS entries in the `hosts.rev` files and then reboot the server or reload its data as explained in "Forcing `in.named` to Reload DNS Data" on page 494.

Other DNS Syntax Errors

Symptoms:

Error messages in console or syslog with operative phrases like the following are most often caused by syntax errors in DNS data and boot files:

- No such...
- Unknown field...
- Non-authoritative answer:
- Database format error...

- illegal or (illegal)
- error receiving zone transfer

Check the relevant files for spelling and syntax errors.

A common syntax error is misuse of the trailing dot in domain names (either using the dot when you should not, or not using it when you should). See “Trailing Dots in Domain Names ” on page 493.

FNS Problems and Solutions

This section presents problem scenarios with a description of probable causes, diagnoses, and solutions.

See “FNS Error Messages” on page 557 for general information about FNS error messages, and Appendix B.

Cannot Obtain Initial Context

Symptom:

You 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:

When you run `fnlist` to see what is in the initial context, you 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 `nisls` 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:

You 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 413. 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:

You 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 `nislsls` command to see what subdirectories exist. Run `nislsls` 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 you 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 who are part of that group to subsequently 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 You Created

Symptom:

When you run `fndestroy` on the `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:

You 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:

You use the `-s` option with `fnbind` and `fncreate`, but for certain names you get “name in use.”

Possible Cause:

`fnbind -s` and `fncreate -soverwrite` 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` Does Not Return Operation Failed

Symptom:

When you do an `fndestroy` or `fnunbind` on certain names that you know do *not* exist, you 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 some common 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, 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 Error Messages

Some of the error messages documented in this chapter are documented more fully in the appropriate man pages.

Some of the error messages documented in this chapter are documented more fully in the appropriate man pages.

Error Message Context

Error messages may appear in pop-up windows, shell tool command lines, user console window, or various 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 (file, map, table or directory) your command is addressing. However, in some cases an error message may be 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 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 naming software and servers make routine 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, then some other 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 a 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 error message may have slightly different meanings depending on which part of various naming software applications generated the message. For example, when a "Not Found" type message is generated by the `nisl` 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

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."
- Error messages beginning with (or containing) the word *NIS+* are alphabetized after messages beginning with (or containing) the word *NIS*.
- Some error messages may be preceded by a date or the name of the host, application, program, or routine that generated the error message, followed by a

colon. In these cases, the initial name of the command is used to alphabetize the message/

- Many messages contain variables such as user IDs, process numbers, domain names, host names, and so forth. In this appendix, these variables are indicated by an *italic typeface*. 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` (where `sales` is a variable) would be listed in this appendix as: `name: is not a table` and would be alphabetized as: `is not a table` among those messages beginning with the letter "I".
- 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.

Numbers in Error Messages

- Many DNS or other messages include an IP address. IP addresses are indicated by *n.n.n.n*.
- Some error messages include numbers such as process ID numbers, number of items, and so forth. Numbers in error messages are indicated: *nmmn*.

FNS Error Messages

FNS messages are encapsulated in the `FN_status_t` object as status codes. See the `FN_status_t` man page for the corresponding status codes

When an error occurs, FNS commands print out the remaining part of the name on which the operation failed. The part of the name that has not been printed has been processed successfully.

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: ''
```

Common Namespace 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 “NIS+ Database Problems” on page 511 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 “NIS+ Database Problems” on page 511 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 532 for information on insufficient memory problems.

`add_pingitem: Couldn't add directoryname to pinglist (no memory)`

See “Insufficient Memory” on page 532 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.

Note - If you started `rpc.nisd` with the `-YB` option, you must also kill the `rpc.nisd_reply` daemon.

`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 table, use `niscat` to check the contents of the table and `nistbladm` to delete any existing contents.
- If you are trying to delete a directory, use `nislsls -l -R` to check for existing files or subdirectories and delete them first.
- If you are trying to dissociate a replica from a domain with `nisrmdir -s`, and the replica is down or otherwise out of communication with the master, you will get this error message. In such cases, you can run `nisrmdir -f -s replicaname` on the master to force the dissociation. Note, however, that if you use `nisrmdir -f -sto` to dissociate an out-of-communication replica, you *must* run `nisrmdir -f -s again` as soon as the replica is back on line in order to clean up the replica's `/var/nis` file system. If you fail to rerun `nisrmdir -f -s replicaname` when the replica is back in service, the old out-of-date information left on the replica could cause problems.

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.

`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_marshal: 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 525 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_marshal: DES encryption failure` error message).

See "NIS+ Security Problems" on page 519 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 to use `rdate servername` to manually resync the client clock to the server clock.

```
authdes_seccreate: keyserv is unable to generate session key
```

This indicates that `keyserv` was unable to generate a random DES key for this session. This requires some action on your part:

- Check to make sure that `keyserv` 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 `keyserv` 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 “NIS+ System Resource Problems” on page 532 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 “NIS+ Security Problems” on page 519 for additional information regarding security key problems.

`authdes_validate: DES decryption failure`

See `authdes_marshal: DES decryption failure` on .

`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 532 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 nnnn*

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*

The system cannot add to the log file. The reason is indicated by the *string*. The most common cause of this message is lack of disk space. See “Insufficient Disk Space” on page 532.

Cannot obtain Initial Context

FNS error message. Indicates an installation problem. See “Cannot Obtain Initial Context” on page 549.

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 “NIS+ Database Problems” on page 511.

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 get any map parameter information.

See "NIS Problems and Solutions" on page 537

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 "NIS+ Security Problems" on page 519 for information on diagnosing and solving these type of problem.

*** *servername.domainname* can't find *machinename*; Server failed.

DNS error message. See "Server Failed and Zone Expired Problems" on page 547 and "Other DNS Syntax Errors" on page 548.

Can't find server name for address 127.0.0.1; server failed.

DNS error message. This message usually indicates that your primary master server is using an outdated `named.ca` file with invalid information. If your network is connected to the Internet, you need to get a current `named.ca` file from the authority that administers your top level domain (`.com`, for instance). For `.com`, `.edu`, `.gov`, `.mil`, `.org`, and others, that authority is InterNIC. If your network is not connected to the Internet, you need to check your `named.ca` file for errors.

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* '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 “NIS+ Security Problems” on page 519 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: could not get secret key for '*string*'

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 “NIS+ Security Problems” on page 519 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 533 for additional information.

Couldn't parse access rights for column *string*

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

At attempt to look up a table has failed. See “NIS+ Object Not Found Problems” on page 514 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`

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 532 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_TYPERMISMATCH`. 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 name 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 `nisserver`.
- 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 `nisserver`.

**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 `nisserver`.
- 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 `nisserver`.

**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 `nisserver`.

****ERROR:** it failed to make the *domainname* directory

The NIS+ command `nismkdir` failed to make the new directory *domainname* when running `nisserver` 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 `nissserver` 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 `nissserver -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 receiving zone transfer

DNS error message. This usually indicates a syntax error in one of the primary server's DNS files. See "Other DNS Syntax Errors" on page 548.

****ERROR:** table *tablename* .org_dir.*domainname* 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 ``*clientname*'' is in both the passwd and hosts tables. You cannot have an username same as the host name.

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 host name (`-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 unsend 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.

getzone: print_update failed

DNS error message. This usually indicates a syntax error in one of the primary server's DNS files. See "Other DNS Syntax Errors" on page 548.

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 509 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 507 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.

in.named [*nnnn*]: lame server on *hostname*

DNS error message. Lame delegation is when an NS record in the `hosts` file of a parent domain server identifies another server as authoritative for a subdomain zone, but that server is not authoritative for that zone. The NS records in the parent's `hosts` file must be a superset that includes all the authoritative servers in all delegated sub zones.

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 9, or add you to a group that does have the required permissions as described in Chapter 11.

See “NIS+ Ownership and Permission Problems” on page 517 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.

You cannot link table entries. A link at the entry level may produce this error message.

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 *number* 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 “l” the same as the numeral “1.”

For other possible causes of this message, see “Login Incorrect Message ” on page 520.

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 “NIS+ Database Problems” on page 511.

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 is that someone changed an existing domain name. Existing domain names should not be changed. See “Changed Domain Name” on page 526.

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 “NIS+ Performance and System Hang Problems” on page 528 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 535 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.

servername named [*nnnn*]: directory *directoryname*: No such file or directory.

DNS error message. This usually indicates a syntax or spelling error in a DNS boot or data file.

servername named [*nnnn*]: /etc/named.boot: line *n* unknown field '*name*'

DNS error message. This often indicates a spelling error in the DNS `named.boot` file. For example, "primary" or "secondary" might be misspelled.

servername named [*nnnn*]: *servername* has CNAME and other data (illegal)

DNS error message. This often indicates a syntax error in, or misuse of, a CNAME record for machine *servername*.

servername named [*nnnn*]: *domainname* Line *n*: Database format error (*n.n.n.n.n*)

DNS error message. The resource record for the machine in domain *name1*, whose IP address is *n.n.n.n* may be missing the type (usually IN) or have some other syntax error.

servername named [*nnnn*]: Line *n* Unknown type: *n.n.n.n*.

DNS error message. The DNS hosts file resource record for the machine whose IP address is *n.n.n.n* does not include the type (usually IN).

servername named [*nnnn*]: secondary zone *zonename* expired.

DNS error message. See “Server Failed and Zone Expired Problems ” on page 547.

servername named [*nnnn*]: zoneref: Masters for secondary zone *zonename* unreachable

DNS error message. See “Server Failed and Zone Expired Problems ” on page 547.

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 -rto` 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 make terminated

A problem caused your NIS make operation to terminate before successful conclusion. Check your NIS make file for problems and syntax errors.

NIS: server not responding for domain *domainname*. Still trying

See “NIS Problems and Solutions” on page 537.

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 532 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: NIS+ server busy try again later.

See “NIS+ Performance and System Hang Problems” on page 528 for possible causes.

NIS+ server busy try again later.

Self explanatory. Try the command later.

See also “NIS+ Performance and System Hang Problems” on page 528 for possible causes.

NIS+ server for *string* not responding still trying

See “NIS+ Performance and System Hang Problems” on page 528 for possible causes.

NIS+ server not responding

See “NIS+ Performance and System Hang Problems” on page 528 for possible causes.

NIS+ server needs to be checkpointed. Use `nisping -Cdomainname`



Caution - 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 509 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 532 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, ignore

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 535 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 ping 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_NOFILESIZE`.

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 “NIS+ System Resource Problems” on page 532 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 550. See also “No Permission” on page 517.

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 “NIS+ Object Not Found Problems” on page 514 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 “NIS+ Object Not Found Problems” on page 514 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* (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 “NIS+ Security Problems” on page 519 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 146 for further information.

Permission denied

Returned when you do not have the permissions required to perform the operation you attempted. See “NIS+ Ownership and Permission Problems” on page 517 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 10 and the “NIS+ Security Problems” on page 519 section of the Appendix A 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 9, 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 “NIS+ Security Problems” on page 519 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 149 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.

Reason: can't communicate with ypbind.

See "NIS Problems and Solutions" on page 537

`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 535 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 errorstring`

This message indicates a problem (identified by the *error string*) in carrying out a dump to a replica. See "Replica Update Failure" on page 535 for further information.

replica_update: *nnnn* updates *nnnn* 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*: 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 535 for further information.

RPC failure: ``RPC failure on yp operation.

This message is returned by `ypcat` when a NIS client’s `nsswitch.conf` file is set to `files` rather than `nis`, and the server is not included in the `/etc/hosts` file

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 518 for additional information.

date: hostname: sendmail (*nnnn*) : gethostbyaddr failed

One common cause of this problem is entering IP addresses in NIS+, NIS, files, or DNS data sets with leading zeros. For example, you should never enter an IP address as `151.029.066.001`. The correct way to enter that address is:
`151.29.66.1`.

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 *nn* 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 146 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 principalname`

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`

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: corrupted window from principalname`

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 principalname`

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: invalid timestamp received from principalname`

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 principalname`

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 principalname`

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 principalname`

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 principalname`

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 resynch 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 principalname`

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 144 or “Password Change Failures” on page 147 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*'

See “NIS+ Object Not Found Problems” on page 514 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: Unable to decrypt secret key for *string*.

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 “NIS+ Security Problems” on page 519 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` man page for additional information.

`update_directory: nnnn` 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 *principalname*

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+ and Login Passwords in `/etc/passwd` File” on page 526 and “Secure RPC Password and Login Passwords Are Different” on page 526 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 “NIS+ System Resource Problems” on page 532).
- 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 `nissserver` 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 `nissserver -rscript` 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 `nissserver -rscript` 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 -rscript` 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*) do not have Secure RPC credentials in NIS+ domain '*string*'

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 “NIS+ Ownership and Permission Problems” on page 517 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 145 for further information.

Your password will expire in *nn* 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 145 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 532.
- Database corruption.
- Some other type of software error. Contact the Sun Solutions Center.

`ypcat: can't bind to NIS server for domain domainname. Reason:
can't communicate with ypbind.`

See “NIS Problems and Solutions” on page 537

`yppoll: can't get any map parameter.`

See “NIS Problems and Solutions” on page 537

Information in NIS+ Tables

This appendix summarizes the information stored in the default NIS+ tables supplied in the Solaris 2.6 release (See Chapter 13, for general information regarding NIS+ tables and the commands used to administer them.)

- “auto_home Table ” on page 605
- “auto_master Table ” on page 605
- “bootparams Table ” on page 606
- “client_info Table ” on page 608
- Table C-4
- “ethers Table ” on page 609
- “group Table ” on page 610
- “hosts Table ” on page 610
- “mail_aliases Table ” on page 611
- “netgroup Table ” on page 612
- “netmasks Table ” on page 613
- “networks Table ” on page 614
- “passwd Table ” on page 614
- “protocols Table ” on page 616
- “rpc Table ” on page 617
- “services Table ” on page 617
- “timezone Table ” on page 618

NIS+ Tables

In an NIS+ environment, most namespace information is stored in NIS+ tables.

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.

Note - This appendix describes only those that are distributed as part of NIS+. Users and application developers frequently create NIS+ compatible tables for their own purposes. For information about tables created by users and developers, you must consult the documentation that they provide.

All NIS+ tables are stored in the domain's `org_dir` NIS+ directory object except the `admin` and `groups` tables that are stored in the `groups_dir` directory object.

Note - Do not link table entries. Tables may be linked to other tables, but do not link an entry in one table to an entry in another table.

NIS+ Tables and Other Name Services

In the Solaris environment the name service switch file (`nsswitch.conf`) allows you to specify one or more sources for different types of namespace information. In addition to NIS+ tables, sources can be NIS maps, DNS zone files, or `/etc` tables. The order in which you specify them in the switch file determines how the information from different sources is combined. (See Chapter 2 for more information on the switch file.)

NIS+ Table Input File Format

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 /---.
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 path name, 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. When creating additional automount maps to supplement auto_master (which is created for you), the column names must be key and value. For more information about the automounter consult your automounter or NFS file system documentation.

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	<p>Root partition: the location (server name and path) of the workstation's root partition</p> <p>Swap partition: the location (server name and path) of the workstation's swap partition</p> <p>Dump partition: the location (server name and path) of the workstation's dump partition</p> <p>Install partition.</p> <p>Domain.</p>

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:

```
buckarooroot=bigriver:/export/root1/buckaroo \  
swap=bigriver:/export/swap1/buckaroo \  
dump=bigriver:/export/dump/buckaroo \  
install=bigriver:/export/install/buckaroo \  
domain=sales.doc.com
```

Additional parameters are available for x86-based workstations. See the `bootparams` man page for additional information.

client_info Table

The `client_info` table is an optional internal NIS+ table used to store server preferences for the domain in which it resides. This table is created and maintained with the `nisprefadm` command.



Caution - Only use `nisprefadm` to work with this table. Do not use any other NIS+ commands on this table.

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 7, 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 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 UNIX user groups. 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

Earlier 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 Release 2x 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, the `netgroup` specification shown below 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, -,doc.com.) (-,joe,doc.com.)
```

The first specification includes one workstation, `host1`, in the `doc.com.` domain, but excludes all users. The second specification includes one user in the `doc.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 zeros 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.)

