



Red Hat Enterprise Linux 6 Identity Management Guide

Managing Identity and Authorization Policies for Linux-Based
Infrastructures
Edition 3.0.0

Ella Deon Lackey

Red Hat Enterprise Linux 6 Identity Management Guide

Managing Identity and Authorization Policies for Linux-Based Infrastructures Edition 3.0.0

Ella Deon Lackey
dlackey@redhat.com

Legal Notice

Copyright © 2013 Red Hat.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Identity and policy management — for both users and machines — is a core function for almost any enterprise environment. IPA provides a way to create an identity domain that allows machines to enroll to a domain and immediately access identity information required for single sign-on and authentication services, as well as policy settings that govern authorization and access. This manual covers all aspects of installing, configuring, and managing IPA domains, including both servers and clients. This guide is intended for IT and systems administrators.

Table of Contents

Preface	13
1. Audience and Purpose	13
2. Examples and Formatting	13
2.1. Brackets	13
2.2. Client Tool Information	13
2.3. Text Formatting and Styles	14
3. Giving Feedback	14
4. Document Change History	15
Chapter 1. Introduction to Identity Management	16
1.1. IdM v. LDAP: A More Focused Type of Service	16
1.1.1. A Working Definition for Identity Management	16
1.1.2. Contrasting Identity Management with a Standard LDAP Directory	17
1.2. Bringing Linux Services Together	18
1.2.1. Authentication: Kerberos KDC	19
1.2.2. Data Storage: 389 Directory Server	20
1.2.3. Authentication: Dogtag Certificate System	20
1.2.4. Server/Client Discovery: DNS	20
1.2.5. Management: NTP	21
1.3. Relationships Between Servers and Clients	21
1.3.1. About IdM Servers and Replicas	21
1.3.2. About IdM Clients	23
Part I. Installing Identity Management Servers and Services	26
Chapter 2. Prerequisites for Installation	27
2.1. Supported Server Platforms	27
2.2. Hardware Recommendations	27
2.3. Software Requirements	27
2.4. Supported Web Browsers	27
2.5. System Prerequisites	27
2.5.1. DNS Records	28
2.5.2. Hostname and IP Address Requirements	29
2.5.3. Directory Server	30
2.5.4. System Files	30
2.5.5. System Ports	30
2.5.6. NTP	31
2.5.7. NSCD	31
2.5.8. Networking	32
Chapter 3. Installing an IdM Server	33
3.1. Installing the IdM Server Packages	33
3.2. About ipa-server-install	33
3.3. Example: Running the Script Interactively and Silently	36
3.3.1. Basic Interactive Installation	36
3.3.2. Silent (Non-Interactive) Installation	38
3.4. Examples: Installing with Different CA Configurations	39
3.4.1. Installing with an Internal Root CA	39
3.4.2. Installing Using an External CA	40
3.4.3. Installing without a CA	42
3.5. Example: Configuring DNS Services within the IdM Domain	43
3.5.1. DNS Notes	43
3.5.2. Installing with an Integrated DNS	43

Chapter 4. Setting up IdM Replicas	46
4.1. Planning the Server/Replica Topologies	46
4.2. Prerequisites for Installing a Replica Server	47
4.3. Installing the Replica Packages	48
4.4. Creating the Replica	48
4.5. Alternate Options for Creating a Replica	51
4.5.1. Different DNS Settings	52
4.5.2. Different CA Settings	52
4.5.3. Different Services	52
Chapter 5. Setting up Systems as IdM Clients	53
5.1. What Happens in Client Setup	53
5.2. System Ports	54
5.3. Configuring a Red Hat Enterprise Linux System as an IdM Client	55
5.4. Manually Configuring a Linux Client	58
5.5. Setting up a Linux Client Through Kickstart	63
5.6. Performing a Two-Administrator Enrollment	64
5.7. Manually Unconfiguring Client Machines	64
Chapter 6. Upgrading Identity Management to Red Hat Enterprise Linux 6.4	66
6.1. Upgrade Notes	66
6.2. Upgrading Packages	66
6.3. Removing Browser Configuration for Ticket Delegation (For Upgrading from 6.2)	67
6.4. Testing Before Upgrading the IdM Server (Recommended)	68
Chapter 7. Uninstalling IdM Servers and Replicas	69
Chapter 8. The Basics of Managing the IdM Server and Services	70
8.1. Starting and Stopping the IdM Domain	70
8.2. About the IdM Client Tools	70
8.2.1. The Structure of the ipa Command	71
8.2.1.1. Adding, Editing, and Deleting Entries with ipa	71
8.2.1.2. Finding and Displaying Entries with ipa	72
8.2.1.3. Adding Members to Groups and Containers with ipa	72
8.2.2. Positional Elements in ipa Commands	73
8.2.3. Managing Entry Attributes with --setattr, --addattr, and --delattr	73
8.2.4. Using Special Characters with IdM Tools	74
8.2.5. Logging into the IdM Domain Before Running	74
8.3. Logging into IdM	74
8.3.1. Logging into IdM	74
8.3.2. Logging in When an IdM User Is Different Than the System User	75
8.3.3. Checking the Current Logged in User	75
8.3.4. Caching User Kerberos Tickets	76
8.4. Using the IdM Web UI	76
8.4.1. Supported Web Browsers	76
8.4.2. About the Web UI	76
8.4.3. Opening the IdM Web UI	77
8.4.4. Configuring the Browser	78
8.4.5. Using a Browser on Another System	80
8.4.6. Logging in with Simple Username/Password Credentials	80
8.4.7. Using the UI with Proxy Servers	82
Chapter 9. Identity: Managing Users and User Groups	83
9.1. Setting up User Home Directories	83
9.1.1. About Home Directories	83
9.1.2. Enabling the PAM Home Directory Module	84

9.1.3. Manually Mounting Home Directories	84
9.2. Managing User Entries	85
9.2.1. About Username Formats	85
9.2.2. Adding Users	85
9.2.2.1. From the Web UI	85
9.2.2.2. From the Command Line	87
9.2.3. Editing Users	88
9.2.3.1. From the Web UI	88
9.2.3.2. From the Command Line	90
9.2.4. Deleting Users	91
9.2.4.1. With the Web UI	91
9.2.4.2. From the Command Line	92
9.3. Managing Public SSH Keys for Users	93
9.3.1. About the SSH Key Format	93
9.3.2. Uploading User SSH Keys Through the Web UI	93
9.3.3. Uploading User SSH Keys Through the Command Line	98
9.3.4. Deleting User Keys	98
9.4. Changing Passwords	99
9.4.1. From the Web UI	100
9.4.2. From the Command Line	101
9.5. Activating and Deactivating User Accounts	101
9.5.1. From the Web UI	102
9.5.2. From the Command Line	103
9.6. Unlocking User Accounts After Password Failures	103
9.7. Managing User Private Groups	103
9.7.1. Disabling Private Groups for a Specific User	104
9.7.2. Disabling Private Groups Globally	104
9.8. Managing Unique UID and GID Number Assignments	104
9.8.1. About ID Number Ranges	104
9.8.2. About ID Range Assignments During Installation	105
9.8.3. A Note on Bad ID Ranges	105
9.8.4. Adding New Ranges	105
9.8.5. Repairing Changed UID and GID Numbers	106
9.9. Managing User and Group Schema	106
9.9.1. About Changing the Default User and Group Schema	109
9.9.2. Applying Custom Object Classes to New User Entries	109
9.9.2.1. From the Web UI	109
9.9.2.2. From the Command Line	111
9.9.3. Applying Custom Object Classes to New Group Entries	111
9.9.3.1. From the Web UI	111
9.9.3.2. From the Command Line	113
9.9.4. Specifying Default User and Group Settings	113
9.9.4.1. Viewing Settings from the Web UI	115
9.9.4.2. Viewing Settings from the Command Line	115
9.10. Managing User Groups	116
9.10.1. Creating User Groups	117
9.10.1.1. With the Web UI	117
9.10.1.2. With the Command Line	118
9.10.2. Adding Group Members	119
9.10.2.1. With the Web UI (Group Page)	119
9.10.2.2. With the Web UI (User's Page)	121
9.10.2.3. With the Command Line	123
9.10.2.4. Viewing Direct and Indirect Members of a Group	124
9.10.3. Deleting User Groups	125
9.10.3.1. With the Web UI	125

9.10.3.2. With the Command Line	126
9.11. Searching for Users and Groups	126
9.11.1. Setting Search Limits	126
9.11.1.1. Types of Search Limits and Where They Apply	126
9.11.1.2. Setting IdM Search Limits	127
9.11.1.2.1. With the Web UI	127
9.11.1.2.2. With the Command Line	128
9.11.1.3. Overriding the Search Defaults	129
9.11.2. Setting Search Attributes	129
9.11.2.1. Default Attributes Checked by Searches	129
9.11.2.2. Changing User Search Attributes	129
9.11.2.2.1. From the Web UI	129
9.11.2.2.2. From the Command Line	130
9.11.2.3. Changing Group Search Attributes	130
9.11.2.3.1. From the Web UI	131
9.11.2.3.2. From the Command Line	131
9.11.2.4. Limits on Attributes Returned in Search Results	132
Chapter 10. Identity: Managing Hosts	133
10.1. About Hosts, Services, and Machine Identity and Authentication	133
10.2. Adding Host Entries	134
10.2.1. Adding Host Entries from the Web UI	134
10.2.2. Adding Host Entries from the Command Line	136
10.3. Disabling and Re-enabling Host Entries	137
10.3.1. Disabling Host Entries	137
10.3.2. Re-enabling Hosts	137
10.4. Managing Public SSH Keys for Hosts	138
10.4.1. About the SSH Key Format	138
10.4.2. About ipa-client-install and OpenSSH	138
10.4.3. Uploading Host SSH Keys Through the Web UI	139
10.4.4. Adding Host Keys from the Command Line	143
10.4.5. Removing Host Keys	144
10.5. Renaming Machines and Reconfiguring IdM Client Configuration	145
10.6. Managing Host Groups	146
10.6.1. Creating Host Groups	147
10.6.1.1. Creating Host Groups from the Web UI	147
10.6.1.2. Creating Host Groups from the Command Line	147
10.6.2. Adding Group Members	148
10.6.2.1. Showing and Changing Group Members	148
10.6.2.2. Adding Group Members from the Web UI	148
10.6.2.3. Adding Group Members from the Command Line	149
Chapter 11. Identity: Managing Services	151
11.1. Adding and Editing Service Entries and Keytabs	151
11.1.1. Adding Services and Keytabs from the Web UI	151
11.1.2. Adding Services and Keytabs from the Command Line	153
11.2. Adding Services and Certificates for Services	154
11.2.1. Adding Services and Certificates from the Web UI	154
11.2.2. Adding Services and Certificates from the Command Line	156
11.3. Storing Certificates in NSS Databases	157
11.4. Configuring Clustered Services	157
11.5. Using the Same Service Principal for Multiple Services	158
11.6. Disabling and Re-enabling Service Entries	158
11.6.1. Disabling Service Entries	158
11.6.2. Re-enabling and Services	158

Chapter 12. Identity: Delegating Access to Hosts and Services	160
12.1. Delegating Service Management	160
12.2. Delegating Host Management	161
12.3. Delegating Host or Service Management in the Web UI	161
12.4. Accessing Delegated Services	162
Chapter 13. Identity: Integrating with NIS Domains and Netgroups	164
13.1. About NIS and Identity Management	164
13.2. Setting the NIS Port for Identity Management	165
13.3. Creating Netgroups	166
13.3.1. Adding Netgroups	166
13.3.1.1. With the Web UI	166
13.3.1.2. With the Command Line	167
13.3.2. Adding Netgroup Members	168
13.3.2.1. With the Web UI	168
13.3.2.2. With the Command Line	169
13.4. Exposing Automount Maps to NIS Clients	169
13.5. Migrating from NIS to IdM	170
13.5.1. Preparing Netgroup Entries in IdM	170
13.5.2. Enabling the NIS Listener in Identity Management	171
13.5.3. Exporting and Importing the Existing NIS Data	171
13.5.3.1. Importing User Entries	171
13.5.3.2. Importing Group Entries	172
13.5.3.3. Importing Host Entries	173
13.5.3.4. Importing Netgroup Entries	174
13.5.3.5. Importing Automount Maps	176
13.5.4. Setting Weak Password Encryption for NIS User Authentication to IdM	176
Chapter 14. Identity: Integrating with Active Directory Through Cross-Realm Kerberos Trusts (TECH PREVIEW)	178
14.1. The Meaning of "Trust"	178
14.1.1. How Trust Works: Transparency Between Kerberos and DNS Realms	178
14.1.1.1. Components Involved in Trusts	178
14.1.1.2. Active Directory and Identity Management Directories	179
14.1.1.3. DNS Domains	180
14.1.1.4. Kerberos Realms, Authentication, and Authorization	181
14.1.2. Trust in Contrast to Synchronization	183
14.1.3. Active Directory Users and IdM Features: sudo and Host-Based Access Control Policies	185
14.1.4. Potential Issues with Group Mapping and SIDs	184
14.1.5. Active Directory Users and IdM Administration	186
14.2. Environment and Machine Requirements to Set Up Trusts	186
14.2.1. Supported Windows Platforms	186
14.2.2. Domain and Realm Names	186
14.2.3. NetBIOS Names	186
14.2.4. Integrated DNS	186
14.2.5. Firewalls and Ports	186
14.2.6. Clock Settings	188
14.2.7. Supported Username Formats	188
14.2.8. Trust Can Only Be Configured Once	189
14.3. Setting up Trust with IdM as a DNS Subdomain of Active Directory	189
14.4. Setting up Trust with IdM and Active Directory in Different DNS Domains	195
14.5. Creating IdM Groups for Active Directory Users	200
14.6. Using SSH from Active Directory Machines for IdM Resources	202
14.7. Using Trust with Kerberized Web Applications	202
Chapter 15. Identity: Integrating with Microsoft Active Directory Through	

Synchronization	204
15.1. Supported Windows Platforms	204
15.2. About Active Directory and Identity Management	204
15.3. About Synchronized Attributes	205
15.3.1. User Schema Differences between Identity Management and Active Directory	207
15.3.1.1. Values for cn Attributes	207
15.3.1.2. Values for street and streetAddress	207
15.3.1.3. Constraints on the initials Attribute	208
15.3.1.4. Requiring the surname (sn) Attribute	208
15.3.2. Active Directory Entries and RFC 2307 Attributes	208
15.4. Setting up Active Directory for Synchronization	209
15.4.1. Creating an Active Directory User for Sync	209
15.4.2. Setting up an Active Directory Certificate Authority	209
15.5. Managing Synchronization Agreements	209
15.5.1. Trusting the Active Directory and IdM CA Certificates	209
15.5.2. Creating Synchronization Agreements	211
15.5.3. Changing the Behavior for Syncing User Account Attributes	212
15.5.4. Changing the Synchronized Windows Subtree	214
15.5.5. Configuring Uni-Directional Sync	215
15.5.6. Deleting Synchronization Agreements	216
15.5.7. Winsync Agreement Failures	216
15.6. Managing Password Synchronization	217
15.6.1. Setting up the Windows Server for Password Synchronization	217
15.6.2. Setting up Password Synchronization	219
15.6.3. Allowing Users to Change Other Users' Passwords Cleanly	221
Chapter 16. Identity: Managing DNS	222
16.1. About DNS in IdM	222
16.2. Using IdM and DNS Service Discovery with an Existing DNS Configuration	222
16.3. Setting up DNS After IdM Server Installation	223
16.4. DNS Notes	223
16.5. Configuring DNS Services in Identity Management	223
16.6. Setting up the rndc Service	224
16.7. Managing DNS Zone Entries	224
16.7.1. Adding Forward DNS Zones	224
16.7.1.1. From the Web UI	224
16.7.1.2. From the Command Line	226
16.7.2. Adding Additional Configuration for DNS Zones	226
16.7.2.1. DNS Zone Configuration Attributes	227
16.7.2.2. Editing the Zone Configuration in the Web UI	229
16.7.2.3. Editing the Zone Configuration in the Command Line	230
16.7.3. Adding Reverse DNS Zones	231
16.7.4. Enabling and Disabling Zones	233
16.7.4.1. Disabling Zones in the Web UI	233
16.7.4.2. Disabling Zones in the Command Line	234
16.8. Managing DNS Record Entries	235
16.8.1. Adding Records to DNS Zones	235
16.8.1.1. Adding DNS Resource Records from the Web UI	235
16.8.1.2. Adding DNS Resource Records from the Command Line	237
16.8.1.2.1. About the Commands to Add DNS Records	237
16.8.1.2.2. Examples of Adding DNS Resource Records	238
16.8.2. Deleting Records from DNS Zones	240
16.8.2.1. Deleting Records with the Web UI	240
16.8.2.2. Deleting Records with the Command Line	243
16.9. Configuring the bind-dyndb-ldap Plug-in	243
16.9.1. Changing the DNS Cache Setting	244

16.9.2. Enabling Zone Refreshes and Persistent Searches	244
16.10. Changing Recursive Queries Against Forwarders	245
16.11. Enabling Dynamic DNS Updates	246
16.11.1. Enabling Dynamic DNS Updates in the Web UI	246
16.11.2. Enabling Dynamic DNS Updates in the Command Line	247
16.12. Configuring Forwarders and Forward Policy	247
16.12.1. Configuring Global Forwarders	248
16.12.2. Configuring Zone Forwarders	248
16.12.3. Configuring Forwarder Policy for a Zone	248
16.13. Enabling Zone Transfers	249
16.14. Defining DNS Queries	249
16.15. Synchronizing Forward and Reverse Zone Entries	249
16.16. Setting DNS Access Policies	250
16.17. Resolving Hostnames in the IdM Domain	251
Chapter 17. Policy: Using Automount	252
17.1. About Automount and IdM	252
17.2. Configuring Automount	252
17.2.1. Configuring autofs on Red Hat Enterprise Linux	253
17.3. Setting up a Kerberized NFS Server	254
17.3.1. Setting up a Kerberized NFS Server	254
17.3.2. Setting up a Kerberized NFS Client	257
17.4. Configuring Locations	258
17.4.1. Configuring Locations through the Web UI	258
17.4.2. Configuring Locations through the Command Line	259
17.5. Configuring Maps	260
17.5.1. Configuring Direct Maps	260
17.5.1.1. Configuring Direct Maps from the Web UI	260
17.5.1.2. Configuring Direct Maps from the Command Line	262
17.5.2. Configuring Indirect Maps	263
17.5.2.1. Configuring Indirect Maps from the Web UI	263
17.5.2.2. Configuring Indirect Maps from the Command Line	264
17.5.3. Importing Automount Maps	265
Chapter 18. Policy: Defining Password Policies	267
18.1. About Password Policies and Policy Attributes	267
18.2. Viewing Password Policies	269
18.2.1. Viewing the Global Password Policy	269
18.2.1.1. With the Web UI	270
18.2.1.2. With the Command Line	271
18.2.2. Viewing Group-Level Password Policies	271
18.2.2.1. With the Web UI	272
18.2.2.2. With the Command Line	273
18.2.3. Viewing the Password Policy in Effect for a User	273
18.3. Creating and Editing Password Policies	274
18.3.1. Creating Password Policies in the Web UI	274
18.3.2. Creating Password Policies with the Command Line	276
18.3.3. Editing Password Policies with the Command Line	277
18.4. Managing Password Expiration Limits	277
18.5. Changing the Priority of Group Password Policies	278
18.6. Setting Account Lockout Policies	278
18.6.1. In the UI	279
18.6.2. In the CLI	280
18.7. Enabling a Password Change Dialog	280
Chapter 19. Policy: Managing the Kerberos Domain	281

19.1. About Kerberos	281
19.1.1. About Principal Names	281
19.1.2. About Protecting Keytabs	282
19.2. Setting Kerberos Ticket Policies	282
19.2.1. Setting Global Ticket Policies	282
19.2.1.1. From the Web UI	282
19.2.1.2. From the Command Line	283
19.2.2. Setting User-Level Ticket Policies	284
19.3. Refreshing Kerberos Tickets	284
19.4. Caching Kerberos Passwords	285
19.5. Removing Keytabs	286
Chapter 20. Policy: Using sudo	287
20.1. About sudo and IPA	287
20.1.1. General sudo Configuration in Identity Management	287
20.1.2. sudo and Netgroups	287
20.1.3. Supported sudo Clients	288
20.2. Setting up sudo Commands and Command Groups	288
20.2.1. Adding sudo Commands	288
20.2.1.1. Adding sudo Commands with the Web UI	288
20.2.1.2. Adding sudo Commands with the Command Line	289
20.2.2. Adding sudo Command Groups	289
20.2.2.1. Adding sudo Command Groups with the Web UI	289
20.2.2.2. Adding sudo Command Groups with the Command Line	291
20.3. Defining sudo Rules	292
20.3.1. About External Users	292
20.3.2. About sudo Options Format	293
20.3.3. Defining sudo Rules in the Web UI	294
20.3.4. Defining sudo Rules in the Command Line	299
20.3.5. Suspending and Removing sudo Rules	303
20.4. Configuring Hosts to Use IdM sudo Policies	303
20.4.1. Applying the sudo Policies to Hosts Using LDAP	303
20.4.2. Applying the Configured sudo Policies to Hosts Using SSSD	305
Chapter 21. Policy: Configuring Host-Based Access Control	307
21.1. About Host-Based Access Control	307
21.2. Creating Host-Based Access Control Entries for Services and Service Groups	308
21.2.1. Adding HBAC Services	308
21.2.1.1. Adding HBAC Services in the Web UI	308
21.2.1.2. Adding Services in the Command Line	309
21.2.2. Adding Service Groups	309
21.2.2.1. Adding Service Groups in the Web UI	310
21.2.2.2. Adding Service Groups in the Command Line	311
21.3. Defining Host-Based Access Control Rules	312
21.3.1. Setting Host-Based Access Control Rules in the Web UI	312
21.3.2. Setting Host-Based Access Control Rules in the Command Line	315
21.4. Testing Host-Based Access Control Rules	319
21.4.1. The Limits of Host-Based Access Control Configuration	319
21.4.2. Test Scenarios for Host-Based Access Control (CLI-Based)	320
21.4.3. Testing Host-Based Access Control Rules in the UI	321
Chapter 22. Policy: Defining SELinux User Maps	324
22.1. About Identity Management, SELinux, and Mapping Users	324
22.2. Configuring SELinux Users in IdM	326
22.2.1. In the Web UI	326
22.2.2. In the CLI	327

22.3. Mapping SELinux Users and IdM Users	328
22.3.1. In the Web UI	329
22.3.2. In the CLI	331
Chapter 23. Policy: Defining Automatic Group Membership for Users and Hosts	334
23.1. About Automembership	334
23.2. Defining Automembership Rules (Basic Procedure)	335
23.2.1. From the Web UI	335
23.2.2. From the CLI	337
23.3. Examples of Using Automember Groups	338
23.3.1. Setting an All Users/Hosts Rule	338
23.3.2. Defining Default Automembership Groups	339
23.3.3. Using Automembership Groups with Windows Users	339
Chapter 24. Configuration: Defining Access Control for IdM Users	341
24.1. About Access Controls for IdM Entries	341
24.1.1. A Brief Look at Access Control Concepts	341
24.1.2. Access Control Methods in Identity Management	342
24.2. Defining Self-Service Settings	342
24.2.1. Creating Self-Service Rules from the Web UI	342
24.2.2. Creating Self-Service Rules from the Command Line	343
24.2.3. Editing Self-Service Rules	344
24.3. Delegating Permissions over Users	345
24.3.1. Delegating Access to User Groups in the Web UI	345
24.3.2. Delegating Access to User Groups in the Command Line	346
24.4. Defining Role-Based Access Controls	347
24.4.1. Creating Roles	351
24.4.1.1. Creating Roles in the Web UI	351
24.4.1.2. Creating Roles in the Command Line	352
24.4.2. Creating New Permissions	353
24.4.2.1. Creating New Permissions from the Web UI	353
24.4.2.2. Creating New Permissions from the Command Line	355
24.4.3. Creating New Privileges	356
24.4.3.1. Creating New Privileges from the Web UI	356
24.4.3.2. Creating New Privileges from the Command Line	358
Chapter 25. Configuration: Configuring the IdM Server	360
25.1. Identity Management Files and Logs	360
25.1.1. A Reference of IdM Server Configuration Files and Directories	360
25.1.2. About default.conf and Context Configuration Files	362
25.1.3. Checking IdM Server Logs	363
25.1.3.1. Enabling Server Debug Logging	365
25.1.3.2. Debugging Command-Line Operations	366
25.2. Disabling Anonymous Binds	369
25.3. Configuring Alternate Certificate Authorities	369
25.4. Configuring CRLs and OCSP Responders	370
25.4.1. Using an OCSP Responder with SELinux	371
25.4.2. Changing the CRL Update Interval	371
25.4.3. Changing the OCSP Responder Location	371
25.5. Changing Domain DNS Configuration	372
25.5.1. Setting DNS Entries for Multi-Homed Servers	372
25.5.2. Setting up Additional Name Servers	372
25.6. Changing Load Balancing for IdM Servers and Replicas	373
25.7. Managing Replication Agreements Between IdM Servers	373
25.7.1. Listing Replication Agreements	374
25.7.2. Creating and Removing Replication Agreements	375

25.7.3. Forcing Replication	375
25.7.4. Reinitializing IdM Servers	375
25.7.5. Resolving Replication Conflicts	376
25.7.5.1. Solving Naming Conflicts	376
25.7.5.2. Solving Orphan Entry Conflicts	377
25.8. Removing a Replica	377
25.9. Starting IdM with Expired Certificates	378
Chapter 26. Migrating from an LDAP Directory to IdM	380
26.1. An Overview of LDAP to IdM Migration	380
26.1.1. Planning the Client Configuration	380
26.1.1.1. Initial Client Configuration (Pre-Migration)	380
26.1.1.2. Recommended Configuration for Red Hat Enterprise Linux Clients	381
26.1.1.3. Alternative Supported Configuration	382
26.1.2. Planning Password Migration	383
26.1.2.1. Method 1: Using Temporary Passwords and Requiring a Change	383
26.1.2.2. Method 2: Using the Migration Web Page	383
26.1.2.3. Method 3: Using SSSD (Recommended)	384
26.1.2.4. Migrating Cleartext LDAP Passwords	384
26.1.2.5. Automatically Resetting Passwords That Do Not Meet Requirements	384
26.1.3. Migration Considerations and Requirements	384
26.1.3.1. LDAP Servers Supported for Migration	384
26.1.3.2. Migration Environment Requirements	385
26.1.3.3. Migration Tools	385
26.1.3.4. Migration Sequence	385
26.2. Examples for Using migrate-ds	386
26.2.1. Migrating Specific Subtrees	386
26.2.2. Specifically Including or Excluding Entries	387
26.2.3. Excluding Entry Attributes	388
26.2.4. Setting the Schema to Use	388
26.3. Scenario 1: Using SSSD as Part of Migration	388
26.4. Scenario 2: Migrating an LDAP Server Directly to Identity Management	390
Troubleshooting Identity Management	392
A.1. Installation Issues	392
A.1.1. Server Installation	392
A.1.1.1. GSS Failures When Running IPA Commands	392
A.1.1.2. named Daemon Fails to Start	392
A.1.2. Replica Installation	392
A.1.2.1. Certificate System setup failed.	392
A.1.2.2. There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.	393
A.1.2.3. The DNS forward record does not match the reverse address	393
A.1.3. Client Installations	394
A.1.3.1. The client can't resolve reverse hostnames when using an external DNS.	394
A.1.3.2. The client is not added to the DNS zone.	395
A.1.4. Uninstalling an IdM Client	395
A.2. UI Connection Problems	395
A.3. IdM Server Problems	396
A.3.1. There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.	396
A.4. Host Problems	397
A.4.1. Certificate Not Found/Serial Number Not Found Errors	397
A.4.2. Debugging Client Connection Problems	397
A.5. Kerberos Errors	398
A.5.1. Problems making connections with SSH when using GSS-API	398

A.5.2. There are problems connecting to an NFS server after changing a keytab	399
A.6. SELinux Login Problems	399
Working with certmonger	400
B.1. Requesting a Certificate with certmonger	400
B.2. Storing Certificates in NSS Databases	401
B.3. Tracking Certificates with certmonger	401
Glossary	402
A	402
B	403
C	404
D	406
E	408
F	408
G	408
H	409
I	409
K	410
L	410
M	411
N	412
O	413
P	414
R	415
S	417
T	420
U	421
V	421
X	421
Index	421
A	421
B	422
C	422
D	422
G	422
H	423
I	423
K	423
L	423
N	423
P	423
R	424
S	424
T	425
U	425
W	425
Z	425

Preface

Identity Management is a Red Hat Enterprise Linux-based way to create a security, identity, and authentication domain. The different security and authentication protocols available to Linux and Unix systems (like Kerberos, NIS, DNS, PAM, and sudo) are complex, unrelated, and difficult to manage coherently, especially when combined with different identity stores.