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

TABLE C-13 passwd Table Shadow Column (continued)

Item	Description
Expire	An absolute date past which the user's account is no longer valid
Flag	Reserved for future use: currently set to 0.

Earlier Solaris releases used a +/- syntax in local `/etc/passwd` files to incorporate or overwrite entries in the NIS password maps. Since the Solaris Release 2x 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 Release 2x 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 00000 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

FNS Reference Formats and Syntax

This appendix contains supplemental information about the use of DNS text (TXT) records and the use of X.500 attributes in XFN references.

- “DNS Text Record Format for XFN References ” on page 619
- “X.500 Attribute Syntax for XFN References ” on page 621

DNS Text Record Format for XFN References

The Solaris environment conforms to the XFN specification for federating global naming systems within DNS. In order to federate a naming system under DNS, you will need to enter information into DNS TXT resource records. This information will then be used to construct an XFN reference for that subordinate naming system. This appendix describes the format of these DNS TXT records.

- See Chapter 25. for the procedures needed to federate DNS.
- For details on how to manipulate records in DNS in general, see *DNS and BIND in a Nutshell*, by Paul Albitz and Crickett Liu, (O’Reilly and Associates, 1992).

The reference type of an XFN reference is constructed from a TXT record that begins with the XFNREF tag. It has the following format:

```
TXT "XFNREF rformat reftype"
```

If spaces occur within the string appearing after TXT, such spaces must be escaped, or the entire string must be enclosed within double quotation marks. The three fields, XFNREF, *rformat* and *reftype*, are separated using space (spaces and tabs). *rformat* specifies format of the reference type identifier. It can be one of

- STRING – Maps to FN_ID_STRING format
- OID – Maps to FN_ID_ISO_OID_STRING format
- UUID – Maps FN_ID_DCE_UUID format

reftype specifies the contents of the reference type identifier.

If no XFNREF TXT record exists, the reference type defaults to an identifier XFN_SERVICE, with an FN_ID_STRING format. If more than one XFNREF TXT record exists, the handling of the record is undefined. The following TXT record is equivalent to the default XFNREF:

```
TXT "XFNREF STRING XFN_SERVICE"
```

The address information for an XFN reference is constructed using TXT records with tags prefixed with the XFN string. Multiple addresses may be specified for a single reference. Records with the same tag are grouped and passed to the handler for each group. Each handler generates zero or more addresses from its group of TXT records and appends the addresses to the reference. The XFNREF tag is special in that it is used only to construct the reference type and thus, it is excluded from the address-construction process.

The syntax of address TXT records is as follows:

```
XFNaddress_type_tag address_specific_data
```

The two fields, *XFN_address_type_tag* and *address_specific_data*, are separated using space (spaces and tabs). The *address_type_tag* specifies the handler to be used for *address_specific_data*.

TXT records have a limitation of 2K bytes of characters per record. If the address-specific data is too long to be stored in a single TXT record, multiple TXT records may be used, as shown:

```
TXT "XFNaddress_type_tag address_specific_data1"
TXT "XFNaddress_type_tag address_specific_data2"
```

When the tag-specific handler is called, both records are passed to it. The handler is responsible for determining the order in which these two lines need to be interpreted.

The order in which TXT records appear is not significant. If lines with different tags are present, lines with the same tag are grouped together before the tag-specific handler is called. In the following example, the handler for *tag1* will be called with two text lines, and the handler for *tag2* will be called with three text lines.

```
TXT "XFNtag1 address_specific_data1"
TXT "XFNtag2 address_specific_data2"
TXT "XFNtag1 address_specific_data3"
TXT "XFNtag2 address_specific_data4"
TXT "XFNtag2 address_specific_data5"
```

Here are some examples of TXT records that can be used for XFN references.

Example 1

```
TXT "XFNREF STRING XFN_SERVICE"
TXT "XFNNISPLUS doc.com. nismaster 129.144.40.23"
```

Example 2

```
TXT "XFNREF OID 1.3.22.1.6.1.3"
TXT "XFNDCE (1 fd33328c4-2a4b-11ca-af85-09002b1c89bb...)"
```

The following is an example of a DNS table with a subordinate naming system bound in it.

```
$ORIGIN test.doc.com
@      IN SOA  foo root.eng.doc.com      (
        100   ;; Serial
        3600  ;; Refresh
        3600  ;; Retry
        3600  ;; Expire
        3600  ;; Minimum
      )
      NS   nshost
      TXT  "XFNREF STRING XFN_SERVICE"
      TXT  "XFNNISPLUS doc.com. nismaster 129.144.40.23"
nshost IN  A  129.144.40.21
```

X.500 Attribute Syntax for XFN References

This section contains supplemental information about the use of X.500 attributes for XFN references. In order to permit an XFN reference to be stored as an attribute in

X.500, the directory schema must be modified to support the object classes and attributes defined in this appendix.

- See Chapter 25. for the procedures needed to federate X.500.
- See *Managing the X.500 Client Toolkit* for information about modifying the X.500 directory schema.

Object Classes

Two new object classes, XFN and XFN-supplement, are introduced to support XFN references. The XFN object class is not relevant in FNS since SunSoft's X.500 directory product cannot support the introduction of new compound ASN.1 syntaxes. Instead, FNS uses the XFN-supplement object class.

The two new object classes are defined in ASN.1 as follows:

```
xFN OBJECT-CLASS ::= {
    SUBCLASS OF      { top }
    KIND             auxiliary
    MAY CONTAIN     { objectReferenceId |
                   objectReference |
                   nNSReferenceId |
                   nNSReference }
    ID              id-oc-xFN
}
id-oc-xFN OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) ansi(840) sun(113536)
    ds-oc-xFN(24)
}
xFNsSupplement OBJECT-CLASS ::= {
    SUBCLASS OF      { top }
    KIND             auxiliary
    MAY CONTAIN     { objectReferenceString |
                   nNSReferenceString }
    ID              id-oc-xFNsSupplement
}
id-oc-xFNsSupplement OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) ansi(840) sun(113536)
    ds-oc-xFNsSupplement(25)
}
```

The XFN-supplement object class is defined as an auxiliary object class so that it may be inherited by all X.500 object classes. It is defined with two optional attributes:

- `objectReferenceString` is used to hold an XFN reference to the object itself.
- `nNSReferenceString` is used to hold an XFN reference to a next naming system.

Both attributes are defined in ASN.1 as follows:

```

objectReferenceString ATTRIBUTE ::= {
    WITH SYNTAX                OCTET STRING
    EQUALITY MATCHING RULE     octetStringMatch
    SINGLE VALUE               TRUE
    ID                         { id-at-objectReferenceString }
}
id-at-objectReferenceString OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) ansi(840) sun(113536)
    ds-at-objectReferenceString(30)
}
nNSReferenceString ATTRIBUTE ::= {
    WITH SYNTAX                OCTET STRING
    EQUALITY MATCHING RULE     octetStringMatch
    SINGLE VALUE               TRUE
    ID                         { id-at-nNSReferenceString }
}
id-at-nNSReferenceString OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) ansi(840) sun(113536)
    ds-at-nNSReferenceString(31)
}

```

Both `objectReferenceString` and `nNSReferenceString` store XFN references in a string form. Their octet string syntax is further constrained to conform to the following BNF definition:

```

<ref> ::= <id> '$' <ref-addr-set>
<ref-addr-set> ::= <ref-addr> | <ref-addr> '$' <ref-addr-set>
<ref-addr> ::= <id> '$' <addr-set>
<addr> ::= <hex-string>
<id> ::= 'id' '$' <string> |
        'uuid' '$' <uuid-string> |
        'oid' '$' <oid-string>
<string> ::= <char> | <char> <string>
<char> ::= <PCS> | '\ ' <PCS>
<PCS> ::= // Portable Character Set:
          // !"#%&'()*+,-./0123456789:;<=>?
          // @ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_
          // `abcdefghijklmnopqrstuvwxyz{|}~
<uuid-string> ::= <uuid-char> | <uuid-char> <uuid-string>
<uuid-char> ::= <hex-digit> | '-'
<oid-string> ::= <oid-char> | <oid-char> <oid-string>
<oid-char> ::= <digit> | '.'
<hex-string> ::= <hex-octet> | <hex-octet> <hex-string>
<hex-octet> ::= <hex-digit> <hex-digit>
<hex-digit> ::= <digit> |
               'a' | 'b' | 'c' | 'd' | 'e' | 'f' |
               'A' | 'B' | 'C' | 'D' | 'E' | 'F'
<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' |
           '6' | '7' | '8' | '9'

```

The following example is a string form XFN reference:

```
id$onc_fn_enterprise$id$onc_fn_nisplus_root$0000000f77697a2e636fd2e2062696762696700
```

The example uses an XFN reference of type `onc_fn_enterprise`. It contains the address type `onc_fn_nisplus_root` and a single address value. The address value is an XDR-encoded string, comprising the domain name, `doc.com`, followed by the host name, `cygnus`.

An XFN reference may be added to an X.500 entry by using the FNS command `fnattr`, as in this example:

```
# fnattr -a ../c=us/o=doc object-class top organization xfn-supplement
```

creates a new entry called `c=us/o=doc` and adds an object class attribute with the values `top`, `organization`, and `xfn-supplement`.

The FNS command `fnbind` binds the NIS+ reference to the named entry and links X.500 to the root of the NIS+ namespace. (Note the use of a trailing slash in the name argument to `fnbind`.)

```
# fnbind -r ../c=us/o=doc/ onc_fn_enterprise onc_fn_nisplus_root  
"doc.com. cygnus"
```

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.
application-level name service	Application-level name services are incorporated in applications offering services such as files, mail, and printing. Application-level name services are bound below enterprise-level name services. The enterprise-level name services provide contexts in which contexts of application-level name services can be bound.
atomic name	An FNS (XFN) term referring to the smallest indivisible component of a name as defined by the naming convention.
attribute	In FNS (XFN), each named object is associated with a set of zero or more attributes. Each attribute in the set has a unique attribute identifier, an attribute syntax, and a set of zero or more distinct attribute values.
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 a>
binding	In FNS (XFN), the association of an atomic name with an object reference. For simplicity, an object reference and the object it refers to are used interchangeably in this guide.
BNF	An FNS (XFN) acronym referring to a Backus-Naur Form.
cache manager	The program that manages the local caches of NIS+ clients (<code>NIS_SHARED_DIRCACHE</code>), 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 <i>domain</i> .
checkpointing	The process of writing changes to NIS+ data that are stored in server memory and recorded in the transaction log to the NIS+ tables stored on disk. In other words, updating the NIS+ tables with recent changes to the NIS+ data set.
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 <i>application</i> 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 <i>Domain Name System (DNS)</i> and file-server/file-client relationships such as <i>NFS</i> and diskless hosts. See also <i>client</i> .
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.
composite name	In FNS (XFN), a name that spans multiple naming systems. It consists of an ordered list of zero or more components. Each component is a name from the namespace of a single naming system. Composite name resolution is the process of resolving a name that spans multiple naming systems.
compound name	In FNS (XFN), a sequence of atomic names composed according to the naming convention of a naming system.
context	In FNS (XFN), an object whose state is a set of bindings with distinct atomic names. Every context has an associated naming convention. A context provides a lookup (resolution) operation, which returns the reference, and may provide operations such as binding names, unbinding names, and listing bound names.

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 <i>key encrypting key</i> .
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 <i>data encryption standard (DES)</i> .
directory	(1) An NIS+ directory is a container for NIS+ objects such as NIS+ tables, groups, or subdirectories (2) In UNIX, a container for files and subdirectories.
directory cache	A local file used to store data associated with directory objects.
distinguished name	A distinguished name is an entry in an X.500 directory information base (DIB) composed of selected attributes from each entry in the tree along a path leading from the root down to the named entry.
DNS	See <i>Domain Name System</i> .
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 usually corresponding to a Local Area Network (LAN) or Wide Area Network (WAN) or a portion of such a network. Syntactically, an Internet domain name consists of a sequence of names (labels) separated by periods (dots). For example, `sales.doc.com`.