Identity Management provides a layer that unifies all of these disparate services and simplifies the administrative tasks for managing users, systems, and security. IdM breaks management down into two categories: *identity* and *policy*. It centralizes the functions of managing the users and entities within your IT environment (identity) and then provides a framework to define authentication and authorization for a global security framework and user-friendly tools like single sign-on (policy).

1. Audience and Purpose

With Identity Management, a Red Hat Enterprise Linux system can easily become the center of an identity/authentication domain and even provide access to the domain for clients of other operating systems. IdM is an integrated system, that builds on existing and reliable technologies like LDAP and certificate protocols, with a robust yet straightforward set of tools (including a web-based UI). The key to identity/policy management with IdM is simplicity and flexibility:

- Centralized identity stores for authentication and single sign-on using both integrated LDAP services (with 389 Directory Server) and, optionally, NIS services
- Clear and manageable administrative control over system services like PAM, NTP, and sudo
- Simplified DNS domains and maintenance
- Scalable Kerberos realms and cross-realms which clients can easily join

This guide is written for systems administrators and IT staff who will manage IdM domains, user systems, and servers. This assumes a moderate knowledge of Linux-based systems administration and familiarity with important concepts like access control, LDAP, and Kerberos.

This guide covers every aspect of using IdM, including preparation and installation processes, administrative tasks, and the IdM tools. This guide also explains the major concepts behind both identity and policy management, generally, and IdM features specifically. Administrative tasks in this guide are categorized as either *Identity* or *Policy* in the chapter title to help characterize the administrative functions.

2. Examples and Formatting

Each of the examples used in this guide, such as file locations and commands, have certain defined conventions.

2.1. Brackets

Square brackets (`[ ]`) are used to indicate an alternative element in a name. For example, if a tool is available in `/usr/lib` on 32-bit systems and in `/usr/lib64` on 64-bit systems, then the tool location may be represented as `/usr/lib[64]`.

2.2. Client Tool Information

The tools for IdM are located in the `/usr/bin` and the `/usr/sbin` directories.

The LDAP tools used to edit the IdM directory services, such as `ldapmodify` and `ldapsearch`, are from OpenLDAP. OpenLDAP tools use SASL connections by default. To perform a simple bind using a

username and password, use the **-x** argument to disable SASL.

2.3. Text Formatting and Styles

Certain words are represented in different fonts, styles, and weights. Different character formatting is used to indicate the function or purpose of the phrase being highlighted.

Formatting Style	Purpose
Monospace with a background	This type of formatting is used for anything entered or returned in a command prompt.
<i>Italicized text</i>	Any text which is italicized is a variable, such as <i>instance_name</i> or <i>hostname</i> . Occasionally, this is also used to emphasize a new term or other phrase.
Bolded text	Most phrases which are in bold are application names, such as Cygwin , or are fields or options in a user interface, such as a User Name Here : field or Save button. This can also indicate a file, package, or directory name, such as /usr/sbin .

Other formatting styles draw attention to important text.



NOTE

A note provides additional information that can help illustrate the behavior of the system or provide more detail for a specific issue.



IMPORTANT

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.



WARNING

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

3. Giving Feedback

If there is any error in this book or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for IdM through Bugzilla, <http://bugzilla.redhat.com/bugzilla>. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

1. Select the Red Hat group and the Red Hat Enterprise Linux 6 product.
2. Set the component to **doc-Enterprise_Identity_Management_Guide**.
3. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct

description of the problem, such as incorrect procedure or typo.

For enhancements, put in what information needs to be added and why.

4. Give a clear title for the bug. For example, "**Incorrect command example for setup script options**" is better than "**Bad example**".

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at docs@redhat.com.

4. Document Change History

Revision 6.5-1	November 20, 2013	Ella Deon Ballard
Bug fixing.		
Revision 6.4-3	August 20, 2013	Ella Deon Lackey
Bug fixing. Reorganizing some chapters.		
Revision 6.4-1	March 1, 2013	Ella Deon Lackey
Adding trusts.		
Revision 6.3-1	October 18, 2012	Ella Deon Lackey
Commenting out sudo configuration example. Removing group sync information from winsync chapter. Commenting out CRL generation section.		
Revision 6.2-8	December 16, 2011	Ella Deon Lackey
Fixing sudoers_debug example in sudo client configuration procedure, Bugzilla #768792. Fixing migration command example, Bugzilla #766089.		
Revision 6.2-7	December 6, 2011	Ella Deon Lackey
Release for GA of Red Hat Enterprise Linux 6.2.		

Chapter 1. Introduction to Identity Management

Red Hat Enterprise Linux IdM is a way to create identity stores, centralized authentication, domain control for Kerberos and DNS services, and authorization policies — all on Linux systems, using native Linux tools. While centralized identity/policy/authorization software is hardly new, Identity Management is one of the only options that supports Linux/Unix domains.

Identity Management provides a unifying skin for standards-defined, common network services, including PAM, LDAP, Kerberos, DNS, NTP, and certificate services, and it allows Red Hat Enterprise Linux systems to serve as the domain controllers.

Identity Management defines a domain, with servers and clients who share centrally-managed services, like Kerberos and DNS. This chapter first explains what Identity Management is. This chapter also covers how all of these services work together within the domain and how servers and clients work with each other.

1.1. IdM v. LDAP: A More Focused Type of Service

At the most basic level, Red Hat Identity Management is a domain controller for Linux and Unix machines. Identity Management defines the domain, using controlling servers and enrolled client machines. This provides centralized structure that has previously been unavailable to Linux/Unix environments, and it does it using native Linux applications and protocols.

1.1.1. A Working Definition for Identity Management

Security information frequently relates to *identities* of users, machines, and services. Once the identity is verified, then access to services and resources can be controlled.

For efficiency, risk management, and ease of administration, IT administrators try to manage identities as centrally as possible and to unite identity management with authentication and authorization policies. Historically, Linux environments have had a very difficult time establishing this centralized management. There are a number of different protocols (such as NIS and Kerberos) which define domains, while other applications store data (such as LDAP) and still others manage access (such as sudo). None of these applications talk to each other or use the same management tools. Every application had to be administered separately and it had to be managed locally. The only way to get a consistent identity policy was to copy configuration files around manually or to try to develop a proprietary application to manage identities and policies.

The goal of Identity Management is to simplify that administrative overhead. Users, machines, services, and policies are all configured in one place, using the same tools. Because IdM creates a domain, multiple machines can all use the same configuration and the same resources simply by joining the domain. Users only have to sign into services once, and administrators only have to manage a single user account.

IdM does three things:

- ▶ Create a Linux-based and Linux-controlled domain. Both IdM servers and IdM clients are Linux or Unix machines. While IdM can synchronize data with an Active Directory domain to allow integration with Windows servers, it is not an administrative tool for Windows machines and it does not support Windows clients. Identity Management is a management tool for Linux domains.
- ▶ Centralize identity management and identity policies.
- ▶ Build on existing, native Linux applications and protocols. While IdM has its own processes and configuration, its underlying technologies are familiar and trusted by Linux administrators and are well established on Linux systems.

In a sense, Identity Management isn't making administrators do something new; it is helping them do it better. There are a few ways to illustrate that.

On one extreme is the *low control* environment. Little Example Corp. has several Linux and Unix servers, but each one is administered separately. All passwords are kept on the local machine, so there is no central identity or authentication process. Tim the IT Guy just has to manage users on every machine, set authentication and authorization policies separately, and maintain local passwords. With IdM, things come to order. There is a simple way to have central user, password, and policy stores, so Tim the IT Guy only has to maintain the identities on one machine (the IdM server) and users and policies are uniformly applied to all machines. Using host-based access control, delegation, and other rules, he can even set different access levels for laptops and remote users.

In the middle is the *medium control* environment. Mid-Example Corp. has several Linux and Unix servers, but Bill the IT Guy has tried to maintain a greater degree of control by creating a NIS domain for machines, an LDAP directory for users, and Kerberos for authentication. While his environment is well under control, every application has to be maintained separately, using different tools. He also has to update all of the services manually whenever a new machine is added to his infrastructure or when one is taken offline. In this situation, IdM greatly reduces his administrative overhead because it integrates all of the different applications together seamlessly, using a single and simplified tool set. It also makes it possible for him to implement single sign-on services for all of the machines in his domain.

On the other extreme is the *absent control* environment. At Big Example Corp., most of the systems are Windows based and are managed in a tightly-knit Active Directory forest. However, development, production, and other teams have many Linux and Unix systems — which are basically excluded from the Windows controlled environment. IdM brings native control to the Linux/Unix servers, using their native tools and applications — something that is not possible in an Active Directory forest. Additionally, because IdM is Windows-aware, data can be synchronized between Active Directory and IdM, preserving a centralized user store.

IdM provides a very simple solution to a very common, very specific problem: identity management.

1.1.2. Contrasting Identity Management with a Standard LDAP Directory

The closest relative to Identity Management is a standard LDAP directory like 389 Directory Server, but there are some intrinsic differences between what they do and what they're *intended* to do.

First, it helps to understand what a directory service is. A directory service is a collection of software, hardware, and processes that stores information. While directory services can be highly specific (for example, DNS is a directory service because it stores information on hostnames), a generic directory service can store and retrieve any kind of information. LDAP directories like 389 Directory Server are generic directories. They have a flexible schema that supports entries for users, machines, network entities, physical equipment, and buildings, and that schema can be customized to define entries of almost anything. Because of its extensibility, LDAP servers like 389 Directory Server are frequently used as backends that store data for other applications. 389 Directory Server not only contains information, it organizes information. LDAP directories use a hierarchical structure, a *directory tree*, that organize entries into root entries (suffixes), intermediate or container entries (subtrees or branches), and leaf entries (the actual data). Directory trees can be very complex, with a lot of branch points, or very simple (flat) with few branch points.

The primary feature of an LDAP directory is its generality. It can be made to fit into a variety of applications.

Identity Management, on the other hand, has a very specific purpose and fits a very specific application. It is not a general LDAP directory, it is not a backend, and it is not a general policy server. It is not generic.

Identity Management focuses on identities (user and machine) and policies that relate to those identities and their interactions. While it uses an LDAP backend to store its data, IdM has a highly-customized and specific set of schema that defines a particular set of identity-related entries and defines them in detail. It has a relatively flat and simple directory tree because it has only a handful of entry types and relationships that are relevant to its purpose. It has rules and limitations on how the IdM server can be deployed because it can only be deployed for a specific purpose: managing identities.

The restrictions on IdM also give it a great deal of administrative simplicity. It has a simple installation process, a unified set of commands, and a clearly defined role in the overall IT infrastructure. An IdM domain is easy to configure, easy to join, and easy to manage, and the functions that it serves — particularly identity/authentication tasks like enterprise-wide single sign-on — are also easier to do with IdM than with a more general-purpose directory service.

Table 1.1. Identity Management Compared to 389 Directory Server

	389 Directory Server	Identity Management
Use	General purpose	Single domain, focused on identity management
Flexibility	Highly-customizable	Limitations to focus on identity and authentication
Schema	Default LDAP schema	Optimized, special schema for identity management
Directory Tree	Standard and flexible hierarchy	Flat tree with a fixed hierarchy
Authentication	LDAP	Kerberos or Kerberos and LDAP
Active Directory Synchronization	Bi-directional	Unidirectional, Active Directory to Identity Management
Password Policies	LDAP-based	Kerberos-based
User Tools	Java Console and standard LDAP utilities	Web-based UI and special Python command-line tools

LDAP directories like 389 Directory Server have flexibility and adaptability which makes them a perfect backend to any number of applications. Its primary purpose is to store and retrieve data efficiently.

IdM fills a very different niche. It is optimized to perform a single task very effectively. It stores user information and authentication and authorization policies, as well as other information related to access, like host information. Its single purpose is to manage identities.

1.2. Bringing Linux Services Together

Identity Management unifies disparate yet related Linux services into a single management environment. From there, it establishes a simple, easy way to bring host machines into the domain of those services.

An IdM server is, at its core, an identity and authentication server. The primary IdM server, essentially a domain controller, uses a Kerberos server and KDC for authentication. An LDAP backend contains all of the domain information, including users, client machines, and domain configuration.

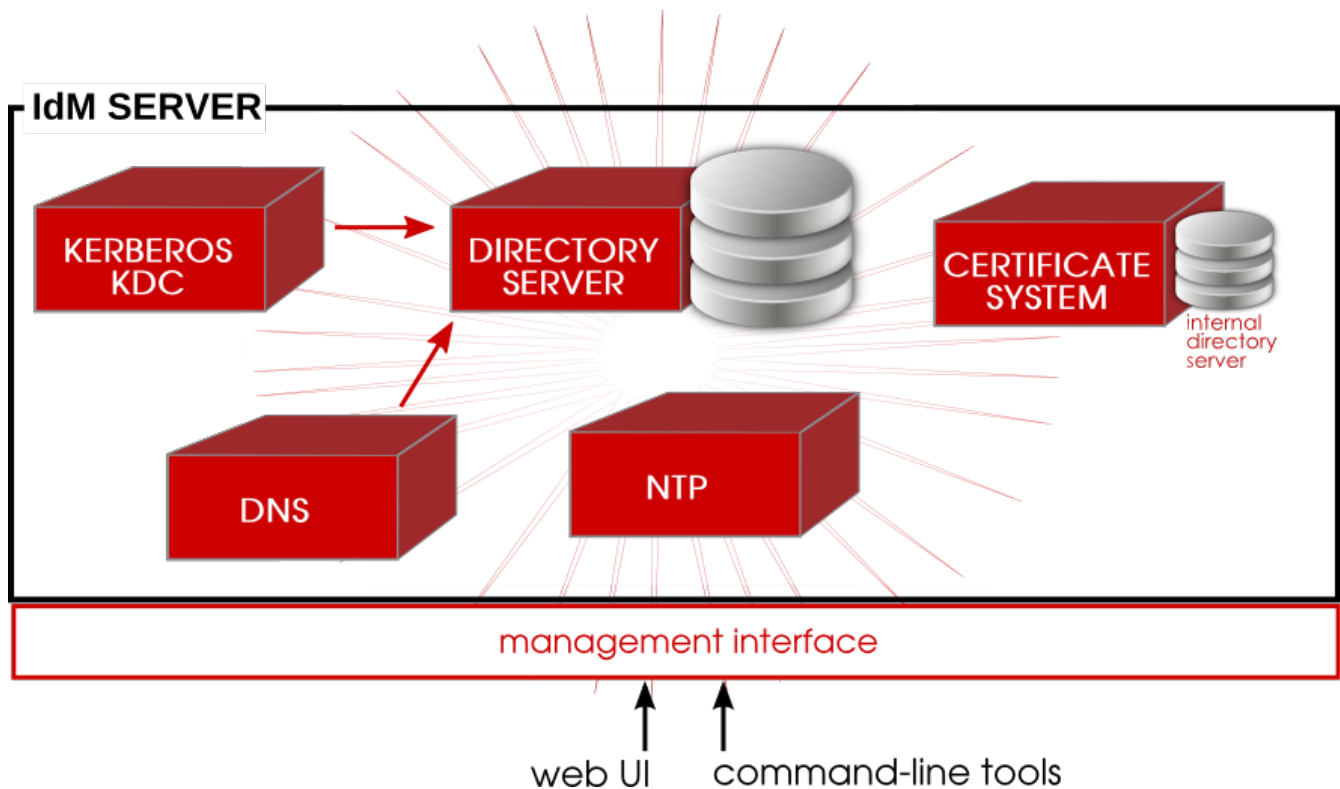


Figure 1.1. The IdM Server: Unifying Services

Other services are included to provide support for the core identity/authentication functions. DNS is used for machine discovery and for connecting to other clients in the domain. NTP is used to synchronize all domain clocks so that logging, certificates, and operations can occur as expected. A certificate service provides certificates for Kerberos-aware services. All of these additional services work together under the control of the IdM server.

The IdM server also has a set of tools which are used to manage all of the IdM-associated services. Rather than managing the LDAP server, KDC, or DNS settings individually, using different tools on local machines, IdM has a single management toolset (CLI and web UI) that allows centralized and cohesive administration of the domain.

1.2.1. Authentication: Kerberos KDC

Kerberos is an authentication protocol. Kerberos uses symmetric key cryptography to generate *tickets* to users. Kerberos-aware services check the ticket cache (a *keytab*) and authenticate users with valid tickets.

Kerberos authentication is significantly safer than normal password-based authentication because passwords are never sent over the network — even when services are accessed on other machines.

In Identity Management, the Kerberos administration server is set up on the IdM domain controller, and all of the Kerberos data are stored in IdM's backend Directory Server. The Directory Server instance defines and enforces access controls for the Kerberos data.

**NOTE**

The IdM Kerberos server is managed through IdM tools instead of Kerberos tools because all of its data are stored in the Directory Server instance. The KDC is unaware of the Directory Server, so managing the KDC with Kerberos tools does not affect the IdM configuration.

1.2.2. Data Storage: 389 Directory Server

Identity Management contains an internal 389 Directory Server instance. All of the Kerberos information, user accounts, groups, services, policy information, DNS zone and host entries, and all other information in IdM is stored in this 389 Directory Server instance.

When multiple servers are configured, they can talk to each other because 389 Directory Server supports *multi-master replication*. Agreements are automatically configured between the initial server and any additional *replicas* which are added to the domain.

1.2.3. Authentication: Dogtag Certificate System

Kerberos can use certificates along with keytabs for authentication, and some services require certificates for secure communication. Identity Management includes a certificate authority, through Dogtag Certificate System, with the server. This CA issues certificates to the server, replicas, and hosts and services within the IdM domain.

The CA can be a root CA or it can have its policies defined by another, external CA (so that it is *subordinate* to that CA). Whether the CA is a root or subordinate CA is determined when the IdM server is set up.

1.2.4. Server/Client Discovery: DNS

Identity Management defines a domain — multiple machines with different users and services, each accessing shared resources and using shared identity, authentication, and policy configuration. The clients need to be able to contact each other, as IdM servers. Additionally, services like Kerberos depend on hostnames to identify their principal identities.

Hostnames are associated with IP addresses using the *Domain Name Service* (DNS). DNS maps hostnames to IP addresses and IP addresses to hostnames, providing a resource that clients can use when they need to look up a host. From the time a client is enrolled in the IdM domain, it uses DNS service discovery to locate IdM servers within the domain and then all of the services and clients within the domain.

The client installation tool automatically configures the local System Security Services Daemon (SSSD) to use the IdM domain for service discovery. SSSD uses DNS already to look for LDAP/TCP and Kerberos / UDP services; the client installation only needs to supply the domain name. SSSD service discovery is covered in the [SSSD chapter in the Red Hat Enterprise Linux Deployment Guide](#).

On the server, the installation script configures the DNS file to set which services the DNS service discovery queries. By default, DNS discovery queries the LDAP service on TCP and different Kerberos services on both UDP and TCP. The DNS file which is created is described in [Section 16.2, “Using IdM and DNS Service Discovery with an Existing DNS Configuration”](#).

**NOTE**

While it is technically possible to configure the IdM domain to use DNS service discovery *without* having an IdM server host the DNS services, this is not recommended.

Multiple DNS servers are usually configured, each one working as an authoritative resource for machines within a specific domain. Having the IdM server also be a DNS server is optional, but it is strongly recommended. When the IdM server also manages DNS, there is tight integration between the DNS zones and the IdM clients and the DNS configuration can be managed using native IdM tools. Even if an IdM server is a DNS server, other external DNS servers can still be used.

1.2.5. Management: NTP

Many services require that servers and clients have the same system time, within a certain variance. For example, Kerberos tickets use time stamps to determine their validity. If the times between the server and client skew outside the allowed range, then any Kerberos tickets are invalidated.

Clocks are synchronized over a network using *Network Time Protocol* (NTP). A central server acts as an authoritative clock and all of the clients which reference that NTP server sync their times to match.

When the IdM server is the NTP server for the domain, all times and dates are synchronized before any other operations are performed. This allows all of the date-related services — including password expirations, ticket and certificate expirations, account lockout settings, and entry creation dates — to function as expected.

The IdM server, by default, works as the NTP server for the domain. Other NTP servers can also be used for the hosts.

1.3. Relationships Between Servers and Clients

Identity Management itself defines a *domain*, a group of machines that have shared configuration, policies, and identity stores. This shared configuration allows the machines (and users) within the domain to be aware of each other and operate together. This awareness can be used to enable cross-platform compatibility, like unifying Windows and Linux systems, or to enable infrastructure-wide single sign-on.

1.3.1. About IdM Servers and Replicas

Identity Management works by having identified servers which are the master stores of information for user and machine identities and domain-wide policies. These servers host domain-related services such as certificate authorities, NTP, Kerberos, SSH, and DNS. The server also acts as a central repository of identity and policy information.

Clients interact indirectly with IdM servers when they attempt to access domain resources, such as fileshares, services, remote machines, or authentication (through SSSD and Kerberos).

As said, an IdM server is a controller for a lot of associated services. While a number of those services are *supported*, most of them are not *required*. For example, a server may have a CA, a DNS server, or an NTP server — or it can be installed without those services.

Once an IdM server is set up, its configuration can be copied and used as the basis for another IdM server. When an IdM server is copied, that copy is called a *replica*.

NOTE

The only real difference between an IdM server and an IdM replica is that a server is a new installation — so that it defines the domain configuration — and a replica is based on an existing server and existing domain configuration.

Once an instance is configured, servers and replicas are basically identical in functionality and behavior within the IdM domain.

There is a good deal of flexibility in the IdM server (and replica) topology. For example, Server A can be installed with a CA and DNS services, while Replica A can be based on Server A's configuration but not host either DNS or CA services. Replica B can be added to the domain, also without CA or DNS services. At any time in the future, a CA or DNS service can be created and configured on Replica A or Replica B.

Servers and replicas both use underlying LDAP directories to store user and host entries, configuration data, policy configuration, and keytabs, certificates, and keys. Servers and replicas propagate data among each other through *multi-master replication agreements*. Both servers and replicas are masters (peers) in the replication topology.

Because the servers within the IdM domain are all LDAP peer servers, the replication topology must conform to the topology limits of a 389 Directory Server domain. This means that there can be no more than 20 peer servers in the IdM domain. Planning the server/replica topology is described more in [Section 4.1, “Planning the Server/Replica Topologies”](#).

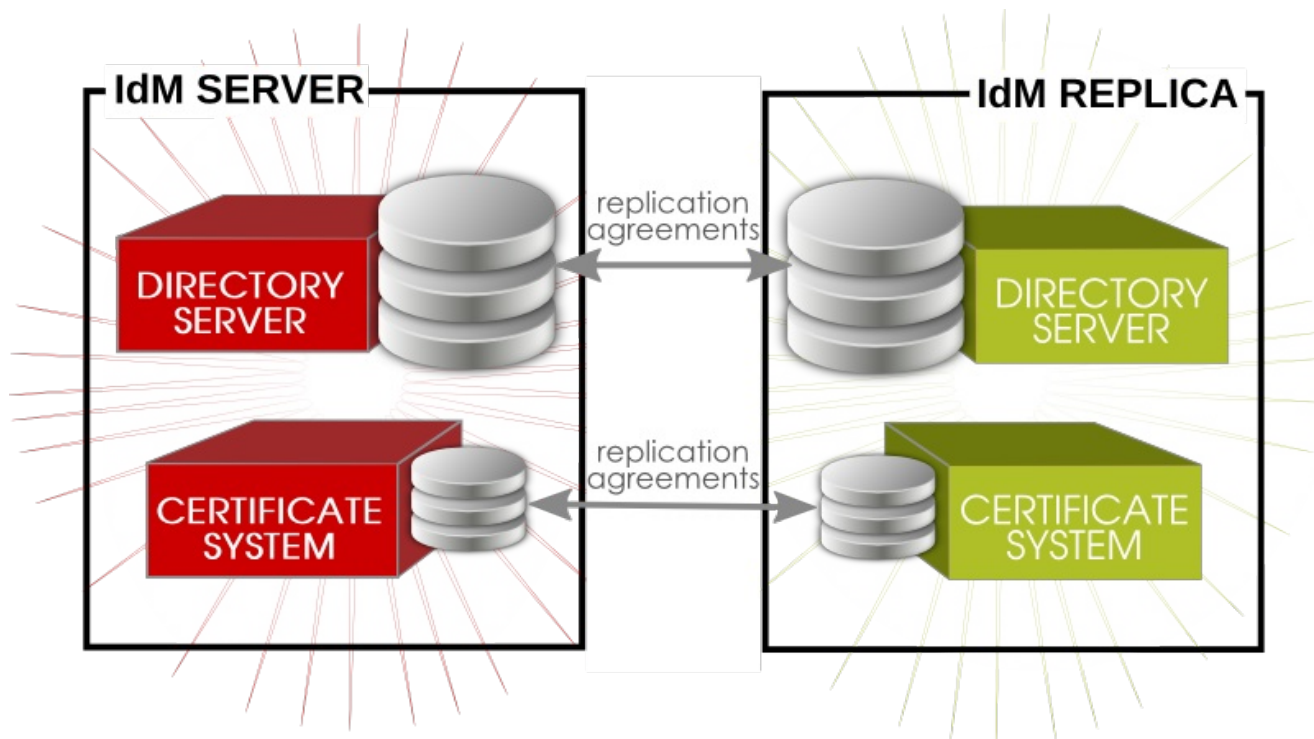


Figure 1.2. Server and Replica Interactions

TIP

The replication topology essentially creates a cloud of IdM servers. One benefit of a server domain is automatic load balancing, using the SRV records in DNS. The SRV record sets the priority order that servers and replicas are contacted, while weight distributes the load between servers/replicas with the same priority. The server and replica DNS entries can be edited to change the load balancing, which is covered in [Example 16.5, “SRV Record”](#) and [Section 25.6, “Changing Load Balancing for IdM Servers and Replicas”](#).

1.3.2. About IdM Clients

A client is simply any machine which is configured to operate within the IdM domain, using its Kerberos and DNS services, NTP settings, and certificate services. That's an important distinction: a client does not require a daemon or (necessarily) an installed product. It requires only system configurations which direct it to use IdM services.

For Red Hat Enterprise Linux systems, a certain number of platform tools are available for IdM to use, such as SSSD. These are *IdM-enabled platform applications*, which is simply a way of saying that they are aspects of the underlying platform that work with IdM services. Other tools, like certain PAM and NSS modules and IdM command-line utilities, are provided as IdM-specific packages that must be installed on the machine. These are *IdM-related components*.

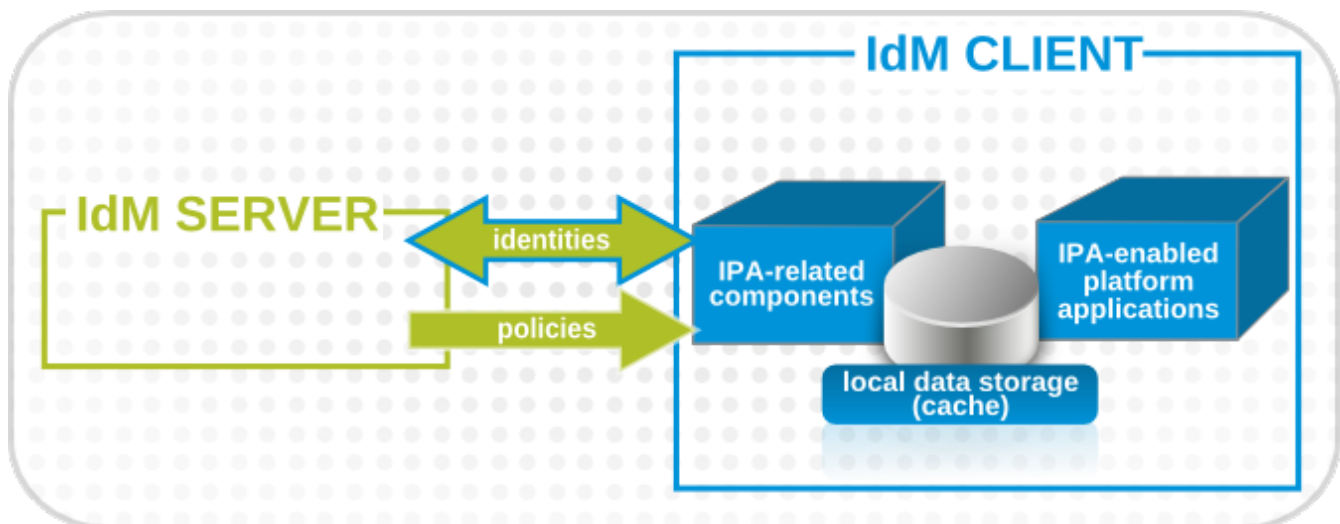


Figure 1.3. Server and Client Interactions

IdM uses the local storage (cache) on a client to improve performance in a few ways:

- Store IdM information when the machine is offline.
- Keep information active beyond its normal timeout period if the client cannot access the central server. The cache is persistent even after rebooting the machine.
- Reduce the round-trip time of requests by checking information locally before looking at the server.

Information is stored either in an LDB database (similar to LDAP) or the local filesystem (as XML files), depending on the *type* of information.

- Identity information (about users, machines, and groups) is stored in the LDB database, which uses the same syntax as an LDAP directory. This identity information is originally stored in the IdM

server's 389 Directory Server instance. Because this information changes frequently and is referenced frequently, it is important to be able to call the more current information quickly, which is possible using an LDB database on the client and the Directory Server on the server.

- Policy information is more static than identity information, and it can include configuration for SELinux or sudo. These policies are set globally on the server and then are propagated to the clients. On the client, the policy information is stored in the filesystem in XML files which can be downloaded and converted into a native file for whatever service is being managed.

A specific set of services on the IdM server interact with a subset of services and modules on the IdM client. A client is any machine (a *host*) which can retrieve a keytab or certificates from the IdM domain.

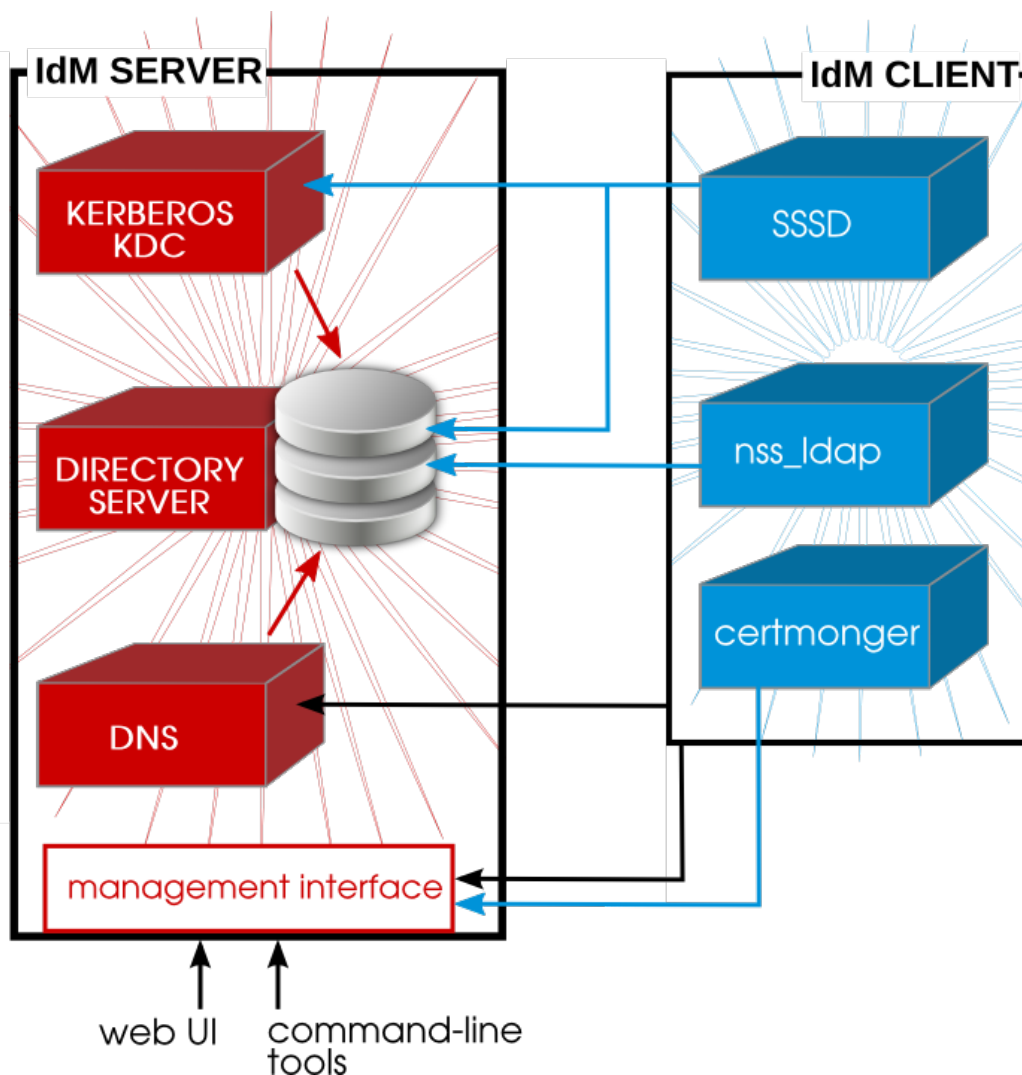


Figure 1.4. Interactions Between IdM Services

[Figure 1.4, “Interactions Between IdM Services”](#) shows that Red Hat Enterprise Linux uses two native daemons to interact with the IdM server:

- **SSSD** provides the user authentication for the machine and enforces host-based access control rules.
- **certmonger** monitors and renews the certificates on the client. It can request new certificates for the services on the system, including virtual machines.

When a Red Hat Enterprise Linux client is added to the domain (*enrolled*), its **SSSD** and **certmonger** are configured to connect to the IdM server and the required Kerberos keytab and host certificates are

created. (The host certificate is not used directly by IdM; it may be used by other services, such as a web server.)

Part I. Installing Identity Management Servers and Services

Chapter 2. Prerequisites for Installation

Before you install IdM, ensure that the installation environment is suitably configured. You also need to provide certain information during the installation and configuration procedures, including realm names and certain usernames and passwords. This section describes the information that you need to provide.

2.1. Supported Server Platforms

IdM 3.0 is supported on these platforms:

- Red Hat Enterprise Linux 6 i386
- Red Hat Enterprise Linux 6 x86_64

2.2. Hardware Recommendations

A basic user entry is about 1 KB in size, as is a simple host entry with a certificate. The most important hardware feature to size properly is RAM. While all deployments are different, depending on the number of users and groups and the type of data stored, there is a rule of thumb to use to help determine how much RAM to use:

- For 10,000 users and 100 groups, have at least 2GB of RAM and 1GB swap space.
- For 100,000 users and 50,000 groups, have at least 16GB of RAM and 4GB of swap space.



TIP

For larger deployments, it is more effective to increase the RAM than to increase disk space because much of the data are stored in cache.

The underlying Directory Server instance used by the IdM server can be tuned to increase performance. For tuning information, see the Directory Server documentation at http://docs.redhat.com/docs/en-US/Red_Hat_Directory_Server/8.2/html/Performance_Tuning_Guide/system-tuning.html.

2.3. Software Requirements

Most of the packages that an IdM server depends on are installed as dependencies when the IdM packages are installed. There are some packages, however, which are required before installing the IdM packages:

- Kerberos 1.10
- The *named* and *bind-dyndb-ldap* packages for DNS

2.4. Supported Web Browsers

The only supported browser to access the IdM web UI is Firefox 3.x or 4.x.

2.5. System Prerequisites

The IdM server is set up using a configuration script, and this script makes certain assumption about the host system. If the system does not meet these prerequisites, then server configuration may fail.

2.5.1. DNS Records

Proper forward and reverse DNS settings are critical for both IdM servers and replicas (copies of servers) to be configured. DNS is used for replicating data between servers, for identifying servers in SSL certificates, and in Kerberos tickets, among other places. Therefore, servers must be resolvable in both forward and reverse DNS configuration.

The DNS settings for a host can be determined easily used **ifconfig** and **dig**.

1. Obtain the hostname.

```
[root@server ~]# hostname
server.example.com
```

2. Get the IP address. In this example, the returned IP address is **196.2.3.4**.

```
[root@server !]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:01:4C:E1:2C
          inet addr:196.2.3.4  Bcast:196.9.8.7  Mask:255.255.255.255
          inet6 addr: 2620:52:0:102f:5054:1ff:fe4c:e12c/64  Scope:Global
          inet6 addr: fe80::5054:1ff:fe4c:e12c/64  Scope:Link
          ...
```

3. Verify that forward DNS is properly configured by using **dig** to query the hostname and check what IP address is returned. In this example, the expected IP address is **196.2.3.4**.

```
[root@server ~]# dig server.example.com
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6 <<>> server.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56680

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 7, ADDITIONAL: 12

;; QUESTION SECTION:
;server.example.com. IN A

;; ANSWER SECTION:
server.example.com. 2946 IN A 196.2.3.4
```

4. Verify the reverse DNS configuration using **dig** with the **-t ptr** to query the PTR records (reverse records) for the address. This is the IP address in reverse order, with **.in-addr.arpa**. appended to the address. This should resolve to the hostname, **server.example.com**. in this example.

```
[root@server ~]# dig -t ptr 4.3.2.196.in-addr.arpa.
; <<>> DiG 9.8.2rc1-RedHat-9.8.2-0.17.rc1.el6 <<>> -t ptr 241.40.16.10.in-addr.arpa
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57899

;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 7, ADDITIONAL: 10

;; QUESTION SECTION:
;4.3.2.196.in-addr.arpa. IN PTR

;; ANSWER SECTION:
4.3.2.196.in-addr.arpa. 21600 IN PTR server.example.com.
```

The DNS records should resolve to whatever hostname is used in the IdM certificates.



NOTE

If the IdM server is configured to host its own DNS server, the IdM DNS service processes all DNS queries. The IdM DNS records take precedence, and any previous existing DNS configuration is ignored.

All systems within the domain must be configured to use the IdM-managed DNS server.

2.5.2. Hostname and IP Address Requirements

Regardless of whether the DNS is within the IdM server or external, the server host must have DNS properly configured:

- The hostname must be a fully-qualified domain name. For example, **ipaserver.example.com**.



IMPORTANT

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.

- The hostname must be all lower-case.
- The server's A record must be set and resolve to its public IP address.

The fully-qualified domain name cannot resolve to the loopback address. It must resolve to the machine's public IP address, not to **127.0.0.1**. The output of the **hostname** command cannot be **localhost** or **localhost6**.

The A and PTR records do not need to match for the server.

- The server's hostname and IP address must be in its own **/etc/hosts** file. The fully-qualified domain name for the IdM server must be listed in the **hosts** file *before* any aliases.

**NOTE**

A misconfigured file can prevent the IdM command-line tools from functioning correctly and can prevent the IdM web interface from connecting to the IdM server.

Additionally, the hostname cannot be part of the localhost entry.

For example, this lists the IPv4 and IPv6 localhost entries for the host (properly), then the IdM server IP address and hostname as the first entry.

```
127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
192.168.1.1 ipaserver.example.com ipaserver
```

- It is recommended that a separate DNS domain be allocated for the IdM server to manage. While not required (clients from other domains can still be enrolled in the IdM domain), this is a convenience for overall DNS management.

2.5.3. Directory Server

There must not be any instances of 389 Directory Server installed on the host machine.

2.5.4. System Files

The server script overwrites system files to set up the IdM domain. The system should be clean, without custom configuration for services like DNS and Kerberos, before configuring the IdM server.

2.5.5. System Ports

IdM uses a number of ports to communicate with its services. These ports, listed in [Table 2.1, “IdM Ports”](#), must be open and available for IdM to work. They cannot be in use by another service or blocked by a firewall. To make sure that these ports are available, try **nc**, **telnet**, or **nmap** to connect to a port or run a port scan.

To open a port:

```
[root@server ~]# iptables -A INPUT -p tcp --dport 389 -j ACCEPT
```

The **iptables** man page has more information on opening and closing ports on a system.

Table 2.1. IdM Ports

Service	Ports	Type
HTTP/HTTPS	80	TCP
	443	
LDAP/LDAPS	389	TCP
	636	
Kerberos	88	TCP and UDP
	464	
DNS	53	TCP and UDP
NTP	123	UDP
Dogtag Certificate System - LDAP	7389	TCP

2.5.6. NTP

Network time protocol (NTP) synchronizes time between systems on a network. An NTP server centralizes and manages that clock synchronization. By default, Identity Management installs and configures an NTP server which is used by the domain to synchronize clocks for other Identity Management servers, replicas, and systems and services within the IdM domain.

Some sort of NTP server must be running for some domain tasks — such as Kerberos ticket maintenance and data replication between servers and replicas in the topology — to function properly. **It is not required that an IdM server host the NTP server, but it is strongly recommended. This is the default configuration.**

If a server is being installed on a virtual machine, that server *should not* run an NTP server. To disable NTP for IdM, use the `--no-ntp` option when the IdM server is configured to prevent an NTP server from being installed.

2.5.7. NSCD

It is strongly recommended that you avoid or restrict the use of **nscd** in an IdM deployment. The **nscd** service is extremely useful for reducing the load on the server, and for making clients more responsive, but there can be problems when a system is also using SSSD, which performs its own caching.

nscd caches authentication and identity information for all services that perform queries through **nsswitch**, including **getent**. Because **nscd** performs both positive and negative caching, if a request determines that a specific IdM user does not exist, it caches this as a negative response. Values stored in the cache remain until the cache expires, regardless of any changes that may occur on the server. The results of such caching is that new users and memberships may not be visible, and users and memberships that have been removed may still be visible.

To avoid clashes with SSSD caches and to prevent locking out users, avoid using **nscd** altogether. Alternatively, use a shorter cache time by resetting the time-to-live caching values in the `/etc/nscd.conf` file:

positive-time-to-live	group	3600
negative-time-to-live	group	60
positive-time-to-live	hosts	3600
negative-time-to-live	hosts	20

2.5.8. Networking

The default networking service used by Red Hat Enterprise Linux is NetworkManager. However, NetworkManager can cause problems with IdM and the KDC. **It is highly recommended that you use the network service to manage the networking requirements in an IdM environment and disable the NetworkManager service.**

1. Boot the machine into single-user mode.
2. Disable off the NetworkManager service in the start list and stop the NetworkManager service.

```
[root@server ~]# chkconfig NetworkManager off; service NetworkManager stop
```

3. If **NetworkManagerDispatcher** is installed, ensure that it is stopped and disabled:

```
[root@server ~]# chkconfig NetworkManagerDispatcher off; service  
NetworkManagerDispatcher stop
```

4. Then, make sure that the **network** service is properly started.

```
[root@server ~]# chkconfig network on; service network start
```

5. Ensure that static networking is correctly configured.
6. Restart the system.

Chapter 3. Installing an IdM Server

The IdM domain is defined and managed by an IdM *server* which is essentially a domain controller. There can be multiple domain controllers within a domain for load-balancing and failover tolerance. These additional servers are called *replicas* of the master IdM server.

Both IdM servers and replicas only run on Red Hat Enterprise Linux systems. For both servers and replicas, the necessary packages must be installed and then the IdM server or replica itself is configured through setup scripts, which configure all of the requisite services.

3.1. Installing the IdM Server Packages

Installing only the IdM server requires a single package, **ipa-server**. If the IdM server will also manage a DNS server, then it requires two additional packages to set up the DNS.

All of these packages can be installed using the **yum** command:

```
[root@server ~]# yum install ipa-server bind bind-dyndb-ldap
```

Installing the **ipa-server** also installs a large number of dependencies, such as *389-ds-base* for the LDAP service and *krb5-server* for the Kerberos service, along with IdM tools.

After the packages are installed, the server instance must be created using the **ipa-server-install** command. The options for configuring the new server instance are described in [Section 3.2, “About ipa-server-install”](#).

3.2. About ipa-server-install

An IdM server instance is created by running the **ipa-server-install** script. This script can accept user-defined settings for services, like DNS and Kerberos, that are used by the IdM instance, or it can supply predefined values for minimal input from the administrator.

The IdM setup script creates a server instance, which includes configuring all of the required services for the IdM domain:

- » The network time daemon (ntpd)
- » A 389 Directory Server instance
- » A Kerberos key distribution center (KDC)
- » Apache (httpd)
- » An updated SELinux targeted policy
- » The Active Directory WinSync plug-in
- » A certificate authority
- » *Optional.* A domain name service (DNS) server

The IdM setup process can be minimal, where the administrator only supplies some required information, or it can be very specific, with user-defined settings for many parts of the IdM services. The configuration is passed using arguments with the **ipa-server-install** script.

**NOTE**

The port numbers and directory locations used by IdM are all defined automatically, as defined in [Section 2.5.5, “System Ports”](#) and [Section 25.1, “Identity Management Files and Logs”](#). These ports and directories *cannot* be changed or customized.

While **ipa-server-install** can be run without any options, so that it prompts for the required information, it has numerous arguments which allow the configuration process to be easily scripted or to supply additional information which is not requested during an interactive installation.

[Table 3.1, “ipa-server-install Options”](#) lists some common arguments with **ipa-server-install**. The full list of options are in the **ipa-server-install** manpage. The **ipa-server-install** options are versatile enough to be customized to the specific deployment environment to install and configure different services as needed.

Table 3.1. ipa-server-install Options

Argument	Description
<code>-a ipa_admin_password</code>	The password for the IdM administrator. This is used for the admin user to authenticate to the Kerberos realm.
<code>--hostname=hostname</code>	The fully-qualified domain name of the IdM server machine.  IMPORTANT This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures. Additionally, the hostname must all be lower-case. No capital letters are allowed.
<code>-n domain_name</code>	The name of the LDAP server domain to use for the IdM domain. This is usually based on the IdM server's hostname.
<code>-p directory_manager_password</code>	The password for the superuser, cn=Directory Manager , for the LDAP service.
<code>-P kerberos_master_password</code>	The password for the KDC administrator. This is randomly generated if no value is given.
<code>-r realm_name</code>	The name of the Kerberos realm to create for the IdM domain.
<code>--subject=subject_DN</code>	Sets the base element for the subject DN of the issued certificates. This defaults to 0=realm .
<code>--forwarder=forwarder</code>	Gives a DNS forwarder to use with the DNS service. To specify more than one forwarder, use this option multiple times.
<code>--no-forwarders</code>	Uses root servers with the DNS service instead of forwarders.
<code>--no-reverse</code>	Does <i>not</i> create a reverse DNS zone when the DNS domain is set up. (If a reverse DNS zone is already configured, then that existing reverse DNS zone is used.) If this option is not used, then the default value is true, which assumes that reverse DNS should be configured by the installation script.
<code>--setup-dns</code>	Tells the installation script to set up a DNS service within the IdM domain. Using an integrated DNS service is optional, so if this option is not passed with the installation script, then no DNS is configured.
<code>--idmax=number</code>	Sets the upper bound for IDs which can be assigned by the IdM server. The default value is the ID start value plus 199999.