(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 DNS administrative files. The domain name is required for the network information service database to work properly. See also <i>domain</i> .
Domain Name System (DNS)	A system that provides the naming policy and mechanisms for mapping domain and machine names to addresses outside of the enterprise, such as those on the Internet. DNS is the network information service used by the Internet.
encryption key	See <i>data encrypting key</i> .
enterprise-level name service	An enterprise-level naming service identifies (names) machines (hosts), users and files within an enterprise-level network. FNS also allows naming of organizational units, geographic sites, and application services.
enterprise-level network	An "enterprise-level" network can be a single Local Area Network (LAN) communicating over cables, infra-red beams, or radio broadcast; or a cluster of two or more LANs linked together by cable or direct phone connections. Within an enterprise-level network, every machine is able to communicate with every other machine without reference to a global naming service such as DNS or X.500/LDAP.
enterprise root	In FNS (XFN), the root context of an enterprise. A context for naming objects found at the root of the enterprise namespace.

entry	A single row of data in a database table.
federated naming service	The service offered by a federated naming system.
federated naming system	An aggregation of autonomous naming systems that cooperate to support name resolution of composite names through a standard interface. Each member of a federation has autonomy in its choice of operations other than name resolution.
federated namespace	An FNS (XFN) term referring to the set of all possible names generated according to the policies that govern the relationships among member naming systems and their respective namespaces.
FNS	See <i>Federated naming service</i> .
generic context	In FNS (XFN), a context for binding names used in applications.
GID	See <i>group ID</i> .
global context	In FNS (XFN), a context for naming objects that have global names (currently, DNS and X.500 are the only global naming systems specified by XFN).
global name service	A global naming service identifies (names) those enterprise-level networks around the world that are linked together via phone, satellite, or other communication systems. This world-wide collection of linked networks is known as the "Internet." In addition to naming networks, a global naming service also identifies individual machines and users within a given network.
group	<p>(1) A collection of users who are referred to by a common name.</p> <p>(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.</p> <p>(3) In UNIX, groups determine a user's access to files. There are two types of groups: default user group and standard user group.</p>
group ID	A number that identifies the default <i>group</i> for a user.
host context	In FNS (XFN), a context for naming objects related to a computer.
implicit naming system pointer	An FNS (XFN) term referring to an unnamed reference that points to a context in another naming system.

indexed name	A naming format used to identify an entry in a table.
initial context	In FNS (XFN), every XFN name is interpreted relative to some context, and every XFN naming operation is performed on a context object. The XFN interface provides a function that allows the client to obtain an initial context object that provides a starting point for resolution of composite names.
initial context function	An FNS function, <code>fn_ctx_handle_from_initial()</code> , that obtains the initial context which allows a client to obtain an initial starting point for name resolution.
Internet	The world-wide collection of networks interconnected by a set of routers that enable them to function and communicate with each other as a single virtual network.
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.
junction	An FNS (XFN) term referring to a name in one namespace bound to a context in the next naming system.
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 2.6 release 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)
naming convention	In FNS (XFN), every name is generated by a set of syntactic rules called a naming convention.
name resolution	The process of translating workstation or user names to addresses.
name server	Servers that run one or more network name services.
name service switch	A configuration file (<code>/etc/nsswitch.conf</code>) that defines the sources from which an NIS+ client can obtain its network information.
name service	A network service that handles machine, user, printer, domain, router, an other network names and addresses.
namespace	<p>(1) A namespace stores information that users, workstations, and applications must have to communicate across the network.</p> <p>(2) The set of all names in a naming system.</p> <p>(3) <i>NIS+ namespace</i>. A collection of hierarchical network information used by the NIS+ software.</p> <p>(4) <i>NIS namespace</i>. A collection of <i>non</i>-hierarchical network information used by the NIS software.</p> <p>(5) <i>DNS namespace</i>. A collection of networked workstations that use the DNS software.</p>
namespace identifier	An FNS (XFN) term referring to a special atomic name used to refer to the root of a namespace.
naming system	In FNS (XFN), a connected set of contexts of the same type (having the same naming convention) and providing the same set of operations with identical semantics. In the UNIX operating system, for example, the set of directories in a given file system (and the naming operations on directories) constitutes a naming system.
network mask	A number used by software to separate the local subnet address from the rest of a given Internet protocol address.

next naming system pointer (NNSP)	In FNS (XFN), a reference to a context in which composite names from subordinate naming systems are resolved.
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 <i>master server</i> and all the <i>replica</i> or <i>slave servers</i> .
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 <i>NIS</i> .
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 <i>master server</i> and all the <i>replica servers</i> .
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 <code>nsswitch.conf</code> file points to <code>nisplus</code> . Or any time a command is run with an option that forces it to operate on objects in an NIS+ namespace (for example, <code>passwd -r nisplus</code>).
NIS+ object	An NIS+ domain, directory, table, or group. See <i>domain</i> , <i>directory</i> , <i>group</i> , and <i>table</i> .
NIS+ principal	See <i>principal</i> .
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.
NNSP	See <i>next naming system pointer</i> .
organizational units	In FNS (XFN), an enterprise is organized into organizational units such as centers, laboratories, departments, divisions, and so on. An organizational unit is a subunit of an enterprise.

organizational unit context	In FNS (XFN), a context for naming objects related to an organizational unit within an enterprise.
parent context	In FNS (XFN), a context in which this context and its siblings are bound.
parent domain	See <i>domain</i> .
pinging	The process by which an NIS+ master server transfers a change a NIS+ data to the domain's replica servers.
preference rank number	A number which a machine uses to rank the order in which it tries to obtain namespace information from NIS+ servers. A machine will first try all servers with a given rank number before trying any server with the next highest rank number. For example, a machine will query NIS+ servers with a rank number of 0 before it queries any server with a rank number of 1.
preferred server	From the point of view of a client machine, a preferred NIS+ server is a server that the client should try to use for namespace information ahead of non-preferred servers. Servers that are listed in a client or domain's preferred server list are considered preferred servers for that client or domain.
preferred server list	A <code>client_info</code> table or a <code>client_info</code> file. Preferred server lists specify the preferred servers for a client or 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: <ul style="list-style-type: none"> ■ <i>Root principal</i>. A machine root user (user ID = 0). Requires only a DES credential. ■ <i>User principal</i>. 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 <i>entry</i> .
reference	An FNS (XFN) term referring to the thing bound to a name. It contains addresses identifying the communication endpoints of the object.
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 <i>slave</i> server.)
reverse resolution	The process of converting workstation IP addresses to workstation names using the DNS software.
root context	In FNS (XFN), a context for naming the objects found in the root of the namespace.
root domain	See <i>domain</i> .
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).
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.

server	<p>(1) In NIS+, NIS, DNS, and FNS (XFN) a host machine providing naming services to a network.</p> <p>(2) In the <i>client-server model</i> for file systems, the server is a machine with computing 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.</p> <p>(3) A <i>daemon</i> that actually handles the providing of files.</p>
server list	See preferred server list.
service context	In FNS (XFN), a context for naming objects that provide services.
site context	In FNS (XFN), a context for naming objects related to a physical site.
slave server	<p>(1) A server system that maintains a copy of the NIS database. It has a disk and a complete copy of the operating system.</p> <p>(2) Slave servers are called <i>replica servers</i> in NIS+.</p>
strong separation	An FNS (XFN) term referring to cases where the XFN context treats the XFN component separator as the naming system boundary.
subcontext	In FNS (XFN), a context bound within another context.
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 <i>Transport Control Protocol (TCP)</i> .
TCP/IP	Acronym for Transport Control Protocol/Interface Program. The protocol suite originally developed for the Internet. It is also called the <i>Internet</i> protocol suite. Solaris 2.6 release 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.
user context	In FNS (XFN), a context for naming objects related to a human user.
weak separation	An FNS (XFN) term referring to cases where the XFN context does not treat the XFN component separator as the naming system boundary.
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.
XFN link	In FNS (XFN), a special form of reference that has a composite name as an address. Like any other type of reference, an XFN link is bound to an atomic name in a context.
X.500	A global-level directory service defined by an Open Systems Interconnection (OSI) standard.

Index

Special Characters

+/- Syntax
 compat, 21
 nsswitch.conf files, 21
 passwd_compat, 21
, *see* FNS,

A

Abort_transaction: Internal database error
 messages (NIS+), 511
access rights, 625
access rights, *see* security,
administrative domain (DNS), 466
administrative domain (DNS), *see* DNS,
 administrative domains,
aging passwords, *see* passwords, aging,
API
 NIS+, 36
Application Programmer's Interface, *see* API,
application-level, 625
applications and FNS, 363
.asc, 312
atomic name, 625
atomic names, *see* FNS,
attempt to remove a non-empty table
 messages (NIS+), 510
attribute, 625
attributes (FNS), *see* FNS,
authentication, 33, 625
 principals, how authenticated, 85 to 87
 time stamp, 88
Authentication denied messages (NIS+), 519

Authentication error messages (NIS+), 519
authorization, 32
automounter
 maps, additional, 61
auto_direct.time maps, 308
auto_home maps, 606
auto_home tables
 nsswitch.conf file, and, 16
auto_home tables (NIS+), 605
 columns, 605
auto_home.time maps, 308
auto_man maps, 606
auto_master maps, 606
auto_master tables
 automount maps, additional, 61
 nsswitch.conf file, and, 16
auto_master tables (NIS+), 605, 606
 columns, 606
auto_programs maps, 606
awk, 312

B

backup-restore (NIS+), 261
 See also nisbackup and nisrestore,
 automating, 264
 backup directory, 265
 backup files, 266
 chronological sequence of, 264
 ctx_dir directories, 264
 data checking not performed, 262
 data on master only, 263
 file system backup, and, 264

- master server only, 262, 263
- namespace, entire, 263, 265
- over-writing, 264
- restoring, 267
- servers, replacing, 270
- specific directories only, 265
- subdirectories, and, 262
- subdomains, and, 264
- target directories, 264
- XDR encoding, 266
- backup_list files, 266
- Berkeley Internet Name Domain, *see* DNS,
- BIND, *see* DNS,
- binding, 625
- BNF, 625
- boot file (DNS), *see* named.boot files,
- bootparams tables (NIS+), 606
 - input file format, 607
- Busy try again later messages (NIS+), 528

C

- cache files (DNS), *see* named.ca files,
- cache manager, 48, 199, 625
 - missing, 530
 - server preference (NIS+), 243
 - server preferences and, 246
 - starting, 199
 - stopping, 199
- caching-only servers, *see* DNS, servers,
- Callback: - select failed messages (NIS+), 511
- CALLBACK_SVC: bad argument messages (NIS+), 511
- Can't find messages (DNS), 547
- Can't find suitable transport messages (NIS+), 514
- Cannot find messages (NIS+), 514
- Cannot get public key messages (NIS+), 519
- Cannot obtain initial context messages (FNS), 549
- Cannot remove replica messages (NIS+), 510
- "Cannot [do something] with log" type messages (NIS+), 532
- canonical, 377
- canonical identifiers (FNS), *see* FNS,
- Changing Key messages (NIS+), 518
- checkpointing, 626
- checkpointing, *see* nisping,

- child domain, 626
- chkey, 79, 92, 99, 101, 102, 106, 107, 153, 527
 - Chkey failed messages, 519
 - root password, changing, 147
- Chkey failed messages (NIS+), 519
- CHKPIPE, 309
- client, 626
- client-server model, 626
- clients
 - keys, updating, 114
 - NIS, 285
 - NIS+, 47
 - NIS+ initializing, 197
 - preferred servers, designating (NIS+), 242
 - search behavior (NIS+), 242
- client_info files, 249, 253
- client_info files and tables, 243, 244
 - changing, 244
 - rank numbers, 244, 245
 - single client, 246
 - subnet, 246
- client_info tables (NIS+), 608
- cold-start file, 626
- cold-start files, 48
 - nisupdkeys and, 112
- column values (NIS+ tables), *see* nistbladm,
 - column values,
- composite name, 626
- composite names (FNS), *see* FNS,
- compound name, 626
- compound names (FNS), *see* FNS,
- configuration files (NIS), *see* NIS maps,
- configuration, *see* setup,
- context, 626
- contexts (FNS), *see* FNS,
- core files, 508
- Corrupt database messages (NIS+), 511
- Corrupt log messages (NIS+), 511
- create, 119
- create rights, *see* security, access rights,
- cred table
 - contents, displaying, 227
- cred tables
 - authentication types, 95
 - details of, 94, 95
 - entries missing, 525
 - links not allowed, 234

- links, and, 94
- cred tables (NIS+), 608
 - columns, 609
 - links not allowed, 608
- cred tables, *see* credentials
- credentials,
 - credentials, 72, 84, 627
 - administration of, 103
 - administrator's, 99, 101
 - authentication components, 85
 - corrupted, 524
 - creating, 95
 - creating credentials, 96, 99 to 103
 - cred table, description of, 94, 95
 - credential information, 84
 - DES, 72, 85, 89
 - DES verification field, 90
 - DES, components of, 87
 - DES, details of, 89
 - DES, generation of, 90
 - how created, 97
 - LOCAL, 73
 - machine, 72
 - modifying credentials, 96
 - passwd, and, 152
 - principal authentication, 85 to 87
 - principal names, 98
 - problem solving, 520
 - removing, 104
 - removing credentials, 96
 - resetting, 520
 - secure RPC netnames, 98
 - storage of, 92
 - time stamp, 87
 - types of and users, 74
 - updating, 103
 - user, 72
- credentials, *see* cred tables
 - cred table,
- credentials, *see* keys
 - keys,
- cron files, 192, 202
- crontab, 316
 - NIS maps propagating, 314
 - NIS, problems, 543
- crontab files, 314
 - backup (NIS+), 264
 - NIS, problems, 543

- .cshrc files, 332
- ctx_dir, 367
 - backup of, 264
- ctx_dir directories, 42, 549
 - FNS mapping to, 413
- ctx_dir directory
 - creation of, 423

D

- daemons
 - in.named, 467
 - NIS, 285
 - NIS, not running, 541
 - npc.nisd, 195
 - npc.nisd EMULYP -Y -B, 196
 - npc.nisd, DNS forwarding, 196
 - npc.nisd, NIS-compatibility, 195
 - npc.nisd, security level, 195
 - npc.nisd, starting, 195, 196
 - npc.nisd, stopping, 197
 - rpc.nisd dies, 530
 - rpc.nisd, failure of, 512
 - rpc.nisd, problems, 511
 - rpc.yppasswdd daemon, 286
 - rpc.yppupdated daemon, 286
 - ypbind daemon, 286
 - ypserv daemon, 286
 - ypupdated, 294
 - ypxfr daemon, 286
- data encrypting key, 627
- data encryption standard, *see* DES,
- data.dict files, 40, 266
- /data directories (NIS+), 266
- Database corrupted messages (NIS+), 511
- Database format error messages (DNS), 549
- db.ADDR files, *see* hosts.rev files,
- db.cache files (DNS), *see* named.ca files,
- db.cache files, *see* named.local files,
- db.domain files, *see* hosts (DNS files),
- dbm, 312, 313
- decimal dotted notation, 627
- defaultdomain files
 - uninstalling NIS+, 277
- DES, 627
- DES credentials, *see* credentials,
- destroy, 119