`--idstart=number`

Sets the lower bound (starting value) for IDs which can be assigned by the IdM server. The default value is randomly selected.

The way that an IdM server is installed can be different depending on the network environment, security requirements within the organization, and the desired topology. These examples illustrate some common options when installing the server. These examples are not mutually exclusive; it is entirely possible to use CA options, DNS options, and IdM configuration options in the same server invocation. These are called out separately simply to make it more clear what each configuration area requires.

3.3. Example: Running the Script Interactively and Silently

3.3.1. Basic Interactive Installation

All that is required to set up an IdM server is to run the **ipa-server-install** script. This launches the script interactively, which prompts for the required information to set up a server, but without more advanced configuration like DNS and CA options.

1. Run the **ipa-server-install** script.

```
[root@server ~]# ipa-server-install
```

2. Enter the hostname. This is determined automatically using reverse DNS.

```
Server host name [ipaserver.example.com]:
```

3. Enter the domain name. This is determined automatically based on the hostname.

```
Please confirm the domain name [example.com]:
```

4. The script then reprints the hostname, IP address, and domain name.

```
The IPA Master Server will be configured with
Hostname:      ipaserver.example.com
IP address:    192.168.1.1
Domain name:   example.com
```

5. Enter the new Kerberos realm name. This is usually based on the domain name.

```
Please provide a realm name [EXAMPLE.COM]:
```

6. Enter the password for the Directory Server superuser, **cn=Directory Manager**. There are password strength requirements for this password, including a minimum password length.

```
Directory Manager password:
Password (confirm):
```

7. Enter the password for the IdM system user account, **admin**. This user is created on the machine.

```
IPA admin password:
Password (confirm):
```

8. After that, the script configures all of the associated services for IdM, with task counts and progress bars.

```
Configuring ntpd
  [1/4]: stopping ntpd
  ...
done configuring ntpd.

Configuring directory server for the CA: Estimated time 30 seconds
  [1/3]: creating directory server user
  ...
done configuring pkids.

Configuring certificate server: Estimated time 6 minutes
  [1/17]: creating certificate server user
  ....
done configuring pki-cad.

Configuring directory server: Estimated time 1 minute
  [1/32]: creating directory server user
  ...
done configuring dirsrv.

Configuring Kerberos KDC: Estimated time 30 seconds
  [1/14]: setting KDC account password
  ...
done configuring krb5kdc.

Configuring kadmind
  [1/2]: starting kadmind
  [2/2]: configuring kadmind to start on boot
done configuring kadmind.

Configuring the web interface: Estimated time 1 minute
  [1/12]: disabling mod_ssl in httpd
  ...
done configuring httpd.
Setting the certificate subject base
restarting certificate server
Applying LDAP updates
Restarting the directory server
Restarting the KDC
Restarting the web server
Sample zone file for bind has been created in /tmp/sample.zone.ygzij5.db
=====
====
Setup complete
```

9. Restart the **SSH** service to retrieve the Kerberos principal and to refresh the name server switch (NSS) configuration file:

```
[root@server ~]# service sshd restart
```

10. Authenticate to the Kerberos realm using the admin user's credentials to ensure that the user is properly configured and the Kerberos realm is accessible.

```
[root@server ~]# kinit admin
Password for admin@EXAMPLE.COM:
```


11. Test the IdM configuration by running a command like **ipa user-find**. For example:

```
[root@server ~]# ipa user-find admin
-----
1 user matched
-----
User login: admin
Last name: Administrator
Home directory: /home/admin
Login shell: /bin/bash
Account disabled: False
Member of groups: admins
-----
Number of entries returned 1
-----
```

3.3.2. Silent (Non-Interactive) Installation

As shown in [Section 3.3.1, “Basic Interactive Installation”](#), only a few pieces of information are required to configure an IdM server. While the setup script can prompt for this information in interactive mode, this information can also be passed with the setup command to allow automated and unattended configuration:

- Passwords for the IdM administrative user and the Directory Server super user (Directory Manager)
- The server hostname
- The Kerberos realm name
- The DNS domain name

This information can be passed with the **ipa-server-install**, along with the **-U** to force it to run without requiring user interaction.

Example 3.1. Basic Installation without Interaction

```
[root@server ~]# ipa-server-install -a secret12 --
hostname=ipaserver.example.com -r EXAMPLE.COM -p secret12 -n example.com -U
```

The script then prints the submitted values:

```
To accept the default shown in brackets, press the Enter key.

The IPA Master Server will be configured with
Hostname:      ipaserver.example.com
IP address:    192.168.1.1
Domain name:   example.com
```

The server name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures. Additionally, the hostname must all be lower-case. No capital letters are allowed.

Then the script runs through the configuration progress for each IdM service, as in [Section 3.3.1, “Basic Interactive Installation”](#).

3.4. Examples: Installing with Different CA Configurations

Identity Management uses an integrated certificate authority (CA) to create the certificates and keytabs used by users and hosts within the domain. Even internal domain services, such as the LDAP server and the Apache server for the Identity Management web UI, require server certificates to establish secure connections with each other.

A Dogtag Certificate System CA is almost always installed with the IdM server. That CA uses a *CA signing certificate* to create and sign all of the server and user certificates created within the IdM domain. That CA certificate itself has to be signed by the CA which issued it, and there are two different ways that a CA can sign the Dogtag Certificate System CA signing certificate:

- The Dogtag Certificate System can sign *its own* certificate. This means that the Dogtag Certificate System instance is a *root* CA. There are no higher CAs, and the root CA can set its own certificate policies.

This is the default configuration.

- The Dogtag Certificate System CA can be signed by an externally-hosted CA (such as Verisign). In that case, the external CA is the root CA, and the configured Dogtag Certificate System CA is *subordinate* to that root. This means that the certificates issued within the IdM domain are potentially subject to restrictions set by the root CA for attributes like the validity period.

Referencing an external CA still uses a Dogtag Certificate System instance to issue all of the IdM domain certificates; the only difference is that the initial domain CA certificate is issued by a different CA.

There is one other option: installing without a CA at all. This requires that all certificates used within the IdM domain be created, uploaded, and renewed manually. There may be some environments where the additional maintenance burden is sustainable because of other restrictions within the infrastructure, but, in general, most deployments will use an integrated Dogtag Certificate System instance (and **certmonger**) to manage IdM domain certificates.



IMPORTANT

It is not possible to change the CA configuration after the domain is created and it is not possible to migrate from one configuration to another. It is crucial that the CA requirements be considered before beginning the installation process.

3.4.1. Installing with an Internal Root CA

The default configuration is to install a Dogtag Certificate System which signs its own root CA certificate. There are no additional parameters or configuration steps required when the **ipa-server-install** command is run.

```
[root@server ~]# ipa-server-install
... &< ...

The IPA Master Server will be configured with:
Hostname:      hp-dl180-04.lab.bos.redhat.com
IP address:    10.16.40.150
Domain name:   lab.bos.redhat.com
Realm name:    LAB.BOS.REDHAT.COM

Continue to configure the system with these values? [no]: yes

The following operations may take some minutes to complete.
Please wait until the prompt is returned.

... &< ...

Configuring directory server for the CA (pkids): Estimated time 30 seconds
[1/3]: creating directory server user
[2/3]: creating directory server instance
[3/3]: restarting directory server
Done configuring directory server for the CA (pkids).
Configuring certificate server (pki-cad): Estimated time 3 minutes 30 seconds
[1/21]: creating certificate server user
[2/21]: creating pki-ca instance
[3/21]: configuring certificate server instance
[4/21]: disabling nonces
[5/21]: creating CA agent PKCS#12 file in /root
[6/21]: creating RA agent certificate database
[7/21]: importing CA chain to RA certificate database
[8/21]: fixing RA database permissions
[9/21]: setting up signing cert profile
[10/21]: set up CRL publishing
[11/21]: set certificate subject base
[12/21]: enabling Subject Key Identifier
[13/21]: setting audit signing renewal to 2 years
[14/21]: configuring certificate server to start on boot
[15/21]: restarting certificate server
[16/21]: requesting RA certificate from CA
[17/21]: issuing RA agent certificate
[18/21]: adding RA agent as a trusted user
[19/21]: configure certificate renewals
[20/21]: configure Server-Cert certificate renewal
[21/21]: Configure HTTP to proxy connections
Done configuring certificate server (pki-cad).

... &< ...
```

3.4.2. Installing Using an External CA

The IdM server can use a certificate issued by an external CA. This can be a corporate CA or a third-party CA like Verisign or Thawte. As with a normal setup process, using an external CA still uses a Dogtag Certificate System instance for the IdM server for issuing all of its client and replica certificates; the initial CA certificate is simply issued by a different CA.

When using an external CA, there are two additional steps that must be performed: submit the generated certificate request to the external CA and then load the CA certificate and issued server certificate to complete the setup.

**IMPORTANT**

The CA signing certificate generated for the Identity Management server must be a valid CA certificate. **This requires either that the Basic Constraint be set to `CA=true` or that the Key Usage Extension be set on the signing certificate to allow it to sign certificates.**

**IMPORTANT**

It is not possible to change the CA configuration after the domain is created and it is not possible to migrate from one configuration to another. It is crucial that the CA requirements be considered before beginning the installation process.

Example 3.2. Using an External CA

1. Run the **ipa-server-install** script, using the **--external-ca** option.

```
[root@server ~]# ipa-server-install -a secret12 -r EXAMPLE.COM -P password
-p secret12 -n ipaserver.example.com --external-ca
```

2. The script sets up the NTP and Directory Server services as normal.
3. The script completes the CA setup and returns information about where the certificate signing request (CSR) is located, **/root/ipa.csr**. This request must be submitted to the external CA.

```
Configuring certificate server: Estimated time 6 minutes
[1/4]: creating certificate server user
[2/4]: creating pki-ca instance
[3/4]: restarting certificate server
[4/4]: configuring certificate server instance
The next step is to get /root/ipa.csr signed by your CA and re-run ipa-
server-install.
```

4. Submit the request to the CA. The process differs for every service.
It may be necessary to request the appropriate extensions for the certificate. The CA signing certificate generated for the Identity Management server must be a valid CA certificate. This requires either that the Basic Constraint be set to `CA=true` or that the Key Usage Extension be set on the signing certificate to allow it to sign certificates.
5. Retrieve the issued certificate and the CA certificate chain for the issuing CA. Again, the process differs for every certificate service, but there is usually a download link on a web page or in the notification email that allows administrators to download all the required certificates. Be sure to get the full certificate chain for the CA, not just the CA certificate.
6. Rerun **ipa-server-install**, specifying the locations and names of the certificate and CA chain files. For example:

```
[root@server ~]# ipa-server-install --
external_cert_file=/tmp/servercert20110601.p12 --
external_ca_file=/tmp/cacert.p12
```

7. Complete the setup process and verify that everything is working as expected, as in [Section 3.3.1, “Basic Interactive Installation”](#).

3.4.3. Installing without a CA

In very rare cases, it may not be possible to install certificate services with the Identity Management server. In that case, it is possible to install Identity Management without an integrated Certificate System instance, **as long as all required certificates are created and installed independently**.

For installation, three certificates are required:

- An LDAP server certificate
- An Apache server certificate
- An LDAP server certificate

These certificates must be requested from a third-party authority *before* beginning the installation process.

There are some important limitations with how certificates can be managed when there is no integrated Dogtag Certificate System instance:

- **certmonger** is not used to track certificates, so there is no expiration warning.
- There is no way to renew certificates through Identity Management.
- The certificate management tools (**ipa cert-***) cannot be used to view or manage certificates.
- All host certificates and any service certificates must be requested, generated, and uploaded manually. This also affects how host management tools like **ipa host-add** function.
- If a certificate is removed from an entry, it is not automatically revoked.



IMPORTANT

It is not possible to change the CA configuration after the domain is created and it is not possible to migrate from one configuration to another. It is crucial that the CA requirements be considered before beginning the installation process.

Example 3.3. Installing Identity Management Without a CA

There are five options required when installing without a CA, to pass the required certificates directly to the setup process:

- **LDAP server certificate**
 - `--dirsrv_pkcs12`, with the PKCS#12 certificate file for the LDAP server certificate
 - `--dirsrv_pin`, with the password to access the PKCS#12 file
- **Apache server certificate**
 - `--http_pkcs12`, with the PKCS#12 certificate file for the Apache server certificate
 - `--http_pin`, with the password to access the PKCS#12 file
- **Root CA certificate** (to allow the Apache and LDAP server certificates to be trusted across the domain)
 - `--root-ca`, with the PEM file of *only* the CA certificate, not the certificate chain

```
[root@server ~]# ipa-server-install --http_pkcs12 /tmp/http-server.p12 --
http_pin secret1 --dirsrv_pkcs12 /tmp/ldap-server.p12 --dirsrv_pin secret2 --
root-ca-file /tmp/root.pem ...
```

3.5. Example: Configuring DNS Services within the IdM Domain

IdM can be configured to manage its own DNS or use an existing DNS (which is the default). Running the setup script alone does not configure DNS; this requires the **--setup-dns** option.



WARNING

DNS records are vital for nearly all IdM domain functions, including running LDAP directory services, Kerberos, and Active Directory integration.

Be extremely cautious and ensure that you have a tested and functional DNS service available if the IdM domain will not use an IdM-hosted DNS server. It is critical that you have properly configured A and PTR records.

As with a basic setup, the DNS setup can either prompt for the required information or the DNS information can be passed with the script to allow an automatic or unattended setup process.

3.5.1. DNS Notes

- Wildcards cannot be used when configuring DNS names. Only explicit DNS domain names are supported.
- The **rndc** service is not configured by the **--setup-dns** option. This service must be configured manually after the IdM server is configured.

Example 3.4. Interactive DNS Setup

1. Run the **ipa-server-install** script, using the **--setup-dns** option.

```
[root@server ~]# ipa-server-install -a secret12 -r EXAMPLE.COM -P password
-p secret12 -n ipaserver.example.com --setup-dns
```

2. The script configures the hostname and domain name as normal.
3. The script then prompts for DNS forwarders. If forwarders will be used, enter yes, and then supply the list of DNS servers. If IdM will manage its own DNS service, then enter no.

```
Do you want to configure DNS forwarders? [yes]: no
No DNS forwarders configured
```

4. The script sets up the NTP, Directory Server, Certificate System, Kerberos, and Apache services.
5. Before completing the configuration, the script prompts to ask whether it should configure reverse DNS services. If you select yes, then it configures the **named** service.

```
Do you want to configure the reverse zone? [yes]: yes
Configuring named:
[1/9]: adding DNS container
[2/9]: setting up our zone
[3/9]: setting up reverse zone
[4/9]: setting up our own record
[5/9]: setting up kerberos principal
[6/9]: setting up named.conf
[7/9]: restarting named
[8/9]: configuring named to start on boot
[9/9]: changing resolv.conf to point to ourselves
done configuring named.
=====
=====
Setup complete
```

6. The **ipa-dns-install** command (which is run with the install script when the **--setup-dns** option is used) does not automatically configure the system's **rndc** service. This must be configured manually, after DNS is configured for IdM.
 - a. Create the **rndc** configuration file and key.

```
[root@server ~]# /usr/sbin/rndc-confgen -a
[root@server ~]# /sbin/restorecon /etc/rndc.key
```

This may require some user input to create entropy while the key is being created.

- b. Change the owner and permissions of the **rndc** key file.

```
[root@server ~]# chown root:named /etc/rndc.key
[root@server ~]# chmod 0640 /etc/rndc.key
```

7. Verify that everything is working as expected, as in [Section 3.3.1, "Basic Interactive Installation"](#).

If DNS is used with IdM, then two pieces of information are required: any DNS forwarders that will be used and using (or not) reverse DNS. To perform a non-interactive setup, this information can be

passed using the **--forwarder** or **--no-forwarders** option and **--no-reverse** option.

Example 3.5. Setting up DNS Non-Interactively

To set up a DNS server and domain for the IdM server, use the **--setup-dns** option. To configure additional forwarders, use the **--forwarder** option; for multiple forwarders, use multiple invocations of **--forwarder**.

```
[root@server ~]# ipa-server-install ... --setup-dns --forwarder=1.2.3.0 --  
forwarder=1.2.255.0
```

Some kind of forwarder information is required. If no external forwarders will be used with the IdM DNS service, then use the **--no-forwarders** option to indicate that only root servers will be used.

The script always assumes that reverse DNS is configured along with DNS, so it is not necessary to use any options to *enable* reverse DNS. To disable reverse DNS, use the **--no-reverse** option; if a reverse DNS zone is already configured, then using the **--no-reverse** option means that existing reverse DNS zone is used.

```
[root@server ~]# ipa-server-install ... --setup-dns --no-reverse
```

The **ipa-dns-install** command (which is run with the install script when the **--setup-dns** option is used) does not automatically configure the system's **rndc** service. This must be configured manually, after DNS is configured for IdM.

1. Create the **rndc** configuration file and key.

```
[root@server ~]# /usr/sbin/rndc-confgen -a  
[root@server ~]# /sbin/restorecon /etc/rndc.key
```

This may require some user input to create entropy while the key is being created.

2. Change the owner and permissions of the **rndc** key file.

```
[root@server ~]# chown root:named /etc/rndc.key  
[root@server ~]# chmod 0640 /etc/rndc.key
```


Chapter 4. Setting up IdM Replicas

Replicas are essentially clones of existing Identity Management servers, and they share identical core configuration. The replica installation process, then, has two major parts: copying the existing, required server configuration and then installing the replica based on that information.

4.1. Planning the Server/Replica Topologies

In the IdM domain, there are three types of machines:

- Servers, which manage all of the services used by domain members
- Replicas, which are essentially copies of servers (and, once copied, are identical to servers)
- Clients, which belong to the Kerberos domains, receive certificates and tickets issued by the servers, and use other centralized services for authentication and authorization

A replica is a clone of a specific IdM server. The server and replica share the same internal information about users, machines, certificates, and configured policies. These data are copied from the server to the replica in a process called *replication*. The two Directory Server instances used by an IdM server — the Directory Server instance used by the IdM server as a data store and the Directory Server instance used by the Dogtag Certificate System to store certificate information — are replicated over to corresponding consumer Directory Server instances used by the IdM replica. The different Directory Server instances recognize each other through *replication agreements*. An initial replication agreement is created between a master server and replica when the replica is created; additional agreements can be added to other servers or replicas using the `ipa-replica-manage` command.

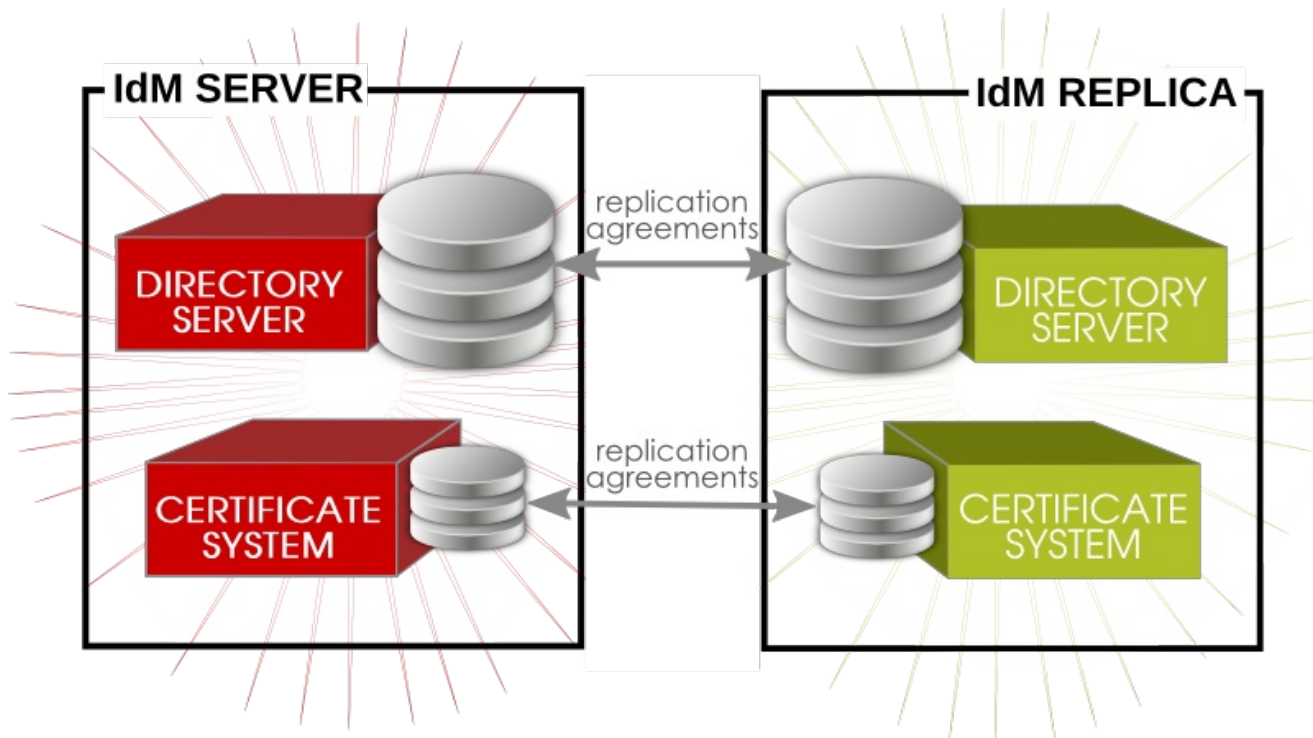


Figure 4.1. Server and Replica Agreements

Once they are installed, replicas are functionally identical to servers.

There are some guidelines with multi-master replication which place restrictions on the overall server/replica topology.

- No more than four replication agreements can be configured on a single server/replica.
- No more than 20 servers and replicas should be involved in a single Identity Management domain.
- Every server/replica should have a minimum of two replication agreements to ensure that there are no orphan servers or replicas cut out of the IdM domain if another server fails.

One of the most resilient topologies is to create a cell configuration for the servers/replicas, where there are a small number of servers in a cell which all have replication agreements with each other (a tight cell), and then each server has one replication agreement with another server *outside* the cell, loosely coupling that cell to every other cell in the overall domain.



Figure 4.2. Example Topology

There are some recommendations on how to accomplish this easily:

- Have at least one IdM server in each main office, data center, or locality. Preferably, have two IdM servers.
- Do not have more than four servers per data center.
- Rather than using a server or replica, small offices can use SSSD to cache credentials and use an off-site IdM server as its data backend.

4.2. Prerequisites for Installing a Replica Server

Replicas are functionally the same as IdM servers, so they have the same installation requirements and packages.

- Make sure that the machine meets all of the prerequisites listed in [Chapter 2, Prerequisites for Installation](#).
- **The replica and the master server must be running the same version of IdM.**

The replica essentially is a copy of a server, based off the existing server configuration. Therefore, the server and the replica (its copy) must be running the same version of Identity Management so that the configuration can be properly copied from the server to the replica.

If the master server is running on Red Hat Enterprise Linux 6.3, IdM version 2.2.x, then the replica must also run on Red Hat Enterprise Linux 6.3 and use the IdM 2.2.x packages.



IMPORTANT

Creating a replica of a different version than the master **is not supported**. Attempting to create a replica using a different version fails when attempting to configure the 389 Directory Server instance.

- If there is an existing Dogtag Certificate System or Red Hat Certificate System instance on the replica machine, make sure that port **7389** is free. This port is used by the master IdM server to communicate with the replica.
- Make sure the appropriate ports are open on both the server and the replica machine during and after the replica configuration. Servers and replicas connect to each other over ports 9443, 9444, 9445, and 7389 during the replica configuration. Once the replica is set up, the server and replica communicate over port 7389.
- The replica must use the same CA configuration as the server and must have the same root CA. For

example, if the server is its own root CA (using Dogtag Certificate System), then that must be the root CA for the replica. If the server used an external CA to issue its certificates, then the replica must use that same external CA.

4.3. Installing the Replica Packages

Since the replica is an IdM server (based on the configuration of an existing server), it is installed from the IdM server packages, **ipa-server**. If the replica will also host DNS services, then include the **bind** and **bind-dyndb-ldap** packages.

```
[root@server ~]# yum install ipa-server bind bind-dyndb-ldap
```



IMPORTANT

Do not run the **ipa-server-install** script.

4.4. Creating the Replica

1. On the master server, create a *replica information file*. This contains realm and configuration information taken from the master server which will be used to configure the replica server. Run the **ipa-replica-prepare** command *on the master IdM server*. The command requires the fully-qualified domain name of the *replica* machine. Using the **--ip-address** option automatically creates DNS entries for the replica, including the A and PTR records for the replica to the DNS.

```
[root@server ~]# ipa-replica-prepare ipareplica.example.com --ip-address 192.168.1.2
```

```
Determining current realm name
Getting domain name from LDAP
Preparing replica for ipareplica.example.com from ipaserver.example.com
Creating SSL certificate for the Directory Server
Creating SSL certificate for the Web Server
Copying additional files
Finalizing configuration
Packaging the replica into replica-info-ipareplica.example.com
```



IMPORTANT

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.
Additionally, the hostname must all be lower-case. No capital letters are allowed.

For more options with **ipa-replica-prepare**, see the **ipa-replica-prepare** manpage. Each replica information file is created in the **/var/lib/ipa/** directory as a GPG-encrypted file. Each file is named specifically for the replica server for which it is intended, such as **replica-info-ipareplica.example.com.gpg**.