- destroy rights, *see* security, access rights,
- .dict files, 40, 516
- directories, *see* NIS+ directories
 - NIS+,
- directory, 627
- directory cache, 48, 627
- directory name error messages (NIS+), 509
- directory objects, *see* NIS+ directories,
- disk space insufficient (NIS+), 532
- distinguished name, 627
- DNS, 8, 319, 464, 627, 628
 - See also* error messages,
 - A record, 487
 - administrative domains, 465, 466
 - bogus name logging, 491
 - boot files, 478
 - cache-only servers, 467
 - Can't find messages, 547
 - changes erratic, 545
 - class fields, 482
 - clients, 464
 - clients, resolver and, 467
 - CNAME record, 489
 - control entries, 483
 - data files, 478
 - data files, names of, 478
 - Database format error messages, 548
 - default domain name, 491
 - domain names, 471
 - domain names, fully qualified, 472
 - domain names, registering, 471
 - domain names, trailing dots, 493
 - domains, 468
 - domains, geographic (Internet), 470
 - domains, organizational (Internet), 470
 - domains, top level, 469
 - email, and, 477
 - EMULYP -Y -B, 196
 - error receiving zone transfer
 - messages, 549
 - filenames, and, 478
 - FNS, and, 369, 406
 - FNS, federating with, 346, 448
 - FNS, text record format, 619
 - FN_ID_DCE_UUID, 620
 - FN_ID_ISO_OID_STRING, 620
 - FN_ID_STRING, 620
 - ftp problems, 548

- HINFO record, 487
- hosts files, 480
- hosts.rev files, 480
- illegal messages, 549
- in-addr.arpa Domain, 474
- in.named, 467
- in.named, updating, 494
- \$INCLUDE control entry, 483
- \$INCLUDE files, 480
- Internet, and, 469
- Internet, joining, 470
- inverse queries, 491
- IP addresses, 464
- LOCALDOMAIN, 491
- machines, adding, 495
- machines, removing, 496
- modifying, 494
- MX record, 490
- MX records, 477
- name fields, 481
- name-address resolution, 464, 466
- named.boot files, 479
- named.ca files, 479
- named.local files, 480
- namespace, 468
- namespace, hierarchy, 468
- network, division into subdomains, 498
- NIS, 20
- NIS and, 318
- NIS+, 20
- NIS, and, 282, 283
- No such... messages, 548
- Non-authoritative answer messages, 548
- NS record, 486
- nsswitch.conf file, and, 12
- nsswitch.conf files, 20, 467
- OID, 620
- \$ORIGIN control entry, 484
- overview, 464
- primary servers, 467
- primary servers, changes on, 495
- problem solving, 544
- PTR record, 489
- record-specific-data fields, 482
- record-type fields, 482
- reloading data, 494
- resolver, 467

- resource records, formats of, 481
- resource records, special characters, 482
- resource records, types of, 484
- reverse domain data problems, 546
- reverse mapping, 473
- reverse resolution, 473
- RFC1535, 491
- rlogin problems, 548
- root domain servers, 465
- rpc.nisd starting, 195, 196
- rsh problems, 548
- secondary servers, 467
- server cannot find machine, 544
- server failed messages, 547
- servers, 464, 474
- servers, adding, 496
- servers, authoritative, 474
- servers, cache-only, 474
- servers, caching-only, 475
- servers, Internet, 476
- servers, master, 475
- servers, non-Internet, 477
- servers, primary master, 475
- servers, root domain, 474, 476
- servers, secondary, 474
- servers, secondary master, 475
- servers, types of, 467
- servers, zone master, 474
- short names, client cannot use, 546
- SOA record, 484
- SOA, changing number, 494
- Solaris implementation of, 491
- subdomains, 468
- subdomains, creating, 497
- subdomains, names of, 499
- subdomains, planning, 498
- subdomains, set up, 499
- syntax errors, 548
- test programs, 491
- TTL fields, 481
- TXT records (FNS), 448
- Unknown field messages, 548
- unreachable messages, 547
- utility scripts, 491
- UUID, 620
- version of, 491
- WKS record, 488
- XFN, text record format, 619
- zone expired messages, 547
- zone files, 473
- zones, 473
- DNS zone files, 627
- DNS zones, 627
- DNS, *see* DNS, servers
 - name servers,
- DNS, *see* hosts (DNS files)
 - hosts files,
- DNS, *see* hosts.rev files
 - hosts.rev files,
- DNS, *see* named.boot files
 - boot files,
- DNS, *see* named.ca files
 - cache files,
- DNS, *see* named.local files
 - named.local files,
- DNS-forwarding, 627
- domain, 627
- domain name, 628
- domain name error messages (NIS+), 509
- Domain Name System, *see* DNS,
- domain names
 - changing (NIS+), 526
 - incorrect (NIS), 538
 - missing (NIS), 538
- domainname
 - uninstalling NIS+, 274
- domains, 43
 - See also* DNS,
 - DNS, trailing dots, 493
 - domain names (DNS), 471
 - domain names, fully qualified, 472
 - domain names, registering, 471
 - geographic (Internet), 470
 - in-addr.arpa, 474
 - Internet, 469
 - NIS, 282, 285
 - NIS and NIS+ mixture, 283
 - NIS+ names of, 53
 - NIS+, checkpointing, 202
 - NIS, changing, 318
 - organizational (Internet), 470
 - passwd, and, 153
- domains, *see* root domains
 - root,

E

- echo, 133
- email, *see* DNS, email,
- EMULYP -Y -B, 196
- encryption key, 628
- enterprise naming services, *see* naming,
- enterprise root, 628
- enterprise-level name service, 628
- enterprise-level network, 628
- entries (table), *see* tables, entries,
- entry, 121, 629
- entry corrupt messages (NIS+), 511
- error messages, 555
 - alphabetization of, 556
 - context of, 555
 - display of, thresholds, 555
 - FNS messages, 557
 - interpretation of, 556
 - numbers in, 557
- error receiving zone transfer messages (DNS), 549
- /etc directories, 332, 527
- /etc files, 8, 21, 32, 277, 283, 287, 423
 - FNS, and, 329, 368
- /etc/.rootkey, 109, 527
- /etc/.rootkey files
 - servers (NIS+) replacing, 270
- /etc/auto* tables, 517
- /etc/auto_master files, 404
- /etc/bootparams files, 607
- /etc/defaultdomain, 515
 - uninstalling NIS+, 277
- /etc/defaultdomain files, 538
- /etc/defaults/passwd files, 169
 - MAXWEEKS, 169, 170
 - MINWEEKS, 169, 170
 - PASSLENGTH, 171
 - password minimum length, 171
 - WARNWEEKS, 169, 170
 - weeks, maximum (default), 169
 - weeks, minimum (default), 169
 - weeks, warning (default), 169
- /etc/fn/ directories, 416
- /etc/fn/x500.conf files, 451
- etc/hosts, 411
- /etc/hosts, 4
- /etc/hosts files, 418

- FNS, and, 400
- /etc/init.d/rpc, 196
- /etc/init.d/yp, 294
- /etc/named.boot files, 479
- /etc/named.pid files, 494
- etc/passwd files, 411
- /etc/passwd files
 - FNS, and, 400
 - nisaddent, and, 238
- /etc/passwd files, *see* password data,
- /etc/printers.conf, 344
- /etc/resolv.conf files
 - NIS and Internet, 319
- /etc/resolve.conf files, 267
- /etc/shadow
 - nisaddent, and, 238
- /etc/syslog.conf
 - error messages, 555
- ethers tables (NIS+), 609
 - address format, 609
 - columns, 609
- expire values, *see* password data,

F

- federated namespace, 629
- federated naming service, 629
- Federated Naming Service, *see* FNS,
- federated naming system, 629
- field, 121
- file contexts, 380
 - administering, 439
 - creating, 342, 440
 - creating, command line, 442
 - creating, input file, 441
 - hosts, creation, 429
 - input formats, 443
 - mounts locations, multiple, 443
 - names in, 332
 - users, creation, 429
- files contexts
 - names, composing in, 383
- files-based naming, 9
- fnattr, 335, 337, 338, 345, 456, 624
 - adding, 456, 457
 - deleting, 456, 458
 - FN_ID_DCE_UUID, 459

- FN_ID_ISO_OID_STRING, 459
- listing, 456, 458
- modifying, 456, 459
- NIS maps, and, 416
- options, 345, 457
- updating, 456
- fnbind, 338, 433
 - NIS maps, and, 416
 - NIS+ users, 339
 - options, 339
 - options for binding, 434
 - options for references, 435
 - syntax for binding names, 433
 - syntax for references, 434
- fncheck, 411
 - options, 411
 - syntax, 411
- fncopy, 338, 347, 419
 - /etc files to NIS, 419
 - NIS to NIS+, 417
 - options, 348, 417
 - syntax, 417
- fncreate, 333, 335, 338, 341, 400, 428
 - all-users contexts, 425
 - creating FNS namespace, 333
 - enterprise contexts, 422
 - fails, 551
 - generic contexts, 428
 - hosts contexts, 424, 425
 - hosts file contexts, 429
 - name service, default, 333
 - name service, non-default, 333
 - NIS maps, and, 416
 - NIS+ users, 339
 - NIS+, and, 334
 - NIS, and, 335, 400
 - NSID contexts, 429
 - options, 341, 422
 - orgunit contexts, 423
 - service contexts, 427
 - single-user contexts, 426
 - site contexts, 428
 - syntax, 422
 - usr file contexts, 429
- fncreate_fs, 338, 342, 444
 - command line, 442
 - compatibility, backward, 444
 - creating file contexts, 440
 - example, 441
 - input file, 441
 - input formats, 443
 - mounts locations, multiple, 443
 - NIS, SKI and, 328, 368
 - onc_fn_fs reference type, 441
 - onc_fn_fs_mount, 441
 - options, 440
 - syntax, 440
 - variables, use of, 443
- fncreate_printer, 338, 343, 344
 - NIS, SKI and, 328, 368
- fndestroy, 338, 345, 437
 - fails, 551
 - NIS maps, and, 416
- fnlist, 335, 413
 - context contents, 335
 - NIS maps, and, 416
 - options, 336, 431
 - suborganizations not listed, 551
 - syntax, 431
- fnlookup, 335, 336, 413
 - NIS maps, and, 416
 - options, 337, 430
 - syntax, 430
- fnrename, 436
 - NIS maps, and, 416
- FNS, 10, 326, 334, 357, 413, 629
 - See also* error messages,
 - ... (enterprise root), 386
 - _ character in names, 376
 - access control, 414
 - access control, changing, 415
 - administration, 338
 - administration of, 371
 - API usage model, 364
 - applications, and, 363, 364, 370
 - applications, calendar service
 - example, 401
 - applications, policies and, 400
 - applications, support for, 370
 - ASN.1, 622
 - atomic names, 359
 - attributes, 327, 453
 - attributes, adding, 456, 457
 - attributes, deleting, 456, 458
 - attributes, listing, 456, 458

- attributes, modifying, 456, 459
- attributes, overview, 360
- attributes, updating, 456
- attributes, viewing, 337, 453
- attributes, working with, 338, 345
- automounter, and, 404
- binding names to references, 433
- binding, command line from, 434
- binding, existing to new, 433
- bindings, creating, 338, 339
- bindings, displaying, 336
- bindings, removing, 338, 341
- bindings, renaming, 436
- Cannot obtain initial context
 - messages, 549
- canonical identifies, 393
- compatibility, backward, 444
- component separators, 377
- composite names, 326, 361, 365
- composite names, examples, 382
- composite names, removing, 436
- compound names, 360
- contexts, 327
- contexts, administering, 430
- contexts, bindings displaying of, 430
- contexts, cannot create, 551
- contexts, cannot remove, 551
- contexts, converting, 347
- contexts, copying, 338, 347
- contexts, creating, 338, 341, 422
- contexts, destroying, 338, 345, 437
- contexts, initial, 362
- contexts, listing, 431
- contexts, listing contents, 335
- contexts, overview, 359
- contexts, populating, 423
- contexts, underscore character, 423
- contexts, “_” character, 423
- creating, 333
- ctx_dir directories, 328
- DNS, 346, 445, 448
- DNS, and, 369
- DNS, federating, 406
- DNS, text record format, 619
- Enterprise Naming Services, 328
- enterprise root, 386
- enterprise root, ..., 386
- enterprise root, //org, 386

- enterprise root, files and, 390
- enterprise root, hosts and, 389
- enterprise root, Organizational
 - Subunits, 387
- enterprise root, printers and, 390
- enterprise root, services and, 389
- enterprise root, sites and, 387
- enterprise root, subordinate contexts, 387
- enterprise root, users and, 388
- /etc files, 329, 368
- /etc files to NIS, 419
- /etc files, and, 335, 418
- examples, 348
- examples, attributes, 351
- examples, attributes changing, 353
- examples, attributes listing, 351
- examples, creating bindings, 350
- examples, listing context bindings, 349
- examples, searching for objects, 355
- federating naming, 364
- file namespace, 375, 380
- file naming, 370
- file system contexts, creating, 338
- file system namespace, 403
- file systems, and, 362
- files namespace, composing names in, 383
- files-based naming, and, 329
- fn_ctx_handle_from_initial(), 392
- FN_ID_DCE_UUID, 436, 620
- FN_ID_ISO_OID_STRING, 436, 620
- FN_ID_STRING, 436, 620
- global namespace policies, 405
- global namespaces, federating, 346, 445
- global naming service, 368
- global naming services, 329
- host bindings, 394
- host bindings, thisens, 395
- host bindings, thishost, 394
- host bindings, thisorgunit, 395
- host namespace, 375, 379
- host namespace, aliases, 379
- host namespace, composing names in, 383
- initial context, empty, 549
- initial contexts, 362
- initial contexts, bindings within, 391, 392
- initial contexts, global namespaces, 406
- Internet domain names, 406