**NOTE**

A replica information file cannot be used to create multiple replicas. It can only be used for the specific replica and machine for which it was created.

**WARNING**

Replica information files contain sensitive information. Take appropriate steps to ensure that they are properly protected.

2. Copy the replica information file to the replica server:

```
[root@server ~]# scp /var/lib/ipa/replica-info-ipareplica.example.com.gpg  
root@ipareplica:/var/lib/ipa/
```

3. On the replica server, run the replica installation script, referencing the replication information file. There are other options for setting up DNS, much like the server installation script. Additionally, there is an option to configure a CA for the replica; while CA's are installed by default for servers, they are optional for replicas.

Some information about DNS forwarders is required. A list can be given of configured DNS forwarders using a **--forwarder** option for each one, or forwarder configuration can be skipped by specifying the **--no-forwarders** option.

For example:

```
[root@ipareplica ~]# ipa-replica-install --setup-ca --setup-dns --no-forwarders /var/lib/ipa/replica-info-ipareplica.example.com.gpg

Directory Manager (existing master) password:

Warning: Hostname (ipareplica.example.com) not found in DNS
Run connection check to master
Check connection from replica to remote master 'ipareplica.example.com':
  Directory Service: Unsecure port (389): OK
  Directory Service: Secure port (636): OK
  Kerberos KDC: TCP (88): OK
  Kerberos Kpasswd: TCP (464): OK
  HTTP Server: Unsecure port (80): OK
  HTTP Server: Secure port (443): OK

The following list of ports use UDP protocol and would need to be
checked manually:
  Kerberos KDC: UDP (88): SKIPPED
  Kerberos Kpasswd: UDP (464): SKIPPED

Connection from replica to master is OK.
Start listening on required ports for remote master check
Get credentials to log in to remote master
admin@EXAMPLE.COM password:

Execute check on remote master
admin@example.com's password:
Check connection from master to remote replica 'ipareplica.example.com':
  Directory Service: Unsecure port (389): OK
  Directory Service: Secure port (636): OK
  Kerberos KDC: TCP (88): OK
  Kerberos KDC: UDP (88): OK
  Kerberos Kpasswd: TCP (464): OK
  Kerberos Kpasswd: UDP (464): OK
  HTTP Server: Unsecure port (80): OK
  HTTP Server: Secure port (443): OK

Connection from master to replica is OK.

Connection check OK
```

Additional options for the replica installation script are listed in the **ipa-replica-install** manpage.

The replica installation script runs a test to ensure that the replica file being installed matches the current hostname. If they do not match, the script returns a warning message and asks for confirmation. This could occur on a multi-homed machine, for example, where mismatched hostnames may not be an issue.

4. Enter the Directory Manager password when prompted. The script then configures a Directory Server instance based on information in the replica information file and initiates a replication process to copy over data from the master server to the replica, a process called *initialization*.
5. Verify that the proper DNS entries were created so that IdM clients can discover the new server. DNS entries are required for required domain services:
 - `_ldap._tcp`
 - `_kerberos._tcp`
 - `_kerberos._udp`
 - `_kerberos-master._tcp`

- `_kerberos-master._udp`
- `_ntp._udp`

If the initial IdM server was created with DNS enabled, then the replica is created with the proper DNS entries. For example:

```
[root@ipareplica ~]# DOMAIN=example.com
[root@ipareplica ~]# NAMESERVER=ipareplica
[root@ipareplica ~]# for i in _ldap._tcp _kerberos._tcp _kerberos._udp
 _kerberos-master._tcp _kerberos-master._udp _ntp._udp; do echo ""; dig
@${NAMESERVER} ${i}.${DOMAIN} srv +nocmd +noquestion +nocomments +nostats
+noaa +noadditional +noauthority; done | egrep -v "^;" | egrep _

_ldap._tcp.example.com. 86400      IN          SRV         0 100 389
ipaserver1.example.com.
_ldap._tcp.example.com. 86400      IN          SRV         0 100 389
ipaserver2.example.com.
_kerberos._tcp.example.com. 86400 IN          SRV         0 100 88
ipaserver1.example.com.
...8<...
```

If the initial IdM server was created without DNS enabled, then each DNS entry, including both TCP and UDP entries for some services, should be added manually. For example:

```
[root@ipareplica ~]# kinit admin
[root@ipareplica ~]# ipa dnsrecord-add example.com _ldap._tcp --srv-rec="0
100 389 ipareplica.example.com."
```

6. *Optional.* Set up DNS services for the replica. These are not configured by the setup script, even if the master server uses DNS.

Use the **ipa-dns-install** command to install the DNS manually, then use the **ipa dnsrecord-add** command to add the required DNS records. For example:

```
[root@ipareplica ~]# ipa-dns-install

[root@ipareplica ~]# ipa dnsrecord-add example.com @ --ns-rec
ipareplica.example.com.
```



IMPORTANT

Use the fully-qualified domain name of the replica, including the final period (.), otherwise BIND will treat the hostname as relative to the domain.

4.5. Alternate Options for Creating a Replica

Much of the core configuration of the replica is identical to that of the server from which it was created, such as the realm name and directory settings. However, while the settings need to match, it is not required that a replica manage the same services as the server. This is true for major services (DNS and CAs) and for minor services (NTP and OpenSSH).

The difference settings can be defined in the **ipa-replica-prepare** command or in the **ipa-replica-install** command.

4.5.1. Different DNS Settings

For DNS, the **ipa-replica-prepare** command can be used to configure DNS settings specific to the replica, meaning its IP address and reverse zone. For example:

```
[root@server ~]# ipa-replica-prepare ipareplica.example.com --ip-address=192.68.0.0 --no-reverse
```

If the server does not host any DNS services, then the replica can be set up to host DNS services for the Identity Management domain. As with installing a server, this is done with the **--setup-dns** option, and then settings for forward and reverse zones. For example, to configure DNS services for the replica with no forwarders and using an existing reverse zone:

```
[root@server ~]# ipa-replica-install ipareplica.example.com --setup-dns --no-forwarders --no-reverse --no-host-dns ...
```

The DNS options are described in the **ipa-replica-prepare** and **ipa-replica-install** manpages.

4.5.2. Different CA Settings

The CA configuration of the replica must echo the CA configuration of the server. If the server is configured with an integrated Dogtag Certificate System instance (regardless of whether it is a root CA or whether it is subordinate to an external CA), then the replica can either create its own integrated CA which is subordinate to the server CA or it can forgo having a CA at all, and forward all requests to the server's CA.

If the replica will have its own CA, then it uses the **--setup-ca** option. The rest of the configuration is taken from the server's configuration.

```
[root@ipareplica ~]# ipa-replica-install ipareplica.example.com --setup-ca ...
```

However, if the server was installed without any CA at all, then is nowhere to forward certificate operations — including the ability to request certificates for the new replica instance. All of the certificates for the replica, as with the server, must be requested and retrieved before installing the replica and then submitted with the installation command. The only exception is the root CA certificate; this is retrieved from the server as part of the replica setup.

```
[root@ipareplica ~]# ipa-replica-install ipareplica.example.com --dirsrv_pkcs12=/tmp/dirsrv-cert.p12 --dirsrv_pin=secret1 --http_pkcs12=/tmp/http-cert.p12 --http_pin=secret2 ...
```

4.5.3. Different Services

There are three support services that are installed on both servers and replicas by default: NTP, OpenSS client, and OpenSSH server. Any or all of this can be disabled on a replica. For example:

```
[root@server ~]# ipa-replica-install ... --no-ntp --no-ssh --no-sshd ...
```

Chapter 5. Setting up Systems as IdM Clients

A *client* is any system which is a member of the Identity Management domain. While this is frequently a Red Hat Enterprise Linux system (and IdM has special tools to make configuring Red Hat Enterprise Linux clients very simple), machines with other operating systems can also be added to the IdM domain.

One important aspect of an IdM client is that *only* the system configuration determines whether the system is part of the domain. (The configuration includes things like belonging to the Kerberos domain, DNS domain, and having the proper authentication and certificate setup.)



NOTE

IdM does not require any sort of agent or daemon running on a client for the client to join the domain. However, for the best management options, security, and performance, clients should run the System Security Services Daemon (SSSD).

For more information on SSSD, see [the SSSD chapter in the Deployment Guide](#).

This chapter explains how to configure a system to join an IdM domain.



NOTE

Clients can only be configured after at least one IdM server has been installed.

5.1. What Happens in Client Setup

Whether the client configuration is performed automatically on Red Hat Enterprise Linux systems using the client setup script or manually on other systems, the general process of configuring a machine to serve as an IdM client is mostly the same, with slight variation depending on the platform:

- Retrieve the CA certificate for the IdM CA.
- Create a separate Kerberos configuration to test the provided credentials.

This enables a Kerberos connection to the IdM XML-RPC server, necessary to join the IdM client to the IdM domain. This Kerberos configuration is ultimately discarded.

Setting up the Kerberos configuration includes specifying the realm and domain details, and default ticket attributes. Forwardable tickets are configured by default, which facilitates connection to the administration interface from any operating system, and also provides for auditing of administration operations. For example, this is the Kerberos configuration for Red Hat Enterprise Linux systems:


```
[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
rdns = false
forwardable = yes
ticket_lifetime = 24h

[realms]
EXAMPLE.COM = {
    kdc = ipaserver.example.com:88
    admin_server = ipaserver.example.com:749
}

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

- Run the **ipa-join** command to perform the actual join.
- Obtain a service principal for the host service and installs it into **/etc/krb5.keytab**. For example, **host/ipa.example.com@EXAMPLE.COM**.
- Enable certmonger, retrieve an SSL server certificate, and install the certificate in **/etc/pki/nssdb**.
- Disable the nscd daemon.
- Configure SSSD or LDAP/KRB5, including NSS and PAM configuration files.
- Configure an OpenSSH server and client, as well as enabling the host to create DNS SSHFP records.
- Configure NTP.

5.2. System Ports

IdM uses a number of ports to communicate with its services. These ports, listed in [Table 5.1, “IdM Ports”](#), must be open and available for IdM to work. They cannot be in use by another service or blocked by a firewall. To make sure that these ports are available, try **iptables** to list the available ports or **nc**, **telnet**, or **nmap** to connect to a port or run a port scan.

To open a port:

```
# iptables -A INPUT -p tcp --dport 389 -j ACCEPT
```

The **iptables** man page has more information on opening and closing ports on a system.

Table 5.1. IdM Ports

Service	Ports	Type
HTTP/HTTPS	80	TCP
	443	
LDAP/LDAPS	389	TCP
	636	
Kerberos	88	TCP and UDP
	464	
DNS	53	TCP and UDP
NTP	123	UDP

5.3. Configuring a Red Hat Enterprise Linux System as an IdM Client

There are two elements to prepare before beginning the client setup process for the Red Hat Enterprise Linux client:

- There must be a way to connect the client machine to the Kerberos domain, either by having an available Kerberos identity (such as the admin user) or by manually adding the client machine to the KDC on the server with a one-time password before beginning the enrollment process for the client machine.
- If there is an Active Directory server on the same network that serves DNS records, the Active Directory DNS records could prevent the client from automatically detecting the IdM server address. The **ipa-client-install** script retrieves the Active Directory DNS records instead of any records that were added for IdM.

In this case, it is necessary to pass the IdM server address directly to the **ipa-client-install** script.

To configure the client:

1. Install the client packages. These packages provide a simple way to configure the system as a client; they also install and configure SSSD.

For a regular user system, this requires only the **ipa-client** package:

```
# yum install ipa-client
```

An administrator machine requires the **ipa-admintools** package, as well:

```
# yum install ipa-client ipa-admintools
```

2. If the IdM server is configured as the DNS server and is in the same domain as the client, add the server's IP address as the first entry in the client's **/etc/resolv.conf** file.

**TIP**

If every machine in the domain will be an IdM client, then add the IdM server address to the DHCP configuration.

3. Run the client setup command.

```
# ipa-client-install --enable-dns-updates
```

The **--enable-dns-updates** option updates DNS with the client machine's IP address. This option should only be used if the IdM server was installed with integrated DNS or if the DNS server on the network accepts DNS entry updates with the GSS-TSIG protocol.

When using the **--server** option to specify the IdM server to register with, the server name must be a fully-qualified domain name.

**IMPORTANT**

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.

Other options for **ipa-client-install** are listed in the **ipa-client-install** manpage.

**NOTE**

There is an **--on-master** option that is used as part of configuring an IdM server (which also is an IdM client, since it is within the domain). This option should *never* be used when configuring a regular IdM client, because it results in slightly different client configuration which may not work on a non-IdM server machine.

4. If prompted, enter the domain name for the IdM DNS domain.

```
DNS discovery failed to determine your DNS domain
Please provide the domain name of your IPA server (ex: example.com):
example.com
```

5. If prompted, enter the fully-qualified domain name of the IdM server. Alternatively, use the **--server** option with the client installation script to supply the fully-qualified domain name of the IdM server.

```
DNS discovery failed to find the IPA Server
Please provide your IPA server name (ex: ipa.example.com):
ipaserver.example.com
```

**IMPORTANT**

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.

6. The client script then prompts for a Kerberos identity to use to contact and then join the Kerberos realm. When these credentials are supplied, then the client is able to join the IdM Kerberos domain and then complete the configuration:

```
Continue to configure the system with these values? [no]: yes
User authorized to enroll computers: admin
Password for admin@EXAMPLE.COM:
Enrolled in IdM realm EXAMPLE.COM
Created /etc/ipa/default.conf
Configured /etc/sss/sss.conf
Configured /etc/krb5.conf for IdM realm EXAMPLE.COM
SSSD enabled
Kerberos 5 enabled
NTP enabled
Client configuration complete.
```

7. Test that the client can connect successfully to the IdM domain and can perform basic tasks. For example, check that the IdM tools can be used to get user and group information:

```
$ id
$ getent passwd userID
$ getent group ipausers
```

8. Set up NFS to work with Kerberos.

**TIP**

To help troubleshoot potential NFS setup errors, enable debug information in the **/etc/sysconfig/nfs** file.

```
RPCGSSDARGS="-vvv"
RPCSVCGSSDARGS="-vvv"
```

- a. On an IdM server, add an NFS service principal for the NFS client.

```
# ipa service-add nfs/ipaclient.example.com@EXAMPLE
```

**NOTE**

This must be run from a machine with the *ipa-admintools* package installed so that the **ipa** command is available.

- b. On the IdM server, obtain a keytab for the NFS service principal.

```
# ipa-getkeytab -s ipaserver.example.com -p
nfs/ipaclient.example.com@EXAMPLE -k /tmp/krb5.keytab
```



NOTE

Some versions of the Linux NFS implementation have limited encryption type support. If the NFS server is hosted on a version older than Red Hat Enterprise Linux 6, use the **-e des-cbc-crc** option to the **ipa-getkeytab** command for any `nfs/<FQDN>` service keytabs to set up, both on the server and on all clients. This instructs the KDC to generate only DES keys.

When using DES keys, all clients and servers that rely on this encryption type need to have the **allow_weak_crypto** option enabled in the **[libdefaults]** section of the **/etc/krb5.conf** file. Without these configuration changes, NFS clients and servers are unable to authenticate to each other, and attempts to mount NFS filesystems may fail. The client's **rpc.gssd** and the server's **rpc.svcgssd** daemons may log errors indicating that DES encryption types are not permitted.

- c. Copy the keytab from the IdM server to the IdM client. For example:

```
# scp /tmp/krb5.keytab root@client.example.com:/etc/krb5.keytab
```

- d. Configure the **/etc/exports** file on the NFS server.

```
/ipashare      gss/krb5p(rw,no_root_squash,subtree_check,fsid=0)
```

- e. On the client, mount the NFS share. Use the same **-o sec** setting as is used in the **/etc/exports** file for the NFS server.

```
[root@client ~]# mount -v -t nfs4 -o sec=krb5p nfs.example.com:/
/mnt/ipashare
```

5.4. Manually Configuring a Linux Client

The **ipa-client-install** command automatically configures services like Kerberos, SSSD, PAM, and NSS. However, if the **ipa-client-install** command cannot be used on a system for some reason, then the IdM client entries and the services can be configured manually.

1. Install SSSD 1.5.x or later, if it is not already installed.
2. *Optional.* Install the IdM tools so that administrative tasks can be performed from the host.

```
# yum install ipa-admintools
```

3. *On an IdM server.* Create a host entry for the client.

```
$ ipa host-add --force --ip-address=192.168.166.31 ipaclient.example.com
```

Creating hosts manually is covered in [Section 10.2, “Adding Host Entries”](#).

4. *On an IdM server.* Create keytabs for the client.
 - a. Log in as IdM administrator.

```
$ kinit admin
```

- b. Set the client host to be managed by the server.

```
$ ipa host-add-managedby --hosts=ipaserver.example.com
ipaclient.example.com
```

- c. Generate the keytab for the client.

```
$ ipa-getkeytab -s ipaserver.example.com -p host/ipaclient.example.com
-k /tmp/ipaclient.keytab
```

5. Copy the keytab to the client machine and rename it **/etc/krb5.keytab**.



TIP

If there is an existing **/etc/krb5.keytab** that should be preserved, the two files can be combined using **ktutil**.

6. Set the correct user permissions and, if necessary, SELinux contexts for the **/etc/krb5.keytab** file.

```
chown root:root 0600
system_u:object_r:krb5_keytab_t:s0
```

7. Configure SSSD by editing the **/etc/sss/sss.conf** file to point to the IdM domain.

```
[sss]
config_file_version = 2
services = nss, pam

domains = example.com
[nss]

[pam]

[domain/example.com]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipaclient.example.com
chpass_provider = ipa
ipa_server = ipaserver.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
```

8. Configure NSS to use SSSD for passwords, groups, users, and netgroups.

```
vim /etc/nsswitch.conf

...
passwd:      files sss
shadow:      files sss
group:       files sss
...
netgroup:    files sss
...
```

9. Configure the **/etc/krb5.conf** file to point to the IdM KDC.

```
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
rdns = false
ticket_lifetime = 24h
forwardable = yes
allow_weak_crypto = true

[realms]
EXAMPLE.COM = {
    kdc = ipaserver.example.com:88
    admin_server = ipaserver.example.com:749
    default_domain = example.com
}

[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
```

10. Update the **/etc/pam.d** configuration to use the **pam_sss.so** modules.

- For **/etc/pam.d/fingerprint-auth**:

```
...
account      [default=bad success=ok user_unknown=ignore] pam_sss.so
...
session      optional      pam_sss.so
```

- For **/etc/pam.d/system-auth**:

```
...
auth          sufficient    pam_sss.so use_first_pass
...
account      [default=bad success=ok user_unknown=ignore] pam_sss.so
...
password     sufficient    pam_sss.so use_authtok
...
session      optional      pam_sss.so
```

- For **/etc/pam.d/password-auth**:

```
...
auth      sufficient    pam_sss.so use_first_pass
...
account   [default=bad success=ok user_unknown=ignore] pam_sss.so
...
password  sufficient     pam_sss.so use_authtok
...
session   optional       pam_sss.so
```

- Enrollment_with_Separation_of_DutiesFor `/etc/pam.d/smartcard-auth`:

```
...
account   [default=bad success=ok user_unknown=ignore] pam_sss.so
...
session   optional       pam_sss.so
```

11. Install the IdM server's CA certificate.

- a. Obtain the certificate from the server.

```
[root@ipaclient ~]# wget -O /etc/ipa/ca.crt
http://ipa.example.com/ipa/config/ca.crt
```

- b. Install the certificate in the system's NSS database.

```
[root@ipaclient ~]# certutil -A -d /etc/pki/nssdb -n "IPA CA" -t CT,C,C
-a -i /etc/ipa/ca.crt
```

12. Set up a host certificate for the host in IdM.

- a. Make sure **certmonger** is running.

```
[root@ipaclient ~]# service certmonger start
```



TIP

Configure **chkconfig** so that the **certmonger** service starts by default.

```
[root@ipaclient ~]# chkconfig certmonger on
```

- b. Use the **ipa-getcert** command, which creates and manages the certificate through **certmonger**. The options are described more in [Section B.1, “Requesting a Certificate with certmonger”](#).

```
$ ipa-getcert request -d /etc/pki/nssdb -n Server-Cert -K
HOST/ipaclient.example.com -N 'CN=ipaclient.example.com,O=EXAMPLE.COM'
```

If administrative tools were not installed on the client, then the certificate can be generated on an IdM server, copied over to the host, and installed using **certutil**.

13. Set up NFS to work with Kerberos.

**TIP**

To help troubleshoot potential NFS setup errors, enable debug information in the `/etc/sysconfig/nfs` file.

```
RPCGSSDARGS="-vvv"
RPCSVCGSSDARGS="-vvv"
```

- a. On an IdM server, add an NFS service principal for the NFS client.

```
# ipa service-add nfs/ipaclient.example.com@EXAMPLE
```

**NOTE**

This must be run from a machine with the `ipa-admintools` package installed so that the **ipa** command is available.

- b. On the IdM server, obtain a keytab for the NFS service principal.

```
# ipa-getkeytab -s ipaserver.example.com -p
nfs/ipaclient.example.com@EXAMPLE -k /tmp/krb5.keytab
```

**NOTE**

Some versions of the Linux NFS implementation have limited encryption type support. If the NFS server is hosted on a version older than Red Hat Enterprise Linux 6, use the **-e des-cbc-crc** option to the **ipa-getkeytab** command for any `nfs/<FQDN>` service keytabs to set up, both on the server and on all clients. This instructs the KDC to generate only DES keys.

When using DES keys, all clients and servers that rely on this encryption type need to have the **allow_weak_crypto** option enabled in the **[libdefaults]** section of the `/etc/krb5.conf` file. Without these configuration changes, NFS clients and servers are unable to authenticate to each other, and attempts to mount NFS filesystems may fail. The client's **rpc.gssd** and the server's **rpc.svcgssd** daemons may log errors indicating that DES encryption types are not permitted.

- c. Copy the keytab from the IdM server to the NFS server. For example, if the IdM and NFS servers are on different machines:

```
# scp /tmp/krb5.keytab root@nfs.example.com:/etc/krb5.keytab
```

- d. Copy the keytab from the IdM server to the IdM client. For example:

```
# scp /tmp/krb5.keytab root@client.example.com:/etc/krb5.keytab
```

- e. Configure the `/etc/exports` file on the NFS server.

```
/ipashare          gss/krb5p(rw,no_root_squash,subtree_check,fsid=0)
```

- f. On the client, mount the NFS share.
 - Always specify the share as *nfs_server:/mountpoint*.
 - Use the same **-o sec** setting as is used in the **/etc/exports** file for the NFS server.

```
[root@client ~]# mount -v -t nfs4 -o sec=krb5p nfs.example.com:/mnt/ipashare
```

5.5. Setting up a Linux Client Through Kickstart

A kickstart enrollment automatically adds a new system to the IdM domain at the time it is provisioned.

This requires pre-creating the hosts on the IdM server, with a predefined password that can be used to authenticate to complete the enrollment operation.

1. Create the host entry on the IdM server and set a temporary Kerberos password for the entry.

When the **ipa-client-install** script is run normally (interactively), it prompts for authentication credentials to access the IdM domain. However, when the script is run automatically, the system has to have some way to access the IdM domain without using an existing IdM user; this is done by setting the host principal in the script and using a Kerberos password (configured for the host account) to access the IdM domain.

For example:

```
[jsmith@ipaserver ~]$ ipa host-add kickstart-server.example.com --password=secret
```

The password expires after the first authentication attempt. After enrollment completes, the host is authenticated using its keytab.

2. Include the **ipa-client** package with the other install packages.

```
%packages
@ X Window System
@ Desktop
@ Sound and Video
ipa-client
...
```

3. Create a post-install instruction that runs the **ipa-client-install** script, passes all the required information to access and configure the IdM domain services, and specifies the pre-set password. Use the **--unattended** option to instruct the script to run non-interactively.

```
%post --log=/root/ks-post.log

# Get the hostname to set as the host principal
/bin/hostname > /tmp/hostname.txt

# Run the client install script
/usr/sbin/ipa-client-install --domain=EXAMPLEDOMAIN --enable-dns-updates --mkhomedir -w secret --realm=EXAMPLEREALM --server=ipaserver.example.com --unattended
```

4. Run the kickstart script.

5.6. Performing a Two-Administrator Enrollment

Enrolling machines as clients in the IdM domain is a two-part process. A host entry is created for the client (and stored in the 389 Directory Server instance), and then a keytab is created to provision the client.

Both parts are performed automatically by the **ipa-client-install** command. It is also possible to perform those steps separately; this allows for administrators to prepare machines and IdM in advance of actually configuring the clients. This allows more flexible setup scenarios, including bulk deployments.

When performing a manual enrollment, the host entry is created separately, and then enrollment is completed when the client script is run, which creates the requisite keytab.



NOTE

There are two ways to set the password. You can either supply your own or have IdM generate a random one.

There may be a situation where an administrator in one group is prohibited from *creating* a host entry and, therefore, from simply running the **ipa-client-install** command and allowing it to create the host. However, that administrator may have the right to run the command *after* a host entry exists. In that case, one administrator can create the host entry manually, then the second administrator can complete the enrollment by running the **ipa-client-install** command.

1. An administrator creates the host entry, as described in [Section 10.2, “Adding Host Entries”](#).
2. The second administrator installs the IdM client packages on the machine, as in [Section 5.3, “Configuring a Red Hat Enterprise Linux System as an IdM Client”](#).
3. When the second administrator runs the setup script, he must pass his Kerberos password and username (principal) with the **ipa-client-install** command. For example:

```
$ ipa-client-install -w secret -p admin2
```

4. The keytab is generated on the server and provisioned to the client machine, so that the client machine is not able to connect to the IdM domain. The keytab is saved with **root:root** ownership and 0600 permissions.

5.7. Manually Unconfiguring Client Machines

A machine may need to be removed from one IdM domain and moved to another domain or a virtual machine may be copied. There are a number of different situations where an IdM client needs to be reconfigured. The easiest solution is to uninstall the client and then configure it afresh.