- large contexts, 416
- LDAP API, 451
- myens, 394
- myorgunit, 393
- myself, 393
- name binding, 433
- Name in Use messages, 552
- name service, default, 333
- name service, non-default, 333
- Name Services, 328
- name services, and, 409
- name services, changing, 410
- name services, default, 412
- name services, selecting, 410, 412
- names, files, 332
- names, hosts, 332
- names, organization, 330
- names, reserved, 381
- names, services, 332
- names, sites, 331
- names, users, 331
- namespace updates, 23
- namespace, example, 385
- namespace, identifiers, 376
- namespace, structure of, 383
- namespace, updating, 338
- namespace, viewing, 335
- namespaces, default, 375, 377
- namespaces, file system, 403
- namespaces, overview, 361
- namespaces, printer, 405
- namespaces, separators, 381
- naming inconsistencies, 411
- naming services, 366
- naming, enterprise level, 396
- NFS file servers, 403
- NIS clients, 328, 368
- NIS makefiles, and, 416
- NIS maps, and, 416
- NIS+ and NIS coexisting, 413
- NIS+ commands, and, 413
- NIS+, administration under, 338
- NIS+, and, 328, 334, 367
- NIS+, disk space, 334
- NIS+, domains, 334
- NIS+, mapping to objects, 413
- NIS+, moving from NIS, 417
- NIS+, user, privileges, 339
- NIS, administration under, 338
- NIS, and, 284, 328, 335, 367, 415
- NIS, fnsypd, and, 329, 368
- NIS, SKI and, 329, 368
- NIS, user, privileges, 339
- nNSReferenceString, 622
- no permission messages, 550
- nsswitch.conf files, 22
- objectReferenceString, 622
- OID, 620
- onc_fn_enterprise, 624
- onc_fn_nisplus_root, 624
- Operation Failed, 553
- //org (enterprise root), 386
- organization namespace, 375, 377
- organization namespace, composing
 - names in, 382
- organization namespace, NIS in, 378
- organization namespace, NIS+ in, 378
- organizations, sub, not listed, 550
- orgunit (NIS+), 328
- overview, 326, 357
- ownership, changing, 415
- policies, 329, 374, 384
- policies, applications and, 400
- policies, calendar service example, 401
- policies, files-based naming, 400
- policies, global namespaces, 405
- policies, NIS and, 399
- policies, NIS+ and, 397
- policies, NIS+ domains, 397
- policies, NIS+ hosts, 398
- policies, NIS+ organization names, 398
- policies, NIS+ security, 398
- policies, NIS+ users, 398
- policies, principles, 365
- policies, summary of, 330
- printer compatibility (/etc files), 419
- printer compatibility (NIS), 417
- printer contexts, creating, 338
- printer namespace, 375, 381, 405
- printer naming, 370
- problem solving, 549
- programming examples, 348
- querying, 337
- references, 359
- references, binding names to, 433

- references, command line, 434
- reserved names, 381
- root reference, X.500, 449
- root references, 446
- root references, NIS, 447
- root references, NIS+, 446
- separators, 377
- servers, NFS, 403
- service namespace, 375, 380
- service namespace, composing names
 - in, 383
- service namespace, reference registry, 381
- shorthand bindings, 395
- shorthand bindings, host, 395
- shorthand bindings, org, 396
- shorthand bindings, site, 396
- shorthand bindings, user, 395
- site namespace, 375, 379
- site namespace, composing names in, 383
- slash, trailing, 381
- Solaris, and, 366
- Solstice AdminSuite, and, 410
- thisens, 395
- thishost, 394
- thisorgunit, 395
- underscore in names, 376
- user bindings, 393
- user bindings, myens, 394
- user bindings, myorgunit, 393
- user namespace, 375, 380
- user namespace, composing names in, 382
- users, and, 362
- users, privileges, 338
- UUID, 620
- variables, use of, 443
- X.500, 346, 406, 445, 449
- X.500 client API, 451
- X.500 syntax, 621
- X.500, and, 369
- X.500, federating, 407
- X.500, object classes, 622
- X/Open Federated Naming, 326
- XDS/XOM API, 451
- XFN, 326
- xfn API, 363
- xfn links, 362
- XFN, and, 357
- FNS, *see* file contexts
 - file contexts,
 - fs contexts,
 - FNS, *see* FNS, enterprise root
 - root, enterprise,
 - FNS, *see* host contexts
 - host contexts,
 - FNS, *see* NISD contexts
 - namespace identifier contexts,
 - FNS, *see* NSID contexts
 - NSID contexts,
 - FNS, *see* org contexts
 - orgunit contexts,
 - FNS, *see* orgunit contexts
 - orgunit contexts,
 - FNS, *see* printer contexts
 - printer contexts,
 - FNS, *see* service contexts
 - service contexts,
 - FNS, *see* site contexts
 - site contexts,
 - FNS, *see* user contexts
 - user contexts,
- fnsearch, 337, 453, 455
 - Boolean operators, 338
 - expressions, 455
 - filter operators, 455
 - objects and attributes, 455
 - options, 454
 - searches, customizing, 455
 - syntax, 454
- fnselect, 333, 410, 412
 - name service, non-default, 333
 - options, 412
 - syntax, 412
- fnsypd, 329, 368
- fns_hosts.attr files, 418
- fns_hosts.attr maps, 416
- fns_hosts.ctx files, 418
- fns_hosts.ctx maps, 416
- fns_org.attr files, 418
- fns_org.attr maps, 416
- fns_org.ctx files, 418
- fns_org.ctx maps, 416
- fns_user.attr files, 418
- fnunbind, 338, 341
 - Name in Use messages, 552
 - NIS maps, and, 416

- NIS+ users, 339
- fn_ctx_initial.so libraries, 549
- FN_ID_DCE_UUID, 436, 459
- FN_ID_ISO_OID_STRING, 436, 459
- FN_ID_STRING, 436
- fs, 332
- ftp, 543
 - problems, 548
- full dump rescheduled messages (NIS+), 536

G

- generic context, 629
- generic contexts
 - creation, 427
- Generic system error messages (NIS+), 509
- gethostbyname, 11
- getpwnam, 11
- getpwuid, 11
- getXbyY, 11
- GID, 629
- global context, 629
- global name service, 629
- group, 629
- group class, 75, 76, 118, 119
- group class access rights, 125
- group ID, 629
- group tables, 76
- group tables (NIS+), 610
 - columns, 610
- group.org_dir directories, 174
- groups, 173
 - changing, 140, 141
 - netgroups, 174
 - netgroups (NIS), 303, 304
 - UNIX, 174
- groups, *see* NIS+ groups
 - NIS+ groups,
- groups.org_dir tables, 215
- groups_dir, 215
 - FNS, and, 367
- groups_dir directories, 42, 55, 76, 77, 174
 - FNS mapping to, 413
 - FNS, and, 328
 - uninstalling NIS+, 275, 276

H

- host context, 629
- host contexts, 379, 425
 - aliases (machine), 379
 - all, creation of, 424
 - host aliases, 425
 - names in, 332
 - names, composing in, 383
 - single, creation of, 425
- host maps
 - FNS, and, 367
- hosts (machines)
 - multihome support (NIS), 295
 - NIS clients, 284
 - NIS domains, changing, 318
 - NIS servers, 284
 - NIS+ names in, 56
- hosts contexts
 - cannot create, 551
- hosts database, 310
- hosts files, 495
- hosts files (DNS), 480
- hosts tables (NIS+), 610
 - columns, 611
- hosts.byaddr, 287
- hosts.byaddr maps
 - YP_INTERDOMAIN key, 319
- hosts.byname, 287
- hosts.byname maps, 287
 - YP_INTERDOMAIN key, 319
- hosts.bynamemaps, 399
- hosts.org_dir tables
 - FNS, and, 398
- hosts.rev files, 480, 495

I

- illegal messages (DNS), 549
- Illegal object type messages (NIS+), 507
- implicit naming system pointer, 629
- in.named, 8, 467
- inactive values, *see* password data,
- indexed name, 630
- indexed names (NIS+ tables), 215
- initial context, 630
- initial context function, 630
- initial contexts (FNS), *see* FNS,

- input files, 65
- installation, *see* setup,
- Insufficient permission messages (NIS+), 517, 519
- insufficient permission messages (NIS+), 527
- Internet, 630
 - See also* DNS,
 - DNS, and, 469
 - domain names, registering, 471
 - domains, geographic, 470
 - domains, organizational, 470
 - domains, top level, 469
 - FNS, and, 406
 - joining, 470
 - NIS, 20
 - NIS+, 20
 - NIS, and, 283
 - nsswitch.conf files, 20
 - root domain servers, 476
- Internet address, 630
- Invalid principal name messages (NIS+), 513
- IP, 630
- IP address, 630
- IP addresses
 - IP addresses, updating, 114
 - updating, 114

J

- junction, 630

K

- key (column), 630
- key (encrypting), 630
- key server, 630
- keylogin, 79, 86, 90 to 92, 102, 106, 107, 153, 527
 - secure and login passwords different, 526
- keylogout, 79, 86, 92
- keys, 105
 - changing, 107 to 111
 - client, updating, 114
 - common, 87, 88
 - DES, 87
 - keylogin, 106
 - keys, updating client's, 114
 - pairs, 98

- passwd, and, 153
- private, principal's, 86, 87
- private, re-encrypting, 108
- private, server's, 86, 88
- problem solving, 522
- public, principal's, 86, 88
- public, server's, 86, 87, 92
- random DES, 87
- re-encrypting private, 108
- time stamp, 87
- updating, 112, 113
- updating stale, 522
- updating, manually, 522
- keys (NIS+ tables), 213
- keyserv, 79, 277, 525
 - failure of, 525
 - uninstalling NIS+, 275, 276
- Keyserv fails to encrypt messages (NIS+), 519
- keyserver
 - nsswitch.conf file, and, 17

L

- LAN, 630
- last.upd files, 266
- LDAP API, *see* X.500,
- links (NIS+), *see* nisl,
- list of, 288
- LOCAL credentials, *see* credentials,
- local files, *see* files-based naming,
- local-area network, *see* LAN,
- Log corrupted messages (NIS+), 511
- log entry corrupt messages (NIS+), 511
- log files
 - disk space, insufficient, 533
- .log, 65
- .log files, 40, 236
 - old files, 516
- logging in, 144
- login, 106
- Login incorrect Message, 144
- login incorrect message, 534
- Login incorrect messages (NIS+), 519
- Login Incorrect messages (NIS+), 520
- .login files, 333
- ls, 404, 527, 537

M

- machines, *see* hosts (machines),
- mail, 365
- mail exchange records, 630
- mail hosts, 630
- mail_aliases tables (NIS+), 611
 - columns, 611
 - input file format, 611
- make, 304, 308, 309, 311, 318
 - NIS maps, 291
 - NIS maps and, 290
- Make files
 - NIS, 287
- makedbm, 286, 287, 291, 309, 311 to 313
 - maps, changing server of, 305, 306
 - slave servers, adding, 316
- Makefile, 306, 308, 309
 - NIS security, 300
 - non-default maps, modifying, 311
 - propagating maps, 313
 - YP_INTERDOMAIN key, 319
- Makefile files
 - 4.x compatibility, 295
 - maps, supported list, 307
 - multihome support, 295
- Makefile files (NIS), *see* NIS and NIS maps,
- maps (NIS), *see* NIS maps,
- master server, 631
- master servers, *see* NIS servers and NIS+ servers,
- max values, *see* password data,
- MAXWEEKS, *see* passwords, MAXWEEKS,
- memory insufficient (NIS+), 532
- messages (NIS+), 536
- min values, *see* password data,
- MINWEEKS, *see* passwords, MINWEEKS,
- MIS, 631
- modify, 119
- modify rights, *see* security, access rights,
- mymap.asc files, 312

N

- Name in Use messages (FNS), 552
- name resolution, 631
- name server, 631
- name service, 631
- name service switch, 33, 631

- name service switch, *see* nsswitch.conf files,
- name services, *see* naming,
- name space
 - DNS, 8
- name-to-address resolution, 464
- named.boot files, 479
- named.ca files, 476, 480
- named.local files, 480
 - TXT records (FNS), 448
- named.pid files, 494
- namespace, 631
- namespace identified contexts, *see* NSID contexts,
- namespace identifier, 631
- namespaces, *see* DNS DNS,
- namespaces, *see* FNS FNS,
- namespaces, *see* NIS NIS,
- namespaces, *see* NIS+ NIS+,
- naming, 3
 - DNS, 8
 - files-based, 9
 - FNS, 10
 - FNS, and, 396
 - name services, changing, 12
 - NIS, 9
 - NIS+, 9, 27, 39
 - NIS+ directories, 42
 - NIS+ structure, 40
 - Solaris name services, 8
- naming convention, 631
- naming system, 631
- naming system pointer, *see* NNSP,
- ndbm, 288
 - slave servers, adding, 316
- ndbm files
 - maps, changing server of, 306
- netgroup files, 303
 - entries, example, 304
- netgroup tables (NIS+), 612
 - columns, 612
 - input file format, 612
 - wildcards in, 613
- netgroup.byhost files, 303

- netgroup.byuser files, 303
- netgroup.org_dir directories, 174
- netgroups.org_dir tables, 215
- netmasks tables (NIS+), 613
 - columns, 613
- netstat
 - testing, 539
- Network Information Service Plus, *see* NIS+
- Network Information Service, *see* NIS,
- network mask, 631
- network password, 632
- network password, *see* Secure RPC password,
- networks tables (NIS+), 614
 - columns, 614
- newkey, 79
- NFS file servers and FNS, 403
- nicknames files, 291
- NIS, 9, 281, 319, 632
 - See also* error messages,
 - See also* NIS maps,
 - 4.x compatibility, 295
 - architecture, 282
 - binding, 292
 - binding, broadcast, 293
 - binding, server-list, 292
 - broadcast binding, 293
 - C2 security, 318
 - client problems, 537
 - clients, 284, 285
 - clients, FNS and, 328, 368
 - clients, not bound to server, 538
 - commands hang, 537
 - components, 285
 - configuration files, modifying, 307
 - crontab, 314
 - daemons, 285
 - daemons, not running, 541
 - DNS and, 318
 - DNS, and, 282, 283
 - domain names, incorrect, 538
 - domain names, missing, 538
 - domains, 282, 285
 - domains, changing, 318
 - earlier versions and, 294
 - /etc/nsswitch.conf files and, 296
 - /etc/nsswitch.conf files and DNS, 296
 - files-based naming on machines in, 283
 - FNS, and, 284, 328, 335, 367
 - FNS, policies and, 399
 - fnssypd, FNS and, 329, 368
 - halting, 320
 - hosts, changing domain of, 318
 - Internet access, 20
 - Internet, and, 283
 - madedbm, 286
 - make, 291
 - Make files, 287
 - makedbm, 291
 - Makefile entries, adding, 308
 - Makefile filtering, 308
 - master servers, 284
 - multihome support, 295
 - ndbm format, 288
 - netgroups, 303, 304
 - NIS+, and, 283
 - NIS+, compared to, 29
 - NIS+, NIS-compatibility mode, 33
 - NIS+, problems with, 512
 - NIS+, using with, 32
 - “not responding” messages, 537
 - NSKit, 294
 - on machines in NIS+ environment, 283
 - Operating systems, different versions
 - of, 320
 - organization namespace (FNS), 378
 - passwd maps auto update, 314
 - passwd maps, updatingpasswd maps, 302
 - passwords, user, 302
 - problems, 537
 - root entry, 300
 - root reference (FNS), 446, 447
 - rpc.yppasswdd, 286, 302
 - rpc.yupdated, 286
 - securenets, 295
 - security, 295, 300
 - server binding not possible, 540
 - server-list binding, 293
 - servers, 284
 - servers not available, 538
 - servers, malfunction, 541
 - servers, maps different versions, 542
 - servers, overloaded, 541
 - SKI, FNS and, 329, 368
 - slave server, adding, 316
 - slave servers, 284

- slave servers, initializing, 317
- software installation, 294
- starting, 294
- stopping, 294, 320
- structure of, 282
- subdomains in NIS+ environment, 283
- SunOS 4.x compatibility, 295
- SUNWypcr, 294
- SUNWypcu, 294
- “unavailable” messages, 537
- updates, automating, 314
- updating via shell scripts, 314
- user password locked, 301
- useradd, 300
- userdel, 301
- users, adding, 300
- users, administering, 300
- utility programs, 286
 - /var/yp/, 287
- versions, earlier, 294
- ypbind, 286, 291, 293
- ypbind fails, 540
- ypbind “can’t” messages, 537
- ypcat, 286, 291
- ypinit, 286, 291
- ypmatch, 286, 291
- yppoll, 286
- yppush, 286, 291
- ypserv, 286, 291, 293
- ypservers files, 316
- ypset, 286, 291
- ypstart, 294
- ypstop, 294
- ypupdated, 294
- ypwhich, 286, 291, 294
- ypwhich inconsistent displays, 539
- ypxfr, 286, 291
- NIS compatibility mode
 - NIS on individual machines, 283
 - rpc.nisd starting, 195, 196
- nis dump result nis_perror messages (NIS+), 536
- NIS maps, 287, 288, 632
 - administering, 304
 - CHKPIPE in Makefile, 309
 - commands related to, 291
 - configuration files, modifying, 307
 - crontab, 314
 - default, 288
 - default maps, modifying, 311
 - descriptions of, 288
 - displaying contents, 304
 - displaying contents of, 290
 - format is ndbm, 288
 - locating, 290
 - Make files, 287
 - Makefile and, 307
 - Makefile entries, adding, 308
 - Makefile entries, deleting, 309
 - Makefile entries, updating, 310
 - Makefile filtering, 308
 - Makefile macros, changing, 310
 - Makefile variables, changing, 310
 - Makefile, DIR variable, 310
 - Makefile, DOM variable, 310
 - Makefile, PWDIR variable, 310
 - making, 290
 - new maps, creating, 312
 - new maps, creating from files, 312
 - new maps, creating from keyboard, 312
 - nicknames, 291
 - non-default, 310
 - non-default maps, modifying, 311
 - NOPUSH in Makefile, 309
 - overview, 287
 - propagating, 313
 - server, changing, 305
 - updates, automating, 314
 - updating, 290
 - updating via shell scripts, 314
 - /var/yp/, 287
 - working with, 290
 - yppush in Makefile, 309
 - ypxfr, crontab file in, 314
 - ypxfr, invoking directly, 315
 - ypxfr, logging, 315
 - ypxfr, shell scripts in, 314
- NIS+, 9, 27, 632
 - See also* error messages,
 - See also* nisclient script,
 - See also* nispopulate script,
 - See also* nisserver script,
 - See also* tables (NIS+),
 - Abort_transaction: Internal database error messages, 511

- access, 74
- access rights, 78
- administration, problems, 507
- administrator, 78
- API, 36
- attempt to remove a non-empty table
 - messages, 510
- authentication, 33, 69, 70, 72
- Authentication denied messages, 519
- Authentication error messages, 519
- authorization, 32, 70, 74
- authorization classes, 75, 77
- automounter, unable to use, 516
- blanks in names, 516
- Busy try again later messages, 528
- cache manager, 48
- cache manager, missing, 530
- Callback: - select failed messages, 511
- CALLBACK_SVC: bad argument
 - messages, 511
- Can't find suitable transport messages, 514
- Cannot find messages, 514
- Cannot get public key messages, 519
- Cannot remove replica messages, 510
- "Cannot [do something] with log" type
 - messages, 532
- Changing Key messages, 518
- checkpoint fails, 508
- Chkey failed messages, 519
- clients, 47
- cold-start files, 48
- commands, 34
- commands, FNS and, 413
- Corrupt database messages, 511
- Corrupt log messages, 511
- cred table entries missing, 525
- credentials, 72
- Database corrupted messages, 511
- de-bugging, 506
- directories, 42
- directories (UNIX), 39
- directories, cannot delete, 510
- directory cache, 48
- directory name error messages, 509
- directory names, 54
- disk space insufficient, 509
- disk space, insufficient, 532
- domain name error messages, 509
- domain name, changing, 526
- domain names, 53
- domains, problems with, 515
- entry corrupt messages, 511
- files, 39
- files, problems with, 516
- FNS, and, 328, 334, 367
- FNS, disk space, 334
- FNS, domains, 334
- FNS, host namespace, 398
- FNS, organization names, 398
- FNS, organization namespace, 397
- FNS, security and, 398
- FNS, upgrading from NIS, 417
- FNS, user namespace, 398
- full dump rescheduled messages, 536
- fully-qualified names, 52
- Generic system error messages, 509
- group class, 75, 76
- group names, 55
- groups, can't add users, 508
- groups_dir, cannot delete, 510
- host names, 56
- Illegal object type messages, 507
- Insufficient permission messages, 517, 519
- insufficient permission messages, 527
- Internet access, 20
- Invalid principal name messages, 512, 513
- Keyserv fails to encrypt messages, 519
- links to tables, 517
- Log corrupted messages, 511
- log entry corrupt messages, 511
- Login incorrect messages, 519
- Login Incorrect messages, 520
- login, user cannot, 533
- logs, cannot truncate, 509
- logs, too large, 509, 530
- machines, moving to new domain, 526
- memory, insufficient, 532
 - messages, 536
- name expansion, 57
- names, allowable characters, 57
- namespace structure, 40
- naming conventions, 52
- NIS compatibility, problems, 512
- nis dump result nis_perror messages, 536
- NIS machines in, 283

- NIS+, policies and, 397
- NIS, and, 283
- NIS, compared to, 29
- NIS, using with, 32
- NIS-compatibility mode, 33, 69
- nisinit fails, 508
- NIS_DUMPLATER, 536
- NIS_PATH variable, 57
- No memory messages, 532
- No public key messages, 519
- nobody class, 75, 77
- Not exist messages, 514
- not exist, messages and problems, 515
- Not found messages, 514
- not have secure RPC credentials
 - messages, 517, 518
- Not responding messages, 528
- nsswitch.conf files, 33
- objects, FNS and, 413
- “object problem” messages, 507
 - one replica is already resyncing messages, 536
- organization namespace (FNS), 378
- org_dir, cannot delete, 510
- Out of disk space messages, 532
- owner class, 75, 76
- ownership problems, 517
- partially-qualified names, 52
- password commands, 71
- password expired messages, 520
- passwords different, 526
- passwords in /etc/passwd, 526
- passwords, cannot change, 535
- passwords, login fails, 513
- passwords, new, cannot use, 534
- performance, problems, 528
- Permission denied messages, 512, 513
- permission denied messages, 518
- Permission denied messages, 519
- permission denied messages, 524, 527
- Permission denied messages, 534
- permission problems, 517
- Possible loop detected in namespace
 - messages, 509
- principal names, 56
- principals, 33, 47
- processes, insufficient, 533
- queries hang, 531
- recursive groups, 529
- replicas, cannot remove directories, 510
- replicas, lagging, 515
- replicas, out of synch, 515
- replicas, too many, 529
- replicas, update failure, 535
- replica_update: messages, 536
 - rescheduling the resync messages, 536
- resource problems, 532
- rlogin, user cannot, 534
- root password change, problems, 528
- .rootkey files, pre-existing, 527
- rpc.nisd dead, 530
- rpc.nisd, failure of, 512
- rpc.nisd, problems, 511
- search paths, 62
- security, 32
- security commands, 79
- Security exception messages, 517, 518
- security levels, 71
- security overview, 69
- security, problems, 519
- Server busy. Try Again messages, 531
- servers, 44
- servers (masters), 45
- servers (replicas), 45
- servers as clients, 51
- servers in parent domain, 51
- servers, slow startup, 530
- switch file problems, 513
- table entry names, 55
- table names, 55
- table paths, 529
- table setup, 64
- table structure, 59
- table updates, 65
- tables, 31, 59
- testing, 535
- time-to-live, 49
- transaction log, 45
- troubleshooting, 506
- Unable to find messages, 514
- Unable to fork messages, 532
- Unable to make request messages, 517
- UNABLE TO MAKE REQUEST
 - messages, 518
- Unable to stat messages, 514, 517

- Unknown user messages, 512, 513
- updates, 45
- user problems, 533
- Wide Area Networks and, 242
- world class, 75, 77
- NIS+ cache
 - See also* cache manager,
 - See also* nisshowcache,
 - contents, displaying, 200
- NIS+ daemon, *see* rpc.nisd daemon,
- NIS+ directories, 186
 - checkpointing, 200, 202
 - contents, listing, 188
 - creating, 189, 190
 - domains, expanding into, 235
 - niscat, 186
 - nisinit, 197
 - nisls, 187
 - nismkdir, 189
 - nisping, 200, 202
 - nisping, forcing, 201
 - nism, 194
 - nismrdir, 192
 - nisshowcache, 199
 - nis_cachemgr, 199
 - non-root, creating, 189
 - objects, removing, 194, 195
 - properties, displaying, 186
 - removing, 193
 - replicas, adding, 191, 192
 - replicas, creating, 190
 - replicas, disassociating from, 193
 - root, creating, 189
 - transaction log, 203
- NIS+ environment, 151, 632
- NIS+ groups, 174
 - creating (NIS+), 179
 - deleting (NIS+), 180
 - explicit members, 176
 - explicit non-members, 176
 - implicit members, 176
 - implicit non-members, 176
 - member types (NIS+), 175
 - members, adding (NIS+), 180, 181
 - members, listing, 177
 - members, listing (NIS+), 181
 - members, removing (NIS+), 182
 - members, testing (NIS+), 182
- NIS+, 174
 - nistbladm, and, 215
 - NIS_DEFAULTS, and, 179
 - NIS_GROUP, setting, 180
 - non-members, 176
 - properties, displaying, 177
 - recursive members, 176
 - recursive non-members, 176
 - recursive, performance degradation, 529
 - removing, 194
 - specifying (NIS+), 176
 - syntax (NIS+), 176
 - users, cannot add, 508
- NIS+ object, 632
- NIS+ principal, 632
- NIS+ tables, 31, 210, 604
 - automount, additional, 218
 - auto_home tables, 605
 - auto_master tables, 605, 606
 - bootparams tables, 606
 - client_info tables, 608
 - column security, 121
 - columns, components, 216
 - columns, searching, 232, 233
 - columns, specifying, 216
 - columns, types of, 217
 - contents, displaying, 226
 - creating, 216
 - cred table, displaying contents, 227
 - cred table, links not allowed, 234
 - cred tables, 608
 - deleting, 218
 - emptying, 194
 - entries, adding, 218, 219, 221
 - entries, editing, 222, 224
 - entries, links not allowed, 233
 - entries, modifying, 222
 - entries, null termination of, 218
 - entries, removing, 224, 225
 - entry security, 121
 - ethers tables, 609
 - files, dumping data to, 240
 - files, transferring data from, 237
 - group tables, 610
 - hosts tables, 610
 - input file format, 604
 - links, 233

- links, do not work, 517
- mail_aliases tables, 611
- maximum size of, 219
- name services, other, 604
- netgroup tables, 612
- netmasks tables, 613
- networks tables, 614
- NIS maps, transferring data from, 238
- nisaddent, 234, 236
- niscat, 226
- nisgrep, 229
- nisln, 233
- nismatch, 229
- nissetup, 234
- nistbladm, 210
- null termination of entries, 218
- operators, and, 230
- passwd tables, 614
- problems with, 508
- properties, displaying, 228
- protocols tables, 616
- regular expressions, in, 230
- rpc tables, 617
- security, 120, 121
- security and levels, 123
- services tables, 617
- table paths, performance degradation, 529
- timezone tables, 618
- transferring data, 236
- transferring data, options, 236
- wildcards, and, 230
- NIS+ tables (NIS+)
 - auto_home tables, 605
- NIS+ transaction log, 632
- NIS+, *see* backup-restore (NIS+)
 - backup,
 - restore,
- NIS+, *see* credentials
 - credentials,
- NIS+, *see* keys
 - keys,
- NIS+, *see* NIS+ cache
 - cache,
- NIS+, *see* NIS+ directories
 - directories,
- NIS+, *see* NIS+ tables
 - tables,
- NIS+, *see* removing NIS+
 - removing,
- NIS+, *see* server preference (NIS+)
 - server preference,
- NIS+, *see* TTL
 - time-to-live,
 - TTL,
- NIS, *see* NIS maps
 - maps,
- NIS-compatibility mode, 632
- nisaddcred, 79, 96, 97, 100, 103, 104
 - changing keys, 109 to 111
 - creating credentials, 96, 100 to 103
 - credential administration, 103
 - credentials, how created, 97
 - modifying credentials, 96
 - removing credentials, 96, 104
 - time stamp, 87
 - uninstalling NIS+, 274
 - updating credentials, 103
- nisaddent, 65, 135, 234, 236
 - automount tables and, 238
 - data transfer options, 236
 - files, data from, 237
 - files, data to, 240
 - NIS maps, data from, 238
 - passwdfiles and, 238
 - syntax, 236
 - tables, non-standard and, 238
- nisbackup, 261, 262, 270
 - See also* backup-restore (NIS+),
 - automating, 265
 - backup directory structure, 265
 - backup files, 266
 - directories, individual back up, 265
 - file-system backup, and, 264
 - interruptions, 262
 - master server only, 263
 - namespace, entire, 265
 - options, 263
 - over-writing, 264
 - syntax, 262
- niscat, 97, 125, 186, 213, 219, 225, 226
 - cred table, displaying contents, 227
 - directory properties, 186
 - FNS, and, 414
 - group members, 177
 - group properties, 177