```
ipa-client-install --uninstall
```

If it is not possible to uninstall the client directly, then the IdM configuration can be manually removed from the virtual machine.

**WARNING**

When a machine is unenrolled, the procedure cannot be undone. The machine can only be enrolled again.

1. Remove the old hostname from the main keytab. This can be done by removing every principal in the realm or by removing specific principals. For example, to remove all principals:

```
$ ipa-rmkeytab -k /etc/krb5.keytab -r EXAMPLE.COM
```

To remove specific principals:

```
$ ipa-rmkeytab -k /etc/krb5.keytab -p host/server.example.com@EXAMPLE.COM
```

2. Disable tracking in **certmonger** for every certificate. Each certificate must be removed from tracking individually.

```
$ ipa-getcert stop-tracking -n Server-Cert -d /etc/pki/nssdb
```

```
$ ipa-getcert stop-tracking -n Server2-Cert -d /etc/pki/nssdb
```

3. Remove the old host from the IdM DNS domain. While this is optional, it cleans up the old IdM entries associated with the system and allows it to be re-enrolled cleanly at a later time.

```
$ ipa host-del server.example.com
```

4. If the system should be re-added to a new IdM domain — such as a virtual machine which was moved from one location to another — then the system can be rejoined to IdM using the **ipa-join** command.

```
$ ipa-join
```

Chapter 6. Upgrading Identity Management to Red Hat Enterprise Linux 6.4

Identity Management is generally updated whenever a system is upgraded to a new release. Upgrades should be transparent and do not require any user or administrative intervention.

6.1. Upgrade Notes

- The update process automatically updates all schema and LDAP configuration, Apache configuration, and other services configuration, and restarts all IdM-associated services.
- When a replica is created, it must be the same version as the master it is based on. This means that **replicas should not be created on an older version of Identity Management while the servers are in the process of being upgraded**. Wait until the upgrade process is completed, and *then* create new replicas.
- Schema changes are replicated between servers. So once one master server is updated, all servers and replicas will have the updated schema, even if their packages are not yet updated. This ensures that any new entries which use the new schema can still be replicated among all the servers in the IdM domain.

The LDAP upgrade operation is logged in the upgrade log at `/var/log/ipaupgrade-log`. If any LDAP errors occur, then they are recorded in that log. Once any errors are resolved, the LDAP update process can be manually initiated by running the updater script:

```
[root@server ~]# ipa-ldap-updater --upgrade
```

- Clients do not need to have new packages installed. The client packages used to configure a Red Hat Enterprise Linux system do not impact the enrollment of the client within the domain.
- Updating client packages could bring in updated packages for other dependencies, such as **certmonger** which contain bug fixes, but this is not required to maintain client functionality or behavior within the IdM domain.

6.2. Upgrading Packages

The IdM server packages are updated when the system packages are updated:

```
[root@ipaserver ~]# yum update
```

This is the easiest way to upgrade the server because it automatically pulls in updates for related services, like SSSD, which provide Identity Management functionality.

To upgrade the IdM server packages specifically, run **yum** on the master server:

```
[root@ipaserver ~]# yum update ipa-server
```

It can take several seconds for the update process to apply all of the changes.

**NOTE**

It is not necessary to update all servers and replicas at precisely the same time; the IdM servers will still work with each other and replicate data successfully. The older IdM servers will simply lack the new features.

6.3. Removing Browser Configuration for Ticket Delegation (For Upgrading from 6.2)

As part of establishing Kerberos authentication, a principal is given a *ticket granting ticket* (TGT). Whenever that principal attempts to contact a service or application within the Kerberos domain, the service checks for an active TGT and then requests its own service-specific ticket from the TGT for that principal to access that service.

As part of configuring the web browser used to access the IdM web UI (and any other Kerberos-aware web applications), previous versions of Identity Management required that the TGT delegation be forwarded to the IdM server. This required adding the **delegation-uris** parameter to the **about:config** setup in Firefox:

```
network.negotiate-auth.delegation-uris .example.com
```

In Red Hat Enterprise Linux 6.3, Identity Management uses the Kerberos Services for User to Proxy (S4U2Proxy), so this additional delegation step is no longer required.

Updating Existing Configured Browsers

For browsers which have already been configured to use the Identity Management web UI, the **delegation-uris** setting can be cleared after upgrading to **ipa-server-2.2.0** or **ipa-client-2.2.0**.

There is no need to restart the browser after changing the **delegation-uris** setting.

Updating `configure.jar` for New Browser Configuration

The browser configuration is defined in the **configure.jar** file. This JAR file is generated when the server is installed and it is not updated with other files when IdM is updated. Any browsers configured will still have the **delegation-uris** parameter set unnecessarily, even after the IdM server is upgraded. However, the **configure.jar** file can be updated.

The **preferences.html** file in **configure.jar** sets the **delegation-uris** parameter. The updated **preferences.html** file can be added to **configure.jar**, and then **configure.jar** can be re-signed and re-deployed on the IdM servers.

**NOTE**

Only update the **configure.jar** file on the *initial* IdM server. This is the master server, and it is the only server which has a signing certificate. Then propagate the updated file to the other servers and replicas.

1. Update the packages on the initial IdM master server (the first instance). This will bring in the 3.0

UI packages, including the **configure.jar** file.

2. Back up the existing **configure.jar** file.

```
[root@ipaserver ~]# mv /usr/share/ipa/html/configure.jar  
/usr/share/ipa/html/configure.jar.old
```

3. Create a temporary working directory.

```
[root@ipaserver ~]# mkdir /tmp/sign
```

4. Copy the updated **preferences.html** file to the working directory.

```
[root@ipaserver ~]# cp /usr/share/ipa/html/preferences.html /tmp/sign
```

5. Use the **signtool** command (one of the NSS utilities) to add the new **preferences.html** file and re-sign the **configure.jar** file.

```
[root@ipaserver ~]# signtool -d /etc/httpd/alias -k Signing-Cert -Z  
/usr/share/ipa/html/configure.jar -e ".html" -p `cat  
/etc/httpd/alias/pwdfilere.txt` /tmp/sign
```

The **-e** option tells the tool to sign only files with a **.html** extension. The **-Z** option creates a new JAR file.

6. Copy the regenerated **configure.jar** file to all other IdM servers and replicas.

6.4. Testing Before Upgrading the IdM Server (Recommended)

It can be beneficial, and safer, to test newer versions of Identity Management before upgrading production systems. There is a relatively simple way to do this by creating a sacrificial replica and testing on that system.

1. Set up a replica based on one of the production servers, with the same version of IdM as is running in production, as described in [Chapter 4, Setting up IdM Replicas](#). For this example, this is called Test Replica. Make sure that Test Replica can successfully connect to the *production* server and domain.
2. After verifying that Test Replica has been successfully added to the production domain, **disconnect Test Replica from the network**.
3. Remove the replication agreements for Test Replica from the original IdM server and from Test Replica.
4. Reconnect Test Replica to the network.
5. Upgrade the packages on Test Replica using **yum** or whatever package update tool is appropriate for your system. For example:

```
[root@ipareplica ~]# yum update ipa*
```

6. Test common things on Test Replica, like getting Kerberos credentials, opening the server UI, and running commands.

Chapter 7. Uninstalling IdM Servers and Replicas

To uninstall both an IdM server and an IdM replica, pass the **--uninstall** option to the **ipa-server-install** command:

```
[root@ipareplica ~]# ipa-server-install --uninstall
```


Chapter 8. The Basics of Managing the IdM Server and Services

All of the access to Identity Management, both through the web UI and through the command line, is done by a user authenticating to the IdM domain. This chapter covers the basics of setting up browsers to handle Kerberos authentication, logging into Identity Management, and troubleshooting some common connection issues.

8.1. Starting and Stopping the IdM Domain

When an IdM server is installed, there are a number of different services which can be installed and configured with it in any combination, including (but not limited to) a Directory Server, a certificate authority, a web server, DNS, NTP, certmonger, and Kerberos.

All of these services work together in concert. Because there are dependencies between the services, the order in which services are started and stopped is critical.

When changes are made to a single service (such as the LDAP directory or the web server), then that individual service can be started and stopped using the **service** command. However, when multiple domain services need to be restarted (or the entire IdM server), then use the **ipactl** command, which always starts and stops services in the appropriate order.

Which services are configured for a specific IdM server are defined in the 389 Directory Server configuration, based on the hostname of the IdM server. ^[1] The 389 Directory Server service is always the first service started and the last service stopped. The rest of the run order depends on the configured services.

The **ipactl** command can start, stop, and restart services.

```
ipactl start | stop | restart
```

The **chkconfig** command sets what services to start automatically when the system restarts. The **ipactl** command can be used to start the domain services in the proper order, without having to configure each one individually in the **chkconfig** run order.

```
[root@server ~]# chkconfig ipactl on
```

8.2. About the IdM Client Tools

IdM creates a domain of recognized services, host machines, and users with universally-applied authentication sources and common policies. From the perspective of a client machine and an IdM user, the domain itself is fairly transparent after the initial configuration. All users need to do is log into the domain using Kerberos, and that's it.

However, an administrator has two ongoing tasks: add principals to the IdM Kerberos domain and set the domain policies and server configuration that govern domain interactions. Identity Management has both command-line and web-based interfaces for administrators to use to manage the domain, services, and IdM entries.

The most common method to maintain the domain is using the command-line tools. Identity Management has an incredibly broad set of scripts and commands that are available to administrators. The entry management functions of the domain are carried out with a single script: **ipa**. This script is a parent or control script for associated subcommands; each subcommand relates to a specific entry type.

The command-line scripts offer a number of benefits:

- ▶ The scripts allow management tasks to be automated and performed repeatedly in a consistent way without manual intervention.
- ▶ Entries can be added with all possible attributes configured (or a desired subset of attributes) in a single step. The web UI frequently requires two steps to fully configure an entry: the first to create the entry and the next to add optional attributes.
- ▶ The command-line scripts support adding additional attributes which may not be available in the UI or even custom attributes to entries, if the schema is configured.

8.2.1. The Structure of the `ipa` Command

The **ipa** command is essentially a big plug-in container. It supports dozens of subcommands; these subcommands are actually plug-ins which manage specific types of objects in Identity Management.

The first type of a subcommand identifies the object type (such as user, sudo, group, host, or dns), and the second part identifies the operation being performed on that object.

```
ipa objectType-operation objectName --option=value
```

For example, adding a user is done using the **user-add** subcommand:

```
ipa user-add entryName options
```

Related subcommands are grouped together into *plug-in modules*. Commands for managing DNS entries like **dnszone-add** and **dnsrecord-add** all belong to the *dns* module or *topic*. All of the information for managing a specific area, with all of the supported commands and examples for each, are available by viewing the help for that topic:

```
ipa help topic
```



TIP

To get a list of all available topics, run the **help** command without a topic name:

```
ipa help
```

All topic or command areas follow a consistent pattern for how entries are managed.

8.2.1.1. Adding, Editing, and Deleting Entries with `ipa`

New entries are added using an **-add* command. For example:

```
$ ipa user-add jsmith
```

For **add** operations, commands usually prompt for any required configuration attributes, which can be passed as command-line options or using **--set/addattr** options ([Section 8.2.3, “Managing Entry Attributes with --setattr, --addattr, and --delattr”](#)).

```
$ ipa user-add
First name: John
Last name: Smith
User login [jsmith]: jsmith
-----
Added user "jsmith"
-----
...
```

Likewise, entries are usually edited through a *** -mod** commands, and then any new or edited attributes are listed as options after it.

```
$ ipa user-mod jsmith --title="Editor III"
```

Last, entries can be deleted using the *** -del** command and the entry's name.

```
$ ipa user-del jsmith
```

8.2.1.2. Finding and Displaying Entries with ipa

Entries for an entire type are searched for using the *** -find** command and an optional search criterion. The criterion is a string which can either be an exact match or a substring of any of the search attribute values. For example, this searches both for the exact match on the string *smith* (such as an **sn** value of Smith) and a substring search for values such as a username of *jsmith* or a longer surname, such as *Smithson*.

```
ipa user-find smith
```

All searches are automatically substring searches; it is not necessary to specify a wildcard.

With no search criterion, every entry of that type is displayed.

Searches (any *** -find** command) have certain limits imposed as part of the server configuration, specifically how many entries are returned (size limits) and how long a search will run (time limits). This is covered in [Section 9.11.1.2, “Setting IdM Search Limits”](#). Part of the server configuration is setting global defaults for size and time limits on searches. While these limits are always enforced in the web UI, they can be overridden with any *** -find** command with the **--sizelimit** and **--timelimit** options. For example, if the default time limit is 60 seconds and a search is going to take longer, the time limit can be increased to 120 seconds:

```
[jsmith@ipaserver ~]$ ipa user-find smith --timelimit=120
```

Not every possible attribute in an entry type can be searched for. A certain subset of attributes are predefined and indexed for searches. (This list is configurable for users and groups, but not for other types of entries.)

When entries are returned, only certain default attributes are displayed with the entry; to return all attributes currently set for entries, use the **--all** option.

To display a specific entry, use the *** -show** command and the entry name. As with searches, only a subset of attributes are displayed with the entry unless the **--all** option is used.

8.2.1.3. Adding Members to Groups and Containers with ipa

Group members are added and removed with separate commands, apart from simply modifying an entry.

Member commands essentially create a relationship between different IdM entries. While this is obvious in traditional group-member roles, it is also true for some policy entries (like SELinux and sudo policies) where entries are associated with another entry.

Most commonly, the command format for adding a member entry is *** -add-member**, although the command may specify an entry type, such as *** -add-user**.

Likewise, entries are removed as members (not deleted) using a *** -remove-member** or *** -remove-type** command.

8.2.2. Positional Elements in ipa Commands

Usually, **ipa** subcommands have only two elements: the name of the entry being modified (the *object*) and then any options available for the subcommand:

```
ipa command entryName --options=values
```

With a few types of entries, however, not only the entry name itself needs to be specified; the entry's *parent* must also be specified. This is the case with **automount** commands, for example. With automount, the location must be included whenever a new key or map is created.

The parent entry name is given first, and then the child entry name. For example, for automount, the location is given first, and then the map or key entry name.

```
ipa command parentEntryName childEntryName --childOptions=childValues
```

8.2.3. Managing Entry Attributes with --setattr, --addattr, and --delattr

All identities and configuration in Identity Management are stored as LDAP entries, with standard attribute-value assertions (AVAs). Whether an entry is created through the UI or the CLI, there are certain attributes which are required and others which are available, depending on the default and custom object classes for that entry type.

For the most common attributes, the **ipa** use specified command-line arguments to set values. For example, adding a mail attribute to a user can be done with the **--mail** argument; enabling dynamic updates for a DNS zone can be done with the **--allow-dynupdate** option with zone commands; and a map key for an automount map is given in the **--key** option.

However, entries can also allow attributes that may not have command-line (or UI) options for setting them. Partially, this is because the underlying LDAP schema is very rich, particularly for user entries, with many possible allowed attributes. Additionally, Identity Management allows schema extensions for users and groups, and those custom schema elements are not necessarily reflected in the UI or command-line tools.

Any supported attribute can be added or edited to an entry using the **--setattr** and **--addattr** options.

Both options have this format:

```
--setattr=attribute=value
```

The **--setattr** option sets one value for the given attribute; any existing values are overwritten, even for multi-valued attributes.

The **--addattr** option adds a new value for an attribute; for a multi-valued attribute, it adds the new

value while preserving any existing values.

Both **--setattr** option and **--addattr** can be used multiple times in the same command invocation. For example:

```
$ ipa user-mod jsmith --addattr=mail=johnnys@me.com --
addattr=mail=jsmith@example.com --setattr=description="backup IT manager for the
east coast branch"
```

Likewise, an attribute or specific attribute value can be removed from an entry using the **--delattr** option. For a single-valued attribute, this removes the attribute; for a multi-valued attribute, it removes only the specified value. For example:

```
$ ipa user-mod jsmith --delattr=mail=johnnys@me.com
```



NOTE

Deleting attributes is evaluated last, after adding or editing attributes. If the same attribute is added and deleted in the same modify operation, it is a no-op.

```
$ ipa user-mod jsmith --addattr=mail=johnnys@me.com --
delattr=mail=johnnys@me.com
```

8.2.4. Using Special Characters with IdM Tools

The IdM command-line tools are run as any other utilities in a shell. If there are special characters in the command — such as angle brackets (> and <), ampersands (&), asterisks (*), and pipes (|) — the characters must be escaped. Otherwise, the command fails because the shell cannot properly parse the unescaped characters.

8.2.5. Logging into the IdM Domain Before Running

Before running any IdM commands (with the exception of the installation scripts, such as **ipa-server-install**), the user must first authenticate to the IdM domain by obtaining a Kerberos ticket. This is done using **kinit**:

```
[jsmith@ipaserver ~]$ kinit admin
```

Different login options are described in [Section 8.3, “Logging into IdM”](#).

8.3. Logging into IdM

Users are authenticated to IdM services, including the command-line tools and the web UI, using Kerberos authentication. This means that logging into Identity Management requires running **kinit**.

Running **kinit** issues the user a Kerberos *ticket*. This ticket is checked by any IdM or Kerberos-aware service, so that a user only needs to log in once to access all domain services. Domain services include the IdM web UI, mounted file shares, wikis, or any other application which uses IdM as its identity/authentication store.

8.3.1. Logging into IdM

Logging into Identity Management requires running **kinit** on a client within the IdM domain.

```
$ kinit
```

The **kinit** command must be run from a machine which has been configured as a client within the IdM domain, so that the client retrieves authenticates with the IdM KDC.

Simply running **kinit** logs into IdM as the currently logged-in user account. This user account must also be an IdM user for them to authenticate to the IdM Kerberos domain successfully. For example, if you are logged into the machine as **jsmith**:

```
$ kinit
Password for jsmith@EXAMPLE.COM:
```

NOTE

If SSSD or **pam_krb5** is configured on the IdM client machine, then when a user logs into the machine, a ticket is created which can be used for machine services which require authentication, such as **sudo**.

8.3.2. Logging in When an IdM User Is Different Than the System User

To specify an IdM username — because a person's system username is different then the IdM username or to switch IdM user accounts — simply rerun the **kinit** command, specifying the new user. For example:

```
$ kinit userName
Password for userName@EXAMPLE.COM:
```

When the server was first set up, an administrative user, **admin**, is created to perform normal administrative activities. To authenticate as the admin user, use the name admin when running **kinit**:

```
$ kinit admin
```

NOTE

Only one set of tickets can be stored per logged-in user. The current stored credentials are the ones that will be used when accessing IdM services.

If you were already connected to the IdM web UI as another user, refresh the browser to display the updated details for the new user.

8.3.3. Checking the Current Logged in User

Use the **klist** command to verify the identity and the ticket granting ticket (TGT) from the server:

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: ipaUser@EXAMPLE.COM

Valid starting      Expires            Service principal
11/10/08 15:35:45  11/11/08 15:35:45  krbtgt/EXAMPLE.COM@EXAMPLE.COM

Kerberos 4 ticket cache: /tmp/tkt500
klist: You have no tickets cached
```

It's important to know who the authenticated user is because the currently-authenticated user is the only one who can access the IdM services. The Kerberos client libraries for **krb5** have some limitation, one of them being that the current ticket is overwritten with any new invocation of **krb5**. Authenticating as User A and then authenticating as User B overwrites User A's ticket.

To allow there to be multiple authenticated users on a machine, set the **KRB5CCNAME** environment variable. This variable keeps credential caches separate in different shells.

8.3.4. Caching User Kerberos Tickets

Only one set of tickets can be stored per logged-in user. The current stored credentials are the ones that will be used when accessing IdM services.

For example, if you authenticated as **admin**, added a new user, set the password, and then tried to authenticate as that user, the administrator's ticket is lost.

To keep separate credential caches in different shells, a special environment variable, **KRB5CCNAME**, can be used.

8.4. Using the IdM Web UI

In order to use the web UI, the user must be authenticated with the IdM Kerberos domain and have an active Kerberos ticket ([Section 8.3, “Logging into IdM”](#)). Generally, the web UI can only be accessed from an IdM server or client machine and the user must be locally authenticated. There are a couple of ways to work around this, either by configuring Kerberos on a non-domain machine to connect to the Kerberos domain ([Section 8.4.5, “Using a Browser on Another System”](#)) or by password authentication to the UI.

8.4.1. Supported Web Browsers

These browsers are supported for connecting to the web UI:

- Firefox 17 and higher
- Internet Explorer 8 (limited to personal account management)

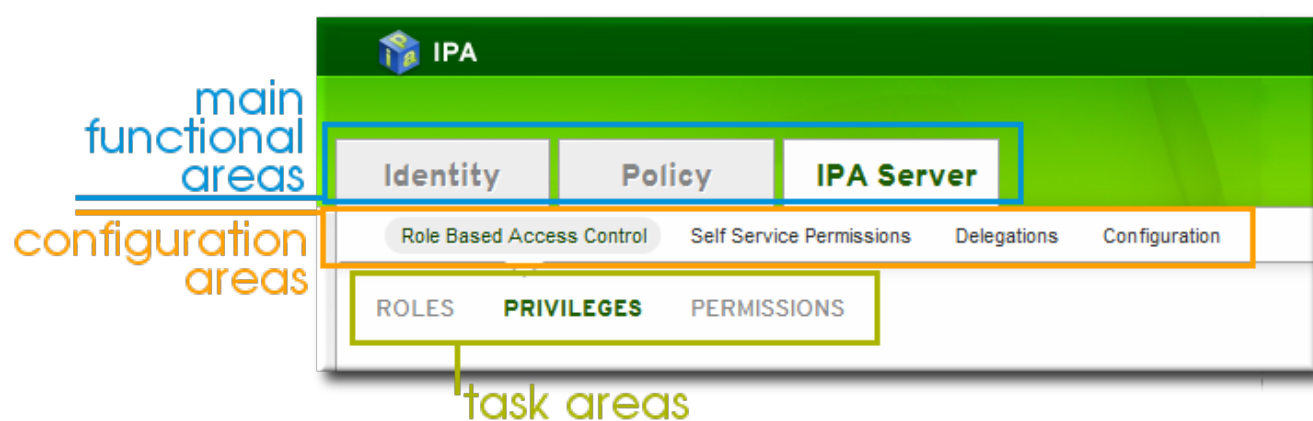
8.4.2. About the Web UI

The web UI has three major functional areas which correspond to each of the major functions of IdM: identity management, policy management, and domain configuration.

Table 8.1. Configuration Areas Per Tab

Main Menu Tab	Configuration Areas
Identity	▸ User entries ▸ User groups entries ▸ Host/client entries ▸ Host group entries ▸ Netgroups entries ▸ Domain services entries ▸ DNS (if configured)
Policy	▸ Host-based access control ▸ Sudo rules ▸ Automount ▸ User password policies ▸ Kerberos ticket policy
IdM Server (access controls within Identity Management)	▸ Role-based access control (permissions based on group membership) ▸ Self permissions ▸ Delegations (user access control over other users)

The *main menu* at the top of every page has three tabs which correspond to the functional areas listed in [Table 8.1, “Configuration Areas Per Tab”](#). When a tab is selected, there is a submenu of the different configuration areas. Some configuration areas may have multiple possible entries; for example, role-based access controls define user roles/groups, the areas that access can be granted or denied (privileges), and then the permissions granted to those areas. Each separate configuration entry has its own task area beneath the primary configuration area.

**Figure 8.1. The Main Menu**

8.4.3. Opening the IdM Web UI

The browser must be properly configured, as described in [Section 8.4.4, “Configuring the Browser”](#), to support Kerberos authentication so that the user can connect to the UI.

To open the web UI:

1. Get a valid Kerberos ticket using **kinit**, as in [Section 8.3, “Logging into IdM”](#).
2. Open the IdM URL. The full URL is **https://IPAserver-FQDN/ipa/ui**, but this service is also accessed simply by opening **https://IPAserver-FQDN**. For example:

```
https://server.example.com
https://server.example.com/ipa/ui
```

8.4.4. Configuring the Browser

Firefox can use Kerberos credentials to authenticate to the IdM UI, but Kerberos negotiation needs to be configured to use the IdM domain. At the first log-in attempt, if Firefox has not been configured to support Kerberos authentication, then an error message appears.