- *NP*, 227
- object properties, displaying, 228
- options, 227
- Server busy. Try Again messages, 531
- syntax, 226
- nischgrp, 34, 140, 174
 - FNS, and, 415
 - group, changing, 140, 141
- nischmod, 34, 120, 135
 - access rights, adding, 136
 - access rights, removing, 136
 - FNS, and, 415
- nischown, 34, 139, 152
 - FNS, and, 415
 - ownership, changing, 139
- nischttl, 34, 49, 205, 206
 - keys, updating, 114
- nisclient, 86, 95, 112, 114, 275, 519
 - uninstalling NIS+, 273
- nisdefaults, 34, 131, 132, 525
 - display options, 131
 - time-to-live, 205
 - TTL, 205
- nisgrep, 34, 229
 - operators, and, 230
 - options, 232
 - regular expressions, in, 230
 - searching, first column, 232
 - searching, multiple columns, 233
 - searching, specific column, 232
 - syntax, 231
 - wildcards, and, 230
- nisgrpadm, 34, 174, 178, 181, 215
 - access rights, 178
 - group members, listing, 177
 - group properties, displaying, 177
 - groups, creating, 179
 - groups, deleting, 180
 - members, adding, 180, 181
 - members, listing, 181
 - members, removing, 182
 - members, testing, 182
 - problems with, 508
 - removing groups, 194
 - syntax, 176
 - syntax for group members, 179
 - syntax for groups, 178
- nisinit, 34, 40, 114, 197, 516, 532
 - client, initializing, 197
 - root directories, 189
 - root master, initializing, 198
 - uninstalling NIS+, 274
- nisinitproblems with, 508
- nisln, 34, 233
 - creating links, 234
 - cred table and, 234
 - options, 234
 - syntax, 234
 - table entries and, 233
- nislog, 34, 203
 - options, 204
 - transaction log, displaying, 204
- nisl, 34, 174, 187, 188, 516, 550
 - directories, contents of, 187, 188
 - FNS, and, 414
- nismatch, 34, 97, 104, 229
 - Changing Key messages, 519
 - operators, and, 230
 - options, 232
 - regular expressions, in, 230
 - searching, first column, 232
 - searching, multiple columns, 233
 - searching, specific column, 232
 - syntax, 231
 - wildcards, and, 230
- nismkdir, 34, 64, 135, 189, 190
 - directories, non-root creating, 189, 190
 - master server only, 190
 - non-root directories, creating, 189
 - replicas, adding, 191
 - replicas, creating, 190
 - root directories, cannot create, 189
- nisspasswd, 71, 146, 149
- nisping, 34, 65, 201, 203
 - checkpoint fails, 508
 - directories, checkpointing, 200, 202
 - forcing, 201
 - performance, effect on, 529
 - replicas, adding, 192
 - replicas, disassociating from directory, 193
 - updates, last, 201
- nispopulate, 65, 96
- nisprefadm, 34, 245 to 247, 251, 257, 260
 - activating, 260
 - changing preference numbers, 254

- client names, 246
- client_info tables, 608
- displaying preferences, 246, 249, 250
- ending, 259
- global preferences, specifying, 250 to 253
- global table, 243
- list, replacing, 256
- local file, 243
- local preferences, specifying, 253
- local to global, 259
- options, 247
- preferences, ending use of, 258
- preferred servers, designating, 242
- Preferred-Only Servers, abandoning, 257
- Preferred-Only Servers, specifying, 257
- rank numbers, 244, 245
- Rank Numbers, specifying, 250
- server names, 246
- server preferences, modifying, 254
- server-use, overview, 243
- servers, removing from list, 255
- servers, replacing in list, 254
- single client, 246
- subnet, 246
- syntax, 247
- nisprefadm, *see* client_info files and tables
 - client_info files and tables,
- nisrestore, 34, 267, 268
 - See also* backup-restore (NIS+),
 - directory names, 269
 - directory, restoring, 268
 - lookup error, 269
 - namespace corrupted, restoration of, 269
 - options, 268
 - prerequisites, 267
 - procedures, 268
 - replica setup, 269
 - resolve.conf files, 267
 - rpc.nisd and, 267
 - servers, replacing, 270
 - syntax, 268
- nism, 34, 194, 195
- nismdir, 34, 192
 - cannot delete directories, 510
 - directories, removing, 193
 - objects, removing, 194, 195
 - replicas, disassociating, 193
- nissserver, 189, 277
 - replicas (NIS+), setup, 269
- nissserver script, 64
- nissetup, 64, 234
 - directories into domains, 235
- nisshowcache, 34, 49, 199
 - contents, displaying, 200
- nisstat, 34
- nistbladm, 34, 64, 120, 135, 137, 168, 210, 213, 224, 529
 - access rights, columns, 137 to 139
 - automount tables, additional, 218
 - column values, 212
 - columns, components, 216
 - columns, null termination of, 218
 - columns, specifying, 216
 - columns, types, 217
 - creating a table, 216
 - days, number of, 156
 - emptying tables, 194
 - entries, adding, 218, 219
 - entries, editing, 222, 224
 - entries, forcing, 221
 - entries, identical, 220
 - entries, modifying, 222
 - entries, multiple, 220
 - entries, over-writing, 221
 - entries, removing, 224, 225
 - expire values, 156
 - groups, and, 215
 - groups, NIS+ and, 175
 - inactive days, setting, 167, 168
 - inactive values, 155
 - indexed names, 215
 - keys, 213
 - max values, 155
 - min values, 154
 - netgroups, and, 215
 - NIS+ groups, and, 174, 215
 - options, 211
 - password aging, 165
 - password expiration, setting, 166
 - password expiration, unsetting, 167
 - passwords, and, 153
 - searchable columns, 213
 - shadow column fields, 154
 - syntax, 210
 - tables, deleting, 218

- UNIX groups, and, 215
- unused values, 156
- warn values, 155
- nistest, 34
- nisupdkeys, 34, 79, 92, 110 to 112, 114
 - arguments, 113
 - cold-start files, and, 112
 - updating keys, examples, 112, 113
 - updating stale keys, 522
- nis_cachemgr, 34
 - See also* cache manager,
 - uninstalling NIS+, 274 to 276
- nis_cachmgr, 92
- nis_checkpoint, 34
- NIS_COLD_START files
 - servers (NIS+) replacing, 270
- NIS_COLD_START files, *see* cold-start files,
- NIS_DEFAULTS, 120, 126, 131, 133
 - displaying value of, 133
 - resetting, 134
- \$NIS_DEFAULTS, 134
- NIS_DUMPLATER, 536
- NIS_GROUP, 126, 180
- NIS_OPTIONS
 - de-bugging, 506
 - options, 506
- NIS_PATH
 - performance, effect on, 529
 - problems with, 514
- NIS_PATH variable, 57
- NIS_SHARED_DIRCACHE files, 199
- NIS_SHARED_DIRCACHE files
 - See also* cache manager,
- NNSP, 632
- No memory messages (NIS+), 532
- no permission messages (FNS), 550
- No public key messages (NIS+), 519
- No such... messages (DNS), 548
- nobody class, 75, 77, 119
- Non-authoritative answer messages
 - (DNS), 548
- NOPUSH in Makefile, 309
- Not exist messages (NIS+), 514
- Not found messages (NIS+), 514
- not have secure RPC credentials messages
 - (NIS+), 517, 518
- Not responding messages (NIS+), 528
- “not responding” messages (NIS), 537

- *NP*, 227
- npc.nisd, 195
- nscd
 - uninstalling NIS+, 275, 276
- NSID contexts
 - creation, 429
- NSKit, *see* NIS,
- nsswitch.conf, xxx
- nsswitch.conf files, 8, 11, 16, 17, 22, 33, 149, 499, 545
 - +/- Syntax, 21
 - actions, 14
 - Auto_home table, 16
 - Auto_master table, 16
 - comments in, 16
 - compat, 21
 - continue, 14
 - default file, 20
 - default template files, 17
 - DNS and NIS, 20
 - DNS and NIS+, 20
 - DNS, and, 12, 20, 467
 - examples, 18, 19
 - FNS, and, 22
 - FNS, consistency with, 22
 - FNS, updates, 23
 - format of, 12
 - incorrect syntax, 16
 - information sources, 13
 - Internet access, 20
 - keyserver entry, 17
 - messages, status, 14
 - missing, 16
 - modifying, 15
 - NIS, 283
 - NIS and, 296
 - NIS and DNS, 296
 - NIS+ - NIS compatibility problems, 512
 - NIS+ problems with, 514
 - NIS+, and, 33
 - NOTFOUND=continue, 15
 - nsswitch.files files, 17
 - nsswitch.nis files, 17
 - nsswitch.nisplus files, 17
 - options, 14
 - passwd_compat, 21
 - password data, 22

- passwords, and, 534
- problems, 513
- publickey entry, 17
- return, 14
- search criteria, 13, 15
- sources, 13
- status messages, 14, 15
- SUCCESS=return, 15
- templates, 12, 17
- timezone table, 16
- TRYAGAIN=continue, 15
- UNAVAIL=continue, 15
- uninstalling NIS+, 277
- nsswitch.nis, 18
- null termination, 218

O

- “object problem” messages (NIS+), 507
- one replica is already resyncing messages (NIS+), 536
- Operation Failed message (FNS), 553
- org contexts, 424
- org//, *see* orgunit contexts,
- org//service/printer, 332
- organization maps, 367
- organizational unit context, 633
- organizational units, 632
- orgunit contexts, 377
 - creating, 423
 - example, 423
 - names in, 330
 - names, composing in, 382
 - NIS+, and, 423
 - NIS, and, 423
 - population of, 424
- org_dir
 - FNS, and, 367
- org_dir directories, 42, 55, 64, 77, 423
 - FNS mapping to, 413
 - FNS, and, 328
 - uninstalling NIS+, 275, 276
- org_dir directories, 510
- Out of disk space messages (NIS+), 532
- owner class, 75, 76, 118, 119

P

- parent context, 633
- parent domain, 633
- PASSLENGTH, *see* passwords, PASSLENGTH,
- passwd, 71, 79, 81, 147, 149, 150, 157, 160, 171, 302, 513
 - access rights, 152
 - age limit, 162
 - aging, turning off, 165
 - changing passwords, 146, 159
 - credentials, and, 152
 - data, displaying, 157
 - domains, other, 153
 - forcing users to change, 162, 165
 - keys, and, 153
 - locking passwords, 160
 - minimum life, setting, 163
 - NIS map auto updated, 314
 - NIS+ environment, 151
 - nispasswd, and, 149
 - password aging, 161
 - password aging limitations, 161
 - permissions, and, 152
 - rlogin problems, 534
 - root's, changing, 147
 - unlocking passwords, 161
 - user cannot change password, 535
 - user problems, 533
 - vacation locks, 160
 - warning period, setting, 164
 - yppasswd, and, 150
- passwd files
 - 4.x compatibility (NIS), 295
 - nisaddent, and, 238
 - Solaris 1.x formats, 300
 - users, adding (Solaris 1.x), 301
- passwd files, *see* password data,
- passwd maps
 - users, adding, 301
- passwd maps, *see* password data,
- passwd tables, 390
- passwd tables (NIS+), 614
 - +/- syntax, 616
 - columns, 615
 - shadow column data, 615
- passwd tables, *see* password data,
- passwd.adjunct files, 303, 307, 318

- passwd.byname maps
 - FNS, and, 399
- passwd.org_dir tables
 - FNS, and, 398
- password
 - secure and login different, 526
- password commands, 71
- password data
 - See also* security,
 - days, number of, 156
 - displaying, 157
 - expire values, 156
 - inactive values, 155
 - login different from secure, 91
 - login password, 91
 - max values, 155
 - min values, 154
 - NIS, and, 300
 - nsswitch.conf file, and, 150
 - nsswitch.conf file, over-riding, 151
 - nsswitch.conf files, 22
 - root in NIS maps, 300
 - secure different from login, 91
 - secure RPC password, 91
 - shadow column fields, 154
 - unused values, 156
 - warn values, 155
- password expired Message, 145
- password expired messages (NIS+), 520
- password will expire Message, 145
- passwords
 - See also* password data,
 - administering, 149
 - age limiting, 162
 - aging, 161, 166
 - aging limitations, 161
 - aging, turning off, 165
 - changing, 146, 159
 - choosing, 147
 - criteria, setting defaults, 168
 - default criteria, setting, 168
 - expiration of privilege, setting, 166
 - expiration of privilege, unsetting, 167
 - forcing users to change, 162, 165
 - inactive days, setting, 167, 168
 - locking, 160
 - logging in, 144
 - login fails after change, 513
 - login failures, maximum, 171
 - Login incorrect Message, 144
 - login, maximum time for, 172
 - maximum tries, 171
 - MAXWEEKS, 169, 170
 - minimum length, setting, 171
 - minimum life, setting, 163
 - MINWEEKS, 169, 170
 - new, cannot use, 534
 - NIS+ environment, 151
 - NIS, and, 302
 - nistbladm, 153
 - not have secure RPC credentials,
 - messages, 518
 - nsswitch.conf file, and, 149, 150, 534
 - nsswitch.conf file, over-riding, 151
 - PASSLENGTH, 171
 - passwd, 150
 - password expired Message, 145
 - Permission denied Message, 146
 - privileges (user), 165, 166
 - requirements, 147
 - rlogin problems, 534
 - root change, problems, 528
 - root's, changing, 147
 - rpc.yppasswdd (NIS), 302
 - Sorry: less than Message, 146
 - unlocking, 161
 - user cannot change, 535
 - user problems, 533
 - using, 144
 - vacation locks, 160
 - warning period, setting, 164
 - WARNWEEKS, 169, 170
 - weeks, maximum (default), 169
 - weeks, minimum (default), 169
 - weeks, warning (default), 169
 - will expire Message, 145
- performance
 - NIS+ server search, 242
- Permission denied Message, 146
- Permission denied messages (NIS+), 512, 513
- permission denied messages (NIS+), 518
- Permission denied messages (NIS+), 519
- permission denied messages (NIS+), 524, 527
- Permission denied messages (NIS+), 534
- ping, 541

- pinging, 633
- populate tables, 634
- populating NIS+ tables, *see* NIS+ tables,
- Possible loop detected in namespace messages (NIS+), 509
- preference rank number, 633
- preference rank numbers, *see* client_info files and tables,
- preferred server, 633
- preferred server list, 633
- principal, 633
- principals, 33
- printer contexts, 381
 - creating, 343
 - creation, 427
- printers.conf files, 344
- private key, 633
- processes, insufficient (NIS+), 533
- protocols tables, 616
- protocols tables (NIS+), 616
 - columns, 616
- public key, 634
- PWDIR/security/passwd.adjunct files, 318
- \$PWDIR/security/passwd.adjunct, 307
- \$PWDIR/shadow, 295

R

- rcp, 365, 543
 - NIS maps, transferring, 315
- rdist
 - NIS maps, transferring, 315
- read rights, *see* security, access rights,
- record, 634
- reference, 634
- reference registry (FNS), 381
- remote procedure call, *see* RPC,
- removing NIS+, 273
 - client, from, 273
 - namespace, from, 276
 - server, from, 274
- replica server, 634
- replica servers, *see* NIS+,
- replica_update: messages (NIS+), 536
- rescheduling the resync messages (NIS+), 536
- resolv.conf files, 499
 - NIS and Internet, 319
- resolve.conf files, 267
- resolver, 467
- resolver, *see* DNS, resolver,
- resource record, 481
- restore (NIS+), *see* backup-restore (NIS+),
- reverse resolution, 634
- rlogin, 534
 - problems, 534, 548
 - user problems, 533
- root context, 634
- root domain, 634
- root master server, 634
- root reference, *see* FNS, root reference,
- root replica server, 634
- root servers
 - See also* servers,
 - initializing, 198
- root.cache files (DNS), *see* named.ca files,
- root.object files (NIS+), 266
- .rootkey files, 274, 527
 - pre-existing, 527
 - servers (NIS+) replacing, 270
 - uninstalling NIS+, 276
- root_dir files (NIS+), 266
- rows (table), *see* tables, entries,
- RPC, 634
- rpc files, 196
- rpc tables (NIS+), 617
 - columns, 617
 - example, 617
- rpc.nisd, 40, 196, 269, 270, 530, 532
 - dies, 530
 - DNS forwarding, 195, 196
 - EMULYP -Y -B, 196
 - failure of, 219
 - multiple parent processes, 511
 - NIS-compatibility mode, 195, 196
 - options, 196
 - restore (NIS+), and, 267
 - security level, default, 195
 - stopping, 197
 - table size and, 219
 - uninstalling NIS+, 275, 276
- rpc.nispasswd
 - maximum password tries, 171
 - password login failures, 171
 - password maximum login time, 172
- rpc.yppasswdd, 302, 303

- 4.x compatibility (NIS), 296
- passwd updates maps, 314
- rpc.yppasswdd daemon, 286
- rpc.yppupdated daemon, 286
- rsh
 - problems, 548

S

- scripts (NIS+), *see* NIS+
- secure RPC netname, 87, 89
 - description of, 98
 - principal name, and, 98
- Secure RPC password, 634
- securenets files, 295
- security
 - See also* password data,
 - access, 74
 - access rights, 118 to 120
 - access rights (create), 124
 - access rights (destroy), 124
 - access rights (modify), 124
 - access rights (read), 124
 - access rights, adding, 136
 - access rights, changing, 120, 135
 - access rights, column (table), 121
 - access rights, columns, 137 to 139
 - access rights, combining, 119
 - access rights, command specifying, 127
 - access rights, concatenation, 119
 - access rights, default, 120
 - access rights, granting, 127
 - access rights, levels, 123
 - access rights, non-default, 135
 - access rights, removing, 136
 - access rights, syntax, 128 to 131
 - access rights, table, 120
 - access rights, table entries, 121
 - access rights, tables, 121, 123
 - access rights, viewing, 125
 - administrator, 78
 - administrator's credentials, 99, 101
 - authentication, 69, 70, 72
 - authorization, 70, 74
 - authorization classes, 75, 77
 - C2 security, NIS and, 318
 - commands, access specification in, 127
 - create rights, 124

- credentials, 72
- defaults, changing, 134
- defaults, displaying, 131
- defaults, setting, 133
- destroy rights, 124
- expire values, 156
- group class, 75, 76
- groups, changing, 140, 141
- inactive values, 155
- keys, 105
- levels (NIS+), 71
- max values, 155
- min values, 154
- modify rights, 124
- NIS, 295
- NIS+ access rights, 78
- NIS+ commands, 79
- NIS+ overview, 67, 69
- NIS+, and, 32
- NIS, and, 300
- NIS, C2 security and, 318
- NIS-compatibility mode, 69
- nobody class, 75, 77
- owner class, 75, 76
- ownership, changing, 139
- password commands, 71
- read rights, 124
- root in NIS maps, 300
- secure RPC netnames, 98
- securenets files, 295
- servers, granting access, 127
- shadow column fields, 154
- specifying non-default access rights, 135
- syntax, access rights, 128 to 131
- table columns, 121
- table entries, 121
- tables, 120, 121
- tables and levels, 123
- unused values, 156
- warn values, 155
- world class, 75, 77
- Security exception messages (NIS+), 517, 518
- security, *see* credentials
 - credentials,
- security, *see* keys
 - keys,
- sed, 312

- sendmail
 - mail_aliases Table, and, 611
- server, 635
- Server busy. Try Again messages (NIS+), 531
- server failed message (DNS, 547
- server list, 635
- server preference (NIS+), 241, 246
 - activating, 260
 - all servers, 245
 - cache manager and, 246
 - cache manager required, 243
 - changing numbers, 254
 - client names, 246
 - client search behavior, 242
 - default, 245
 - displaying, 246, 249, 250
 - ending, 259
 - ending use of, 258
 - global, 243
 - global, specifying, 250 to 253
 - list, replacing, 256
 - local, 243
 - local to global, 259
 - local, specifying, 253
 - modifying, 254
 - preferred only servers, 245
 - preferred servers, designating, 242
 - Preferred-Only Servers, abandoning, 257
 - Preferred-Only Servers, specifying, 256, 257
 - rank numbers, 244, 245
 - Rank Numbers, specifying, 250
 - server names, 246
 - server-use, overview, 243
 - servers, removing from list, 255
 - servers, replacing in list, 254
 - single client, 246
 - subnet, 246
 - when take effect, 246
- servers
 - access rights, granting of, 127
 - DNS, 474
 - DNS, types of, 467
 - NIS slave, adding, 316
 - NIS slaves, initializing, 317
 - NIS+, 44
 - NIS+ master, 45
 - NIS+ replicas, 45
 - NIS+ replicas, adding, 191
 - NIS+ replicas, checkpointing, 200
 - NIS+ replicas, creating, 190
 - NIS+ replicas, last update, 201
 - NIS+, domains they reside in, 191
 - not available (NIS), 538
 - replacing (NIS+), 270
 - replicas (NIS+), setup by restore, 269
 - ypservers files, 316
- servers, *see* root servers
 - root servers,
- servers, *see* server preference (NIS+)
 - server preference,
- service, 332
- service context, 635
- service contexts, 380
 - creation, 426
 - names in, 332
 - names, composing in, 383
 - reference registry, 381
- services tables (NIS+), 617
 - columns, 618
- setenv, 134
- setup
 - replacing NIS+ servers, 270
 - replicas (NIS+), setup by restore, 269
 - tables (NIS+), 64
- setup scripts (NIS+), *see* NIS+
- shadow column, 615
- shadow files
 - See also* password data,
 - NIS and, 295
 - nisaddent, and, 238
 - Solaris 1.x formats, 300
- site context, 635
- site contexts, 379, 428
 - creation, 428
 - names in, 331
 - names, composing in, 383
- sites.byname, 305
- sites.byname files
 - maps, changing server of, 306
- slave server, 635
- slave servers, *see* NIS,
- snoop, 524
- Solaris name services, 8
 - See also* naming,

- Sorry: less than Message, 146
- strong separation, 635
- subcontext, 635
- subnet, 635
- SUNWnskr, 294
- SUNWnsktu, 294
- SUNWyp, 294
- SUNWypu, 294
- switch files
 - nsswitch.nis, 18
- switch files, *see* nsswitch.conf files,
- switch, *see* nsswitch.conf files,
- syslog, 546
- syslog files
 - checkpointing errors, 508
- syslog.conf files
 - error messages, 555

T

- table, 635
- tables (NIS+), 59
 - automount maps, additional, 61
 - columns, 61
 - entries, 61
 - entry names, 55
 - indexed names, 55
 - links, creating, 234
 - names, 55
 - search paths, 62
 - setup, 64
 - structure, 59
 - updates, 65
- tables, *see* NIS+ tables,
- TCP, 635
- TCP/IP, 635
- time stamp, 87, 88
- time-to-live, *see* TTL,
- timezone tables, 16, 618
- timezone tables (NIS+), 618
 - columns, 618
- tmp files
 - disk space, insufficient, 533
- /tmp/CALLS files, 507
- /tmp/temp_file files, 316
- trans.log files, 40, 200
- transaction log
 - contents, displaying, 204

- nislog, 203
- XID, 204
- Transport Control Protocol, 636
- TTL, 205
 - changing, 205
 - nisdefaults, 205
 - objects of, changing, 207
 - options, 206
 - table entries of, changing, 207
 - values, 206

U

- Unable to find messages (NIS+), 514
- Unable to fork messages (NIS+), 532
- Unable to make request messages (NIS+), 517
- UNABLE TO MAKE REQUEST messages (NIS+), 518
- Unable to stat messages (NIS+), 514, 517
- “unavailable” messages (NIS), 537
- uninstalling NIS+, *see* removing NIS+,
- Unknown field messages (DNS), 548
- Unknown user messages (NIS+), 512, 513
- unreachable messages (DNS), 547
- user context, 636
- user contexts, 380
 - all-users, creation of, 425
 - cannot create, 551
 - names in, 331
 - names, composing in, 382
 - single-user, creation of, 426
- User ID 0, 528
- user maps
 - FNS, and, 367
- useradd, 300
 - password is locked, 301
- userdel, 301, 302
- users
 - adding (NIS), 300
 - netgroups, 303, 304
 - NIS, 300
 - passwd maps, updating, 302
 - passwords (NIS), 302
 - useradd, 300
 - userdel (NIS), 301
 - /usr/bin directories, 39
 - /usr/lib directories, 39

- /usr/lib/fn/fn_ctx_initial.so files, 549
- /usr/lib/netsvc/yp directories, 314
- /usr/lib/netsvc/yp/ypstart script
 - NIS security, 300
- /usr/lib/nis, 112, 200
- /usr/lib/nis directories, 39
- /usr/sbin directories, 39
- /usr/sbin/makedbm
 - non-default maps, modifying, 311
- utmp files, 167

V

- /var, 335
- /var/adm/utmp, 167
- /var/fn, 335, 338
- /var/fn directories, 329, 333, 368, 371, 400, 418
- /var/nis, 194, 236
- /var/nis directories, 39, 40
 - old filenames, 516
 - uninstalling NIS+, 275, 277
- /var/nis/client_info, 249
- /var/nis/client_info files and tables, *see*
 - client_info files and tables,
- /var/nis/data.dict, 40
- /var/nis/data directories, 39, 40, 516
- /var/nis/data/trans.log, 200
- /var/nis/NIS_COLD_START
 - servers (NIS+) replacing, 270
- /var/nis/NIS_SHARED_DIRACHE files, 199
- /var/nis/NIS_SHARED_DIRCACHE files, 92
- /var/nis/rep/org_dir directories, 510
- /var/nis/rep/serving_list files, 510
- /var/spool/cron/crontabs/root files
 - NIS, problems, 543
- /var/yp, 239, 538
 - FNS, and, 328
- /var/yp directories
 - NIS security, 300
- /var/yp/, 287, 312
- /var/yp/ directories, 367, 416
 - FNS, and, 400
- /var/yp/binding/ files, 539
- /var/yp/Makefile
 - maps, supported list, 307
- /var/yp/Makefile files
 - 4.x compatibility, 295
- /var/yp/mymap.asc, 312

- /var/yp/nicknames files, 291
- /var/yp/securenets files, 295
- /var/yp/ypxfr.log files, 316

W

- WAN, 636
 - NIS+ and, 242
- warn values, *see* password data,
- WARNWEEKS, *see* passwords,
 - WARNWEEKS,
- weak separation, 636
- wide-area network, *see* WAN,
- workstations, *see* hosts (machines),
- world class, 75, 77, 118, 119

X

- X.500, 636
 - ASN.1, 622, 623
 - client API (FNS), 451
 - FNS syntax, 621
 - FNS, and, 369, 406, 407
 - FNS, federating with, 346, 449
 - LDAP API (FNS), 451
 - nNSReferenceString, 622
 - object classes, FNS, 622
 - objectReferenceString, 622
 - onc_fn_enterprise, 624
 - onc_fn_nisplus_root, 624
 - root reference (FNS), 449
 - XDS/XOM API (FNS), 451
- X/Open Federated Naming, *see* FNS,
- x500.conf files, 451
- XDR encoding (NIS+), 266
- XDS/XOM API, *see* X.500,
- XFN link, 636
- xfn links, *see* FNS,
- XFN, *see* FNS,
- /xfn, 362
- /xfn directories, 403, 404
- XID in transaction log, 204

Y

- ypbind, 291, 293, 305, 541
 - “can’t” messages, 537
 - client not bound, 538

- fails, 540
 - slave servers, adding, 317
- yplib daemon, 286
- yplib “can’t” messages (NIS), 537
- yplib, 21, 286, 290, 291, 297
 - netgroup tables, and, 612
- ypinit, 286, 291, 316
 - default maps, 310
 - slave servers, adding, 317
- ypmatch, 286, 291
- yppasswd, 150
- yppoll, 286
- yppush, 286, 291, 304, 306, 313
 - maps, changing server of, 306
- yppush in Makefile, 309
- yppush maps
 - NIS, problems, 543
- ypserv, 291, 293, 319, 541
 - failure of, 543
 - multihome support, 295
- ypserv daemon, 286
- ypservers, 316
- ypservers files
 - slave server, adding, 316
- ypservers maps
 - NIS, problems, 543
- ypset, 286, 291
- ypstart, 294
- ypstart files, 303
- ypstop, 294, 317
- ypupdate, 33
- ypupdated daemon, 294
- ypwhich, 286, 290, 291, 294
 - display inconsistent, 539
- ypxfr, 33, 286, 291, 312, 542
 - invoking directly, 315
 - logging, 315
 - logging output, 542
 - maps, changing server of, 305, 307
 - shell script, 543
 - shell scripts and, 315
- ypxfr daemon, 286
- ypxfr.log files, 316, 542
- ypxfr_1perday, 314
- ypxfr_1perhour, 314
- ypxfr_2perday, 314
- YP_INTERDOMAIN key, 319

Z

- zone expired messages (DNS), 547
- zones (DNS), *see* DNS,