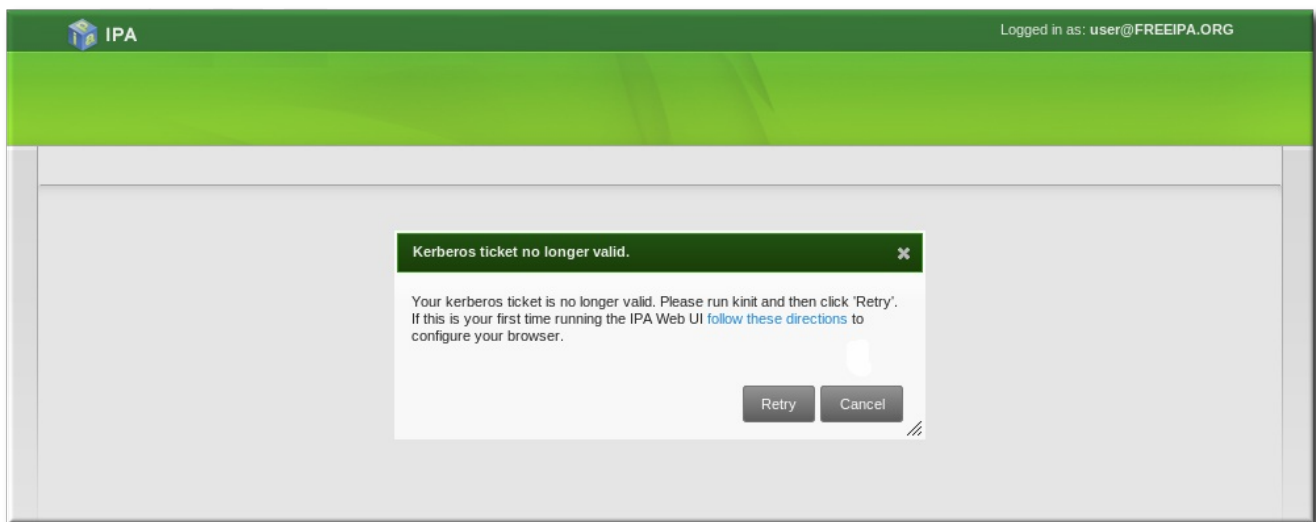
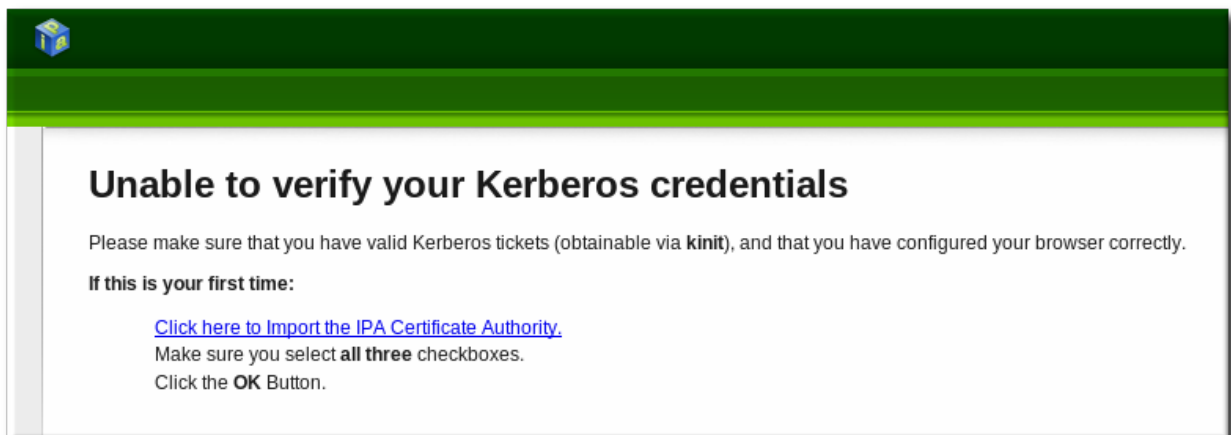


Figure 8.2. Kerberos Authentication Error

If you see that error, then the IdM web UI can perform the required configuration:

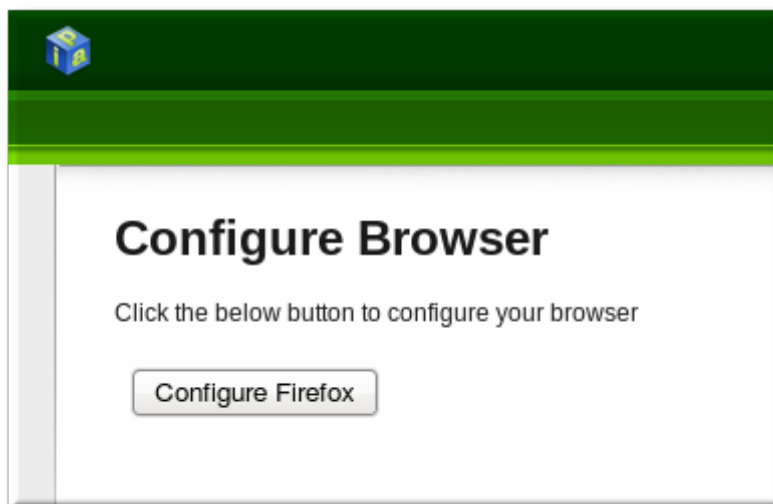
1. Click the **follow these directions** link.
2. Click the link to import the CA certificate for the IdM server.



3. Set the web site and software developer (first and last) trust bits for the CA certificate.



4. Click the **Configure Firefox** button. This automatically fills out all the **negotiate** settings in the Firefox configuration to use the IdM domain settings.



When the process is complete, a success box pops up saying that Firefox has been configured for single sign-on. For there, you are redirected to the IdM web UI.



This can also be done manually:

1. Open Firefox.
2. Type **about:config** in the address bar.
3. In the **Search** field, type **negotiate** to filter out the Kerberos-related parameters.

- On Red Hat Enterprise Linux, enter the domain name for the URI parameters, including the preceding period (.) and set the **gsslib** parameter to true:

```
network.negotiate-auth.trusted-uris .example.com
network.negotiate-auth.using-native-gsslib true
```

On Windows, set the trusted URIs and library path, and disable the built-in Microsoft Kerberos for authentication:

```
network.negotiate-auth.trusted-uris .example.com
network.auth.use-sspi false
network.negotiate-auth.gsslib: C:\Program Files\MIT\Kerberos\bin\gssapi32.dll
```

On a 64-bit system, the library location is in **C:\Program Files(x86)\MIT\Kerberos\bin\gssapi32.dll**.

- Open the web UI by going to the fully-qualified domain name of the IdM server such as **http://ipaserver.example.com**. Make sure that you can open the web UI and that there are no Kerberos authentication errors.
- Next, download the IdM server's CA certificate from **http://ipa.example.com/ipa/config/ca.crt**.
- Select the first (**Trust this CA to identify web sites**) and third (**Trust this CA to identify software developers**) check boxes.

8.4.5. Using a Browser on Another System

It is possible to connect to the Identity Management web UI from a system which is *not* a member of the IdM domain. In this case, it is possible to specify an IdM-specific Kerberos configuration file on the external (non-IdM) machine before running **kinit**, and then the user can authenticate against the IdM server domain.

This is especially useful there are multiple realms or overlapping domains across your infrastructure.

- Copy the **/etc/krb5.conf** file from the IdM server.

```
# scp /etc/krb5.conf root@externalmachine.example.com:/etc/krb5_ipa.conf
```



WARNING

Do not overwrite the existing **krb5.conf** file.

- On the external machine, set the terminal session to use the copied IdM Kerberos configuration file:

```
$ export KRB5_CONFIG=/etc/krb5_ipa.conf
```

- Configure Firefox on the external machine as in [Section 8.4.4, “Configuring the Browser”](#).

8.4.6. Logging in with Simple Username/Password Credentials

If Kerberos authentication fails, then browser login also fails. That prevents access to the IdM web UI. Simple authentication for the UI allows users to log in even if there are problems with the Kerberos service or if the system is outside the IdM domain.

When the IdM server cannot find a valid Kerberos ticket for the user attempting to log into the web UI, it splashes an error message. Since the preferred method of connecting to IdM domain services (including the UI) is using Kerberos authentication, the error first says to renew the Kerberos credentials or to configure the browser to support Kerberos authentication.

The second part of the message offers the alternative of using simple authentication. The **form-based authentication** link opens a login page.

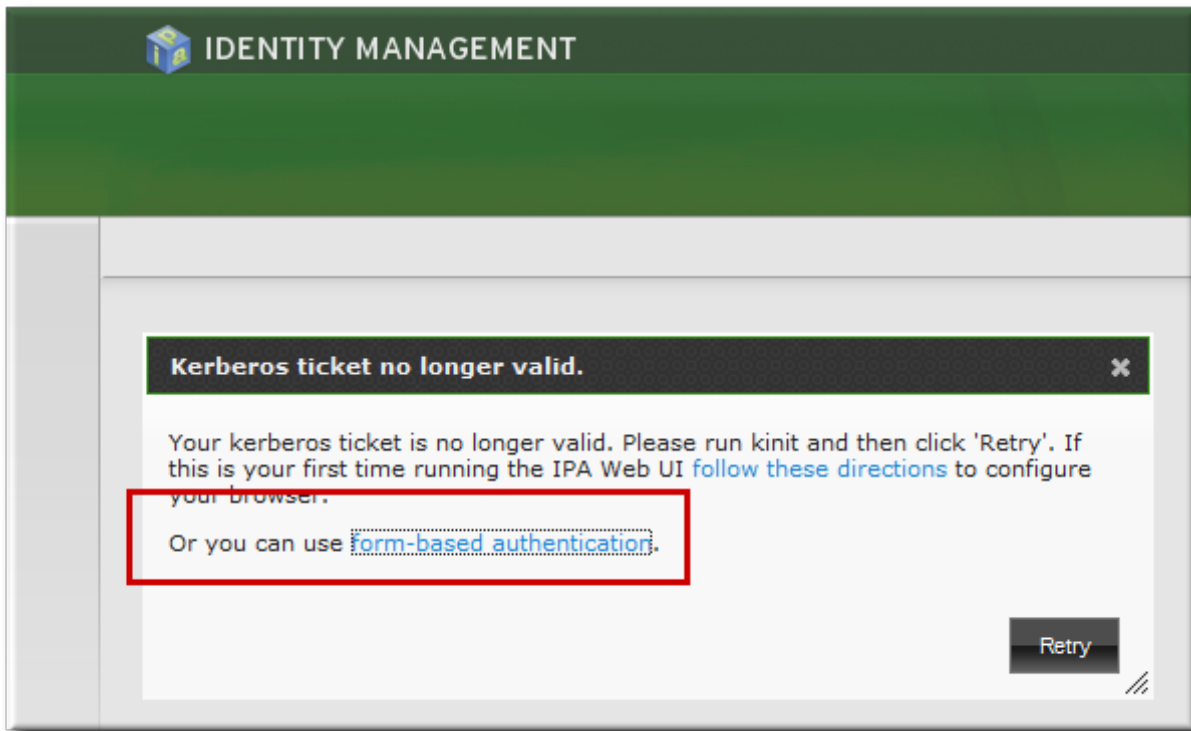


Figure 8.3. IdM Form-Based Login Option

Then simply supply the UID and password for a configured IdM user.

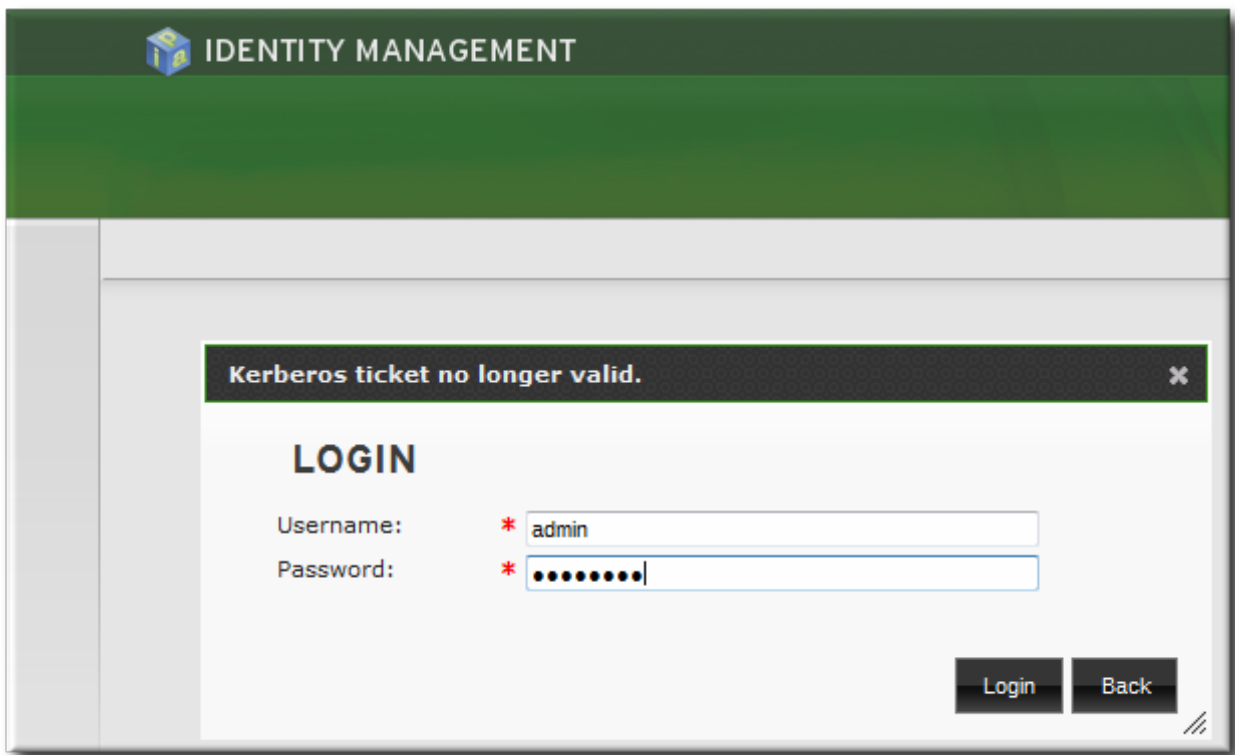


Figure 8.4. IdM Password Prompt

8.4.7. Using the UI with Proxy Servers

Proxy servers can be used to access the web UI without any additional configuration in IdM.

Port forwarding is not supported with the IdM server. However, because it is possible to use proxy servers with IdM, an operation similar to port forwarding can be configured using proxy forwarding with OpenSSH and the SOCKS option. This is described in <http://www.meadowdy.org/~gotoh/ssh/openssh-socks.html>.

[1] The hostname used in the directory lookup can be controlled in the `/etc/ipa/default.conf` configuration file.

Chapter 9. Identity: Managing Users and User Groups

Users in Identity Management are able to access services and servers within the domain through Kerberos authentication. This chapter covers general management tasks for users, groups, password policies, and other configuration for users.

9.1. Setting up User Home Directories

A home directory is required for any IdM user. Without a home directory in the expected location, a user may be unable to log into the domain. While systems administrators can manage home directories outside of IdM, it is also possible to use a PAM module to create home directories automatically on both IdM servers and clients.

9.1.1. About Home Directories

IdM, as part of managing users, can manage user home directories. However, IdM has certain defined parameters for any managed home directories:

- The default prefix for users' home directories is **/home**.
- IdM does not automatically create home directories when users log in. Automatically creating home directories requires either the **pam_oddjob_mkhomedir** module or the **pam_mkhomedir** module. This module can be configured as part of client installation or after installation, as described in [Section 9.1.2, “Enabling the PAM Home Directory Module”](#).

The home directory process for IdM first attempts to use the **pam_oddjob_mkhomedir** module because this requires fewer user privileges and access to create the home directories, as well as integrating smoothly with SELinux. If this module is not available, then the process falls back to the **pam_mkhomedir** module.



NOTE

On Red Hat Enterprise Linux 5 clients, the client installation script uses the **pam_mkhomedir** module even if the **pam_oddjob_mkhomedir** module is available. To use the **pam_oddjob_mkhomedir** module on Red Hat Enterprise Linux 5, edit the PAM configuration manually.

- It is possible to use an NFS file server that provides **/home** that can be made available to all machines in the domain and then automounted on the IdM server.

There are potential issues when using NFS, such as security issues related to granting root access to the NFS user, performance issues with loading the entire **/home** tree, and network performance issues for using remote servers for home directories. There are some general guidelines for using NFS with Identity Management:

- Use automount to mount only the user's home directory and only when the user logs in, rather than loading the entire **/home** tree.
- Use a remote user who has limited permissions to create home directories and mount the share on the IdM server as that user. Since the IdM server runs as an **httpd** process, it is possible to use **sudo** or a similar program to grant limited access to the IdM server to create home directories on the NFS server.
- Use a mechanism, such as the **pam_oddjob_mkhomedir** module, to create the home directory as that user.

Using automounts for home directories is described in [Section 9.1.3, “Manually Mounting Home Directories”](#).

- If a suitable directory and mechanism are not available for to create home directories, users may not be able to log in.

9.1.2. Enabling the PAM Home Directory Module

For a home directory to be created automatically when a user logs in, IdM can use either the **pam_oddjob_mkhomedir** module or the **pam_mkhomedir** module. Because it requires fewer permissions and works well with SELinux, IdM preferentially uses the **pam_oddjob_mkhomedir** module. If that module is not installed, then it falls back to the **pam_mkhomedir** module.



NOTE

IdM does not require the **pam_oddjob_mkhomedir** module or **pam_mkhomedir** module. This is because the ***_mkhomedir** module may try to create home directories even when the shared storage is not available. If the module is unable to create the home directory, then users can be blocked from logging into the IdM domain.

The system administrator must activate this module on each client or server as needed.

There are two ways to enable the **pam_oddjob_mkhomedir** (or **pam_mkhomedir**) module:

- The **--mkhomedir** option can be used with the **ipa-client-install** command. While this is possible for clients, this option is not available to servers when they are set up.
- The **pam_oddjob_mkhomedir** module can be enabled using the system's **authconfig** command. For example:

```
authconfig --enablemkhomedir --update
```

This option can be used for both server and client machines post-installation.



NOTE

On Red Hat Enterprise Linux 5 clients, the client installation script uses the **pam_mkhomedir** module even if the **pam_oddjob_mkhomedir** module is available. To use the **pam_oddjob_mkhomedir** module on Red Hat Enterprise Linux 5, edit the PAM configuration manually.

9.1.3. Manually Mounting Home Directories

While PAM modules can be used to create home directories for users automatically, this may not be desirable behavior in every environment. In that case, home directories can be manually added to the IdM server from separate locations using NFS shares and **automount**.

1. Create a new location for the user directory maps:

```
$ ipa automountlocation-add userdirs
Location: userdirs
```

2. Add a direct map to the new location's **auto.direct** file. In this example, the mount point is **/share**:

```
$ ipa automountkey-add userdirs auto.direct --key=/share --info="-ro,soft,
ipaserver.example.com:/home/share"
```

Key: /share

Mount information: -ro,soft, ipaserver.example.com:/home/share

Using automounts with IdM is described in detail in [Chapter 17, Policy: Using Automount](#).

9.2. Managing User Entries

9.2.1. About Username Formats

The default length for usernames is 32 characters.

IdM supports a wide range of username formats, based on this regular expression:

```
[a-zA-Z0-9_][a-zA-Z0-9_.-]{0,252}[a-zA-Z0-9_.$-]?
```



TIP

The trailing \$ symbol is permitted for Samba 3.x machine support.

Any system limits — such as starting a username with a number on Unix systems — apply to the usernames in IdM.



NOTE

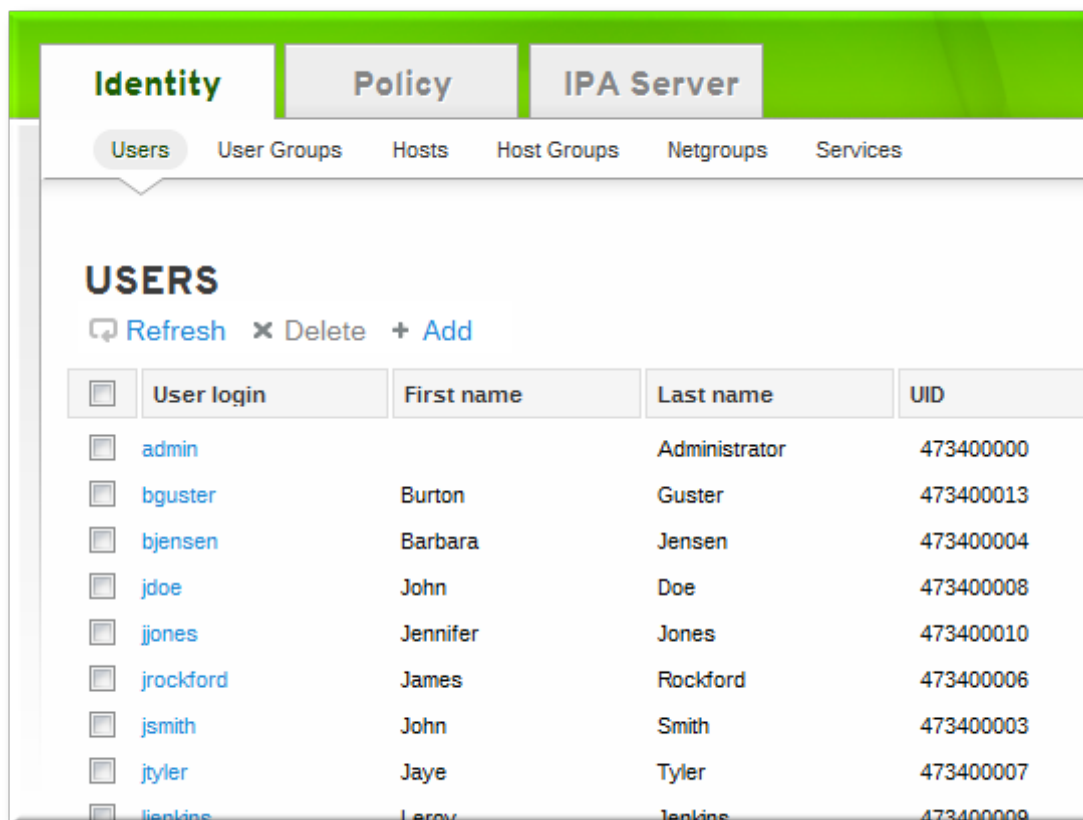
Usernames are case insensitive when they are created, meaning that any case letter can be entered but case is ignored when the username is saved.

Username are automatically normalized to be all lower case, even if the user is created with mixed case or upper case letters.

9.2.2. Adding Users

9.2.2.1. From the Web UI

1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the **Add** link at the top of the users list.



- Fill in the user's first and last names. The user login (UID) is automatically generated based on the user's full name, but this can be set manually by clicking the **Optional field** link.

The screenshot shows the 'Add User' dialog box. It has a title bar with 'Administrator', 'Enabled', and '172200000'. The dialog contains the following fields:

- User login:
- First name: (marked with a red asterisk)
- Last name: (marked with a red asterisk)
- New Password:
- Verify Password:

Below the fields is a legend: *** Required field**. At the bottom are four buttons: Add, Add and Add Another, Add and Edit, and Cancel.



NOTE

Username are case insensitive when they are created, meaning that case is ignored. Username are automatically normalized to be all lower case, even if the user is created with mixed case or upper case letters.

- Click the **Add and Edit** button to go directly to the expanded entry page and fill in more attribute information, as in [Section 9.2.3.1, "From the Web UI"](#). The user entry is created with some basic information already filled in, based on the given user information and the user entry template.

The screenshot shows the Identity Management web interface. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Below these, a navigation bar includes 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', 'Services', and 'DNS'. The 'Users' tab is selected, and the breadcrumb 'Users » jsmith' is shown. The main heading is 'USER: jsmith'. Below this, it says 'jsmith is a member of:' followed by a row of tabs: 'Settings' (selected), 'User Groups (1)', 'Netgroups', 'Roles', 'HBAC Rules', and 'Sudo Rules'. There are three buttons: 'Refresh', 'Reset', and 'Update'. The 'IDENTITY SETTINGS' section contains input fields for 'Job Title', 'First name' (John), 'Last name' (Smith), 'Full name' (John Smith), 'Display name' (John Smith), and 'Initials' (JS). The 'ACCOUNT SETTINGS' section shows 'Status: Enabled: Click to Disable', 'User login: jsmith', 'Password: Reset Password', and 'UID: 172200003'.

9.2.2.2. From the Command Line

New user entries are added with the **user-add** command. Attributes (listed in [Table 9.2, “Default Identity Management User Attributes”](#)) can be added to the entry with specific values or the command can be run with no arguments.

```
$ ipa user-add [username] [attributes]
```

When no arguments are used, the command prompts for the required user account information and uses the defaults for the other attributes, with the defaults printed below. For example:

```
$ ipa user-add
First name: John
Last name: Smith
User login [jsmith]: jsmith
-----
Added user "jsmith"
-----
User login: jsmith
First name: John
Last name: Smith
Home directory: /home/jsmith
GECOS field: jsmith
Login shell: /bin/sh
Kerberos principal: jsmith@EXAMPLE.COM
UID: 387115841
```

Any of the user attributes can be passed with the command. This will either set values for optional attributes or override the default values for default attributes.

```
$ ipa user-add jsmith --first=John --last=Smith --manager=bjensen --
email=johnls@example.com --homedir=/home/work/johns --password
```



NOTE

Username are case insensitive when they are created, meaning that case is ignored. Username are automatically normalized to be all lower case, even if the user is created with mixed case or upper case letters.



IMPORTANT

When a user is created without specifying a UID or GID number, then the user account is automatically assigned an ID number that is next available in the server or replica range. (Number ranges are described more in [Section 9.8, “Managing Unique UID and GID Number Assignments”](#).) This means that a user always has a unique number for its UID number and, if configured, for its private group.

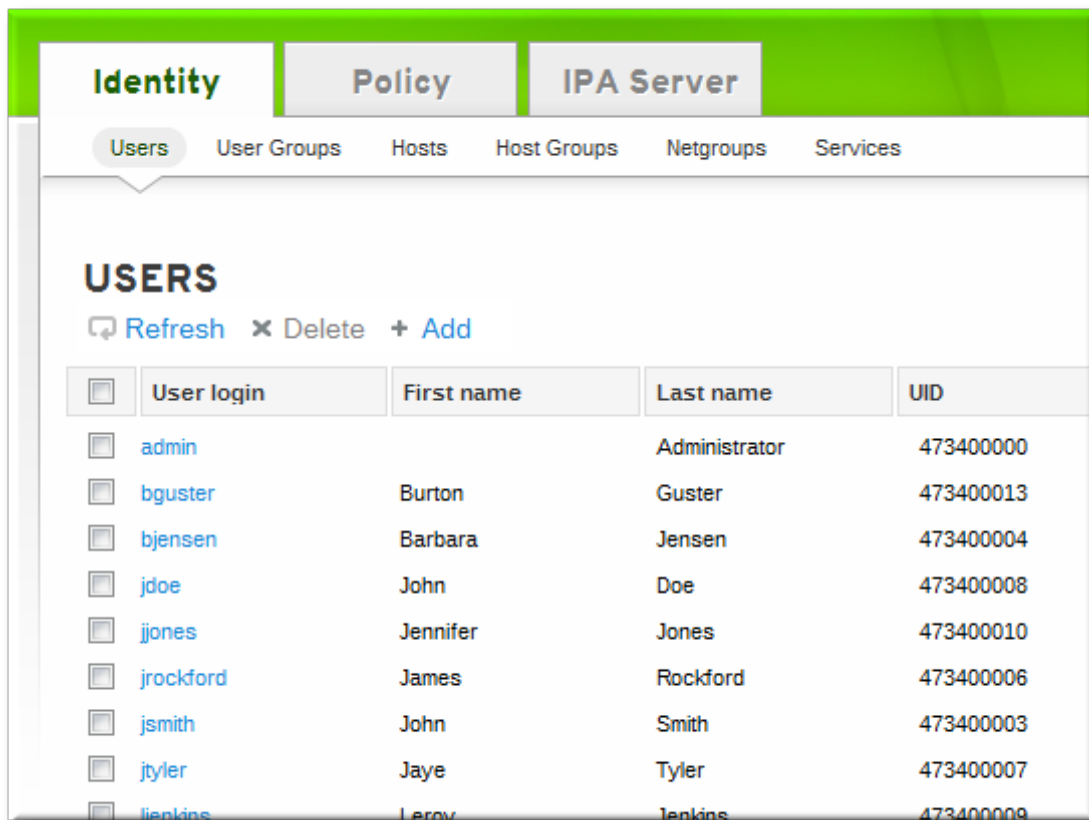
If a number is *manually* assigned to a user entry, the server does not validate that the **uidNumber** is unique. It will allow duplicate IDs; this is expected (though discouraged) behavior for POSIX entries.

If two entries are assigned the same ID number, only the first entry is returned in a search for that ID number. However, both entries will be returned in searches for other attributes or with **ipa user-find --all**.

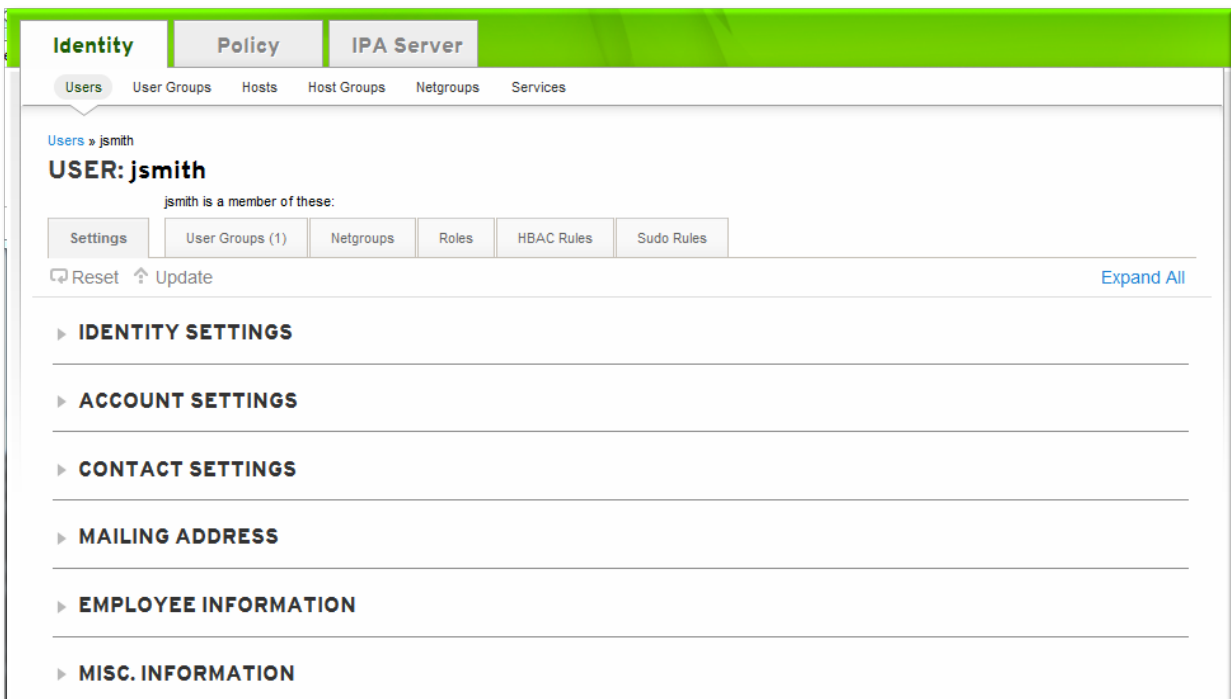
9.2.3. Editing Users

9.2.3.1. From the Web UI

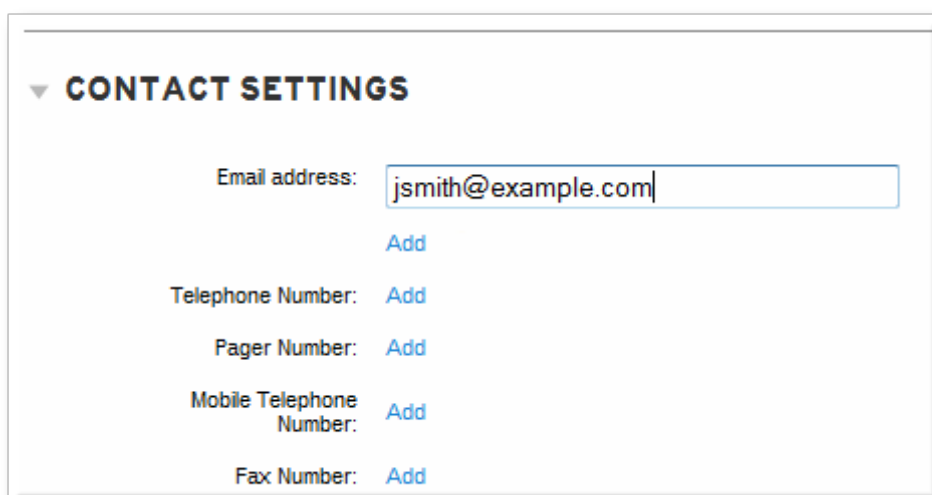
1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user to edit.



3. There are a number of different types of attributes that can be edited for the user. All of the default attributes are listed in [Table 9.2, “Default Identity Management User Attributes”](#). Most of the attributes in the **Identity Settings** and **Account Settings** areas have default values filled in for them, based on the user information or on the user entry template.



4. Edit the fields or, if necessary, click the **Add** link by an attribute to create the attribute on the entry.



▼ **CONTACT SETTINGS**

Email address: [Add](#)

Telephone Number: [Add](#)

Pager Number: [Add](#)

Mobile Telephone Number: [Add](#)

Fax Number: [Add](#)

5. When the edits are done, click the **Update** link at the top of the page.

9.2.3.2. From the Command Line

The **user-mod** command edits user accounts by adding or changing attributes. At its most basic, the **user-mod** specifies the user account by login ID, the attribute to edit, and the new value:

```
$ ipa user-mod loginID --attributeName=newValue
```

For example, to change a user's work title from *Editor II* to *Editor III*:

```
$ ipa user-mod jsmith --title="Editor III"
```

Identity Management allows *multi-valued* attributes, based on attributes in LDAP that are allowed to have multiple values. For example, a person may have two email addresses, one for work and one for personal, that are both stored in the mail attribute. Managing multi-valued attributes can be done using the **--addattr** option.

If an attribute allows multiple values — like mail — simply using the command-line argument will overwrite the value with the new value. This is also true for using **--setattr**. However, using **--addattr** will add a new attribute; for a multi-valued attribute, it adds the new value in addition to any existing values.

Example 9.1. Multiple Mail Attributes

A user is created first using his work email account.

```
$ ipa user-add jsmith --first=John --last=Smith --email=johnls@example.com
```

Then, his personal email account is added.

```
$ ipa user-mod jsmith --addattr=mail=johnnys@me.com
```

Both email addresses are listed for the user.

```
$ ipa user-find jsmith --all
-----
1 user matched
-----
dn: uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
User login: jsmith
.....
Email address: jsmith@example.com, jsmith@new.com
```

To set two values at the same time, use the **--addattr** option twice:

```
$ ipa user-add jsmith --first=John --last=Smith --email=johnls@example.com --
addattr=mail=johnnys@me.com --addattr=mail=admin@example.com
```

9.2.4. Deleting Users

Deleting a user account permanently removes the user entry and all its information from IdM, including group memberships and passwords. External configuration — like a system account and home directory — will still exist on any server or local machine where they were created, but they cannot be accessed through IdM.

Deleting a user account is permanent. The information cannot be recovered; a new account must be created.

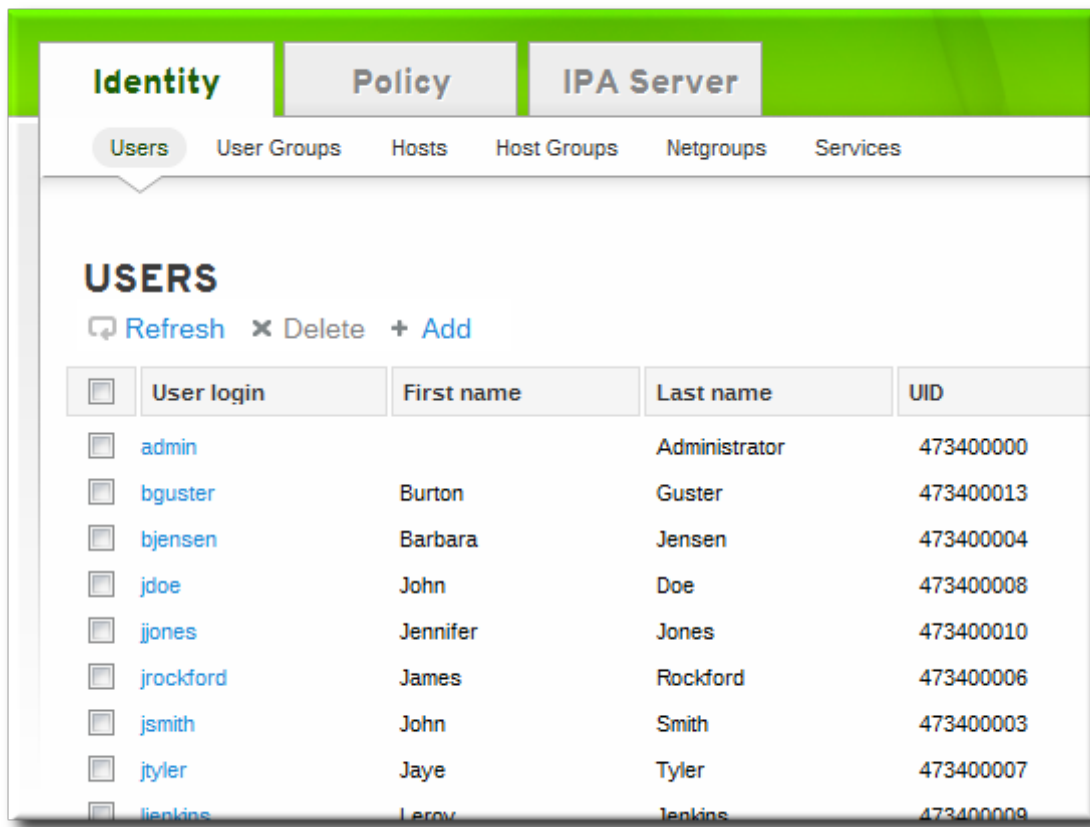
**NOTE**

If all admin users are deleted, then you must use the Directory Manager account to create a new administrative user.

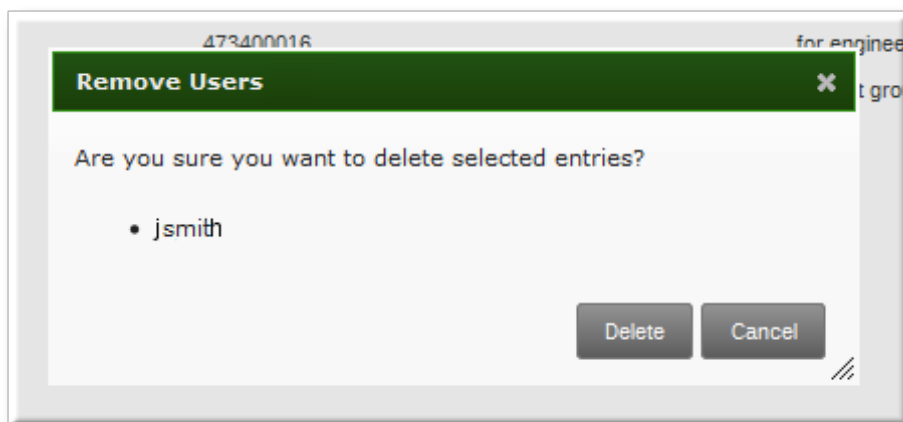
Alternatively, any user who belongs in the group management role can also add a new admin user.

9.2.4.1. With the Web UI

1. Open the **Identity** tab, and select the **Users** subtab.
2. Select the checkboxes by the names of the users to delete.



3. Click the **Delete** link at the top of the task area.
4. When prompted, confirm the delete action.



9.2.4.2. From the Command Line

Users are deleted using the **user-del** command and then the user login. For example, a single user:

```
$ ipa user-del jsmith
```

To delete multiple users, simply list the users, separated by spaces.

```
$ ipa user-del jsmith bjensen mreynolds cdickens
```

When deleting multiple users, use the **--continue** option to force the command to continue regardless of errors. A summary of the successful and failed operations is printed to stdout when the command completes. If **--continue** is not used, then the command proceeds with deleting users *until* it encounters an error, and then it exits.

9.3. Managing Public SSH Keys for Users

OpenSSH uses *public-private key pairs* to authenticate users. A user attempts to access some network resource and presents its key pair. The first time the user authenticates, the administrator on the target machine has to approve the request manually. The machine then stores the user's public key in an **authorized_keys** file. Any time that the user attempts to access the resource again, the machine simply checks its **authorized_keys** file and then grants access automatically to approved users.

There are a couple of problems with this system:

- ▶ SSH keys have to be distributed manually and separately to all machines in an environment.
- ▶ Administrators have to approve user keys to add them to the configuration, but it is difficult to verify either the user or key issuer properly, which can create security problems.

On Red Hat Enterprise Linux, the System Security Services Daemon (SSSD) can be configured to cache and retrieve user SSH keys so that applications and services only have to look in one location for user keys. Because SSSD can use Identity Management as one of its identity information providers, Identity Management provides a universal and centralized repository of keys. Administrators do not need to worry about distributing, updating, or verifying user SSH keys.

9.3.1. About the SSH Key Format

When keys are uploaded to the IdM entry, the key format can be either an [OpenSSH-style key](#) or a raw [RFC 4253-style blob](#). Any RFC 4253-style key is automatically converted into an OpenSSH-style key before it is imported and saved into the IdM LDAP server.

The IdM server can identify the type of key, such as an RSA or DSA key, from the uploaded key blob. However, in a key file such as **id_rsa.pub**, a key entry is identified by its type, then the key itself, and then an additional comment or identifier. For example, for an RSA key associated with a specific hostname:

```
"ssh-rsa ABCD1234...== ipaclient.example.com"
```

All three parts from the key file can be uploaded to and viewed for the user entry, or only the key itself can be uploaded.

9.3.2. Uploading User SSH Keys Through the Web UI

1. Generate a user key. For example, using the OpenSSH tools:


```
[jsmith@server ~]$ ssh-keygen -t rsa -C jsmith@example.com
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jsmith/.ssh/id_rsa):
Created directory '/home/jsmith/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jsmith/.ssh/id_rsa.
Your public key has been saved in /home/jsmith/.ssh/id_rsa.pub.
The key fingerprint is:
a5:fd:ac:d3:9b:39:29:d0:ab:0e:9a:44:d1:78:9c:f2 jsmith@example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           + .       |
|        + =  .       |
|       =  +         |
|      . E S .        |
|     .   . . 0        |
|    . . . . 00.       |
|   . 0 . . +. +0      |
|    0  .0..0+0        |
+-----+

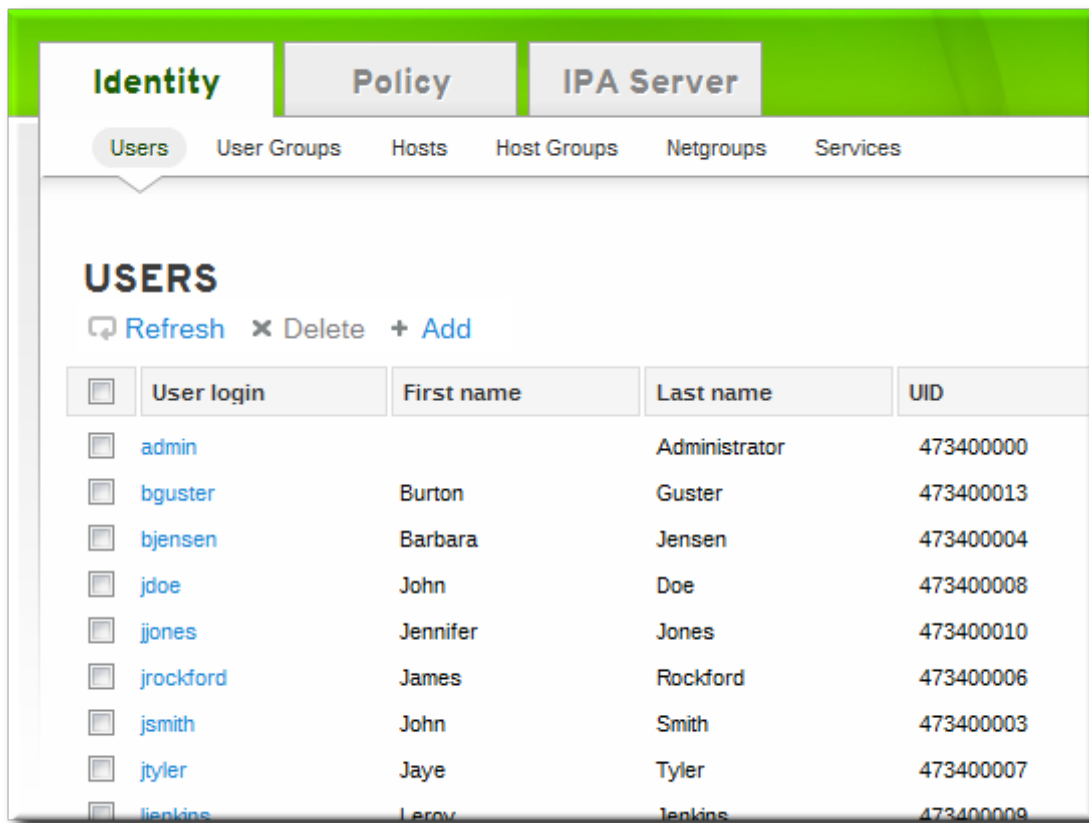
```

2. Copy the public key from the key file. The full key entry has the form *type key== comment*. Only the *key==* is required, but the entire entry can be stored.

```
[jsmith@server ~]$ cat /home/jsmith/.ssh/id_rsa.pub

ssh-rsa AAAAB3NzaC1yc2E...tJG1PK2Mq++wQ== jsmith@example.com
```

3. Open the **Identity** tab, and select the **Users** subtab.
4. Click the name of the user to edit.



5. In the **Account Settings** area of the **Settings** tab, click the **SSH public keys: Add** link.

The screenshot displays the Red Hat Identity Management (IdM) web interface. At the top, there are three main tabs: **Identity**, **Policy**, and **IPA Server**. Under the **Identity** tab, there are sub-tabs: **Users**, **User Groups**, **Hosts**, **Host Groups**, **Netgroups**, and **Services**. The **Users** sub-tab is selected, showing a breadcrumb [Users](#) » **jsmith**. Below this, the user is identified as **✓ USER: jsmith**. To the right of the user name is a dropdown menu with the text "-- select action --" and an **Apply** button. Below the user name, it states "jsmith is a member of:" followed by a row of tabs: **Settings**, **User Groups (1)**, **Netgroups**, **Roles**, **HBAC Rules**, and **Sudo Rules**. The **Settings** tab is active. Below the tabs are three buttons: **Refresh**, **Reset**, and **Update**. The main content area is divided into sections: **IDENTITY SETTINGS** (expanded), **ACCOUNT SETTINGS** (expanded), and **PASSWORD POLICY** (collapsed). Under **ACCOUNT SETTINGS**, the following fields are visible: **User login:** **jsmith**, **Password:** *********, **Password expiration:** **Mon Oct 22 2012 11:14:37 GMT-0500 (Central Daylight Time)**, **UID:** **951400001**, **GID:** **951400001**, **Login shell:** **/bin/sh**, and **Home directory:** **/home/jsmith**. A red box highlights the **SSH public keys:** **Add** link. The **PASSWORD POLICY** section is collapsed.

6. The UI opens a new link, **New: key not set Show/Set key**. Click the **Show/Set key** link.

▼ ACCOUNT SETTINGS

User login: **jsmith**

Password: *********

Password expiration: **Mon Oct 22 2012 11:14:37 GMT-0500 (Central Daylight Time)**

UID: *

GID: *

Login shell:

Home directory:

SSH public keys: **New: key not set** [Show/Set key](#) [undo](#)

[Add](#) [undo all](#)

7. Paste in the public key for the user, and click the **Set** button.

Set SSH key ✕

SSH public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA6HsLXndrd+P+55SdJIrdMIPeIfKJEzucLNioGsC7
59JVwpuul6sQE1Kuu6Vec4Q5Jh7Ork2ERkxSxwxf+Pka5oN+M3sbaA+PBaQykBn4LAQ2
DvG0BSox8ObU2Cyds0Zuk+jQ7Ni13Qxka0rA1lCgyGJT5T0nmFBHqTWhOKs81RBFbtmj
Ps75+MziNP1Mik8a7TD3s6SubH23VtB9SYd9OiKsouuI7+fhbR7+JaFUtT0c8sU9JP4o
olKHUZeDcP7c666nHPmvmP2ItsqnzkkCiGB1JZPKMiOaX2jFqryU709DBDZMiAiAEVOP
qVJCtg5py0MYHyRZ1HzNqjr6xr7Q4w== jsmith@example.com
```

[Set](#) [Cancel](#)

The **SSH public keys** field now shows **New: key set**. Clicking the **Show/Set key** link opens the submitted key.

8. To upload multiple keys, click the **Add** link below the list of public keys, and upload the other keys.
9. When all the keys have been submitted, click the **Update** link at the top of the user's page to save the changes.

When the public key is saved, the entry is displayed as the key fingerprint, the comment (if one was

included), and the key type [2].

ACCOUNT SETTINGS

User login: **jsmith**

Password: *********

Password expiration: **Mon Oct 22 2012 11:14:37 GMT-0500 (Central Daylight Time)**

UID: *

GID: *

Login shell:

Home directory:

SSH public keys: **19:DD:37:6C:7C:7E:70:6A:E7:53:97:D2:F2:45:D4:D8 jsmith@example.com (ssh-rsa)** [Show/Set key](#) [Delete](#)

[Add](#)

ACTIONS

[Reset Password](#)

Figure 9.1. Saved Public Key

After uploading the user keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing user keys. This is covered in the [Red Hat Enterprise Linux Deployment Guide](#).

9.3.3. Uploading User SSH Keys Through the Command Line

The `--sshpubkey` option uploads the 64 bit-encoded public key to the user entry. For example:

```
[jsmith@server ~]$ ipa user-mod jsmith --sshpubkey="ssh-rsa 12345abcde=
ipaclient.example.com"
```

With a real key, the key is longer and usually ends with an equals sign (=).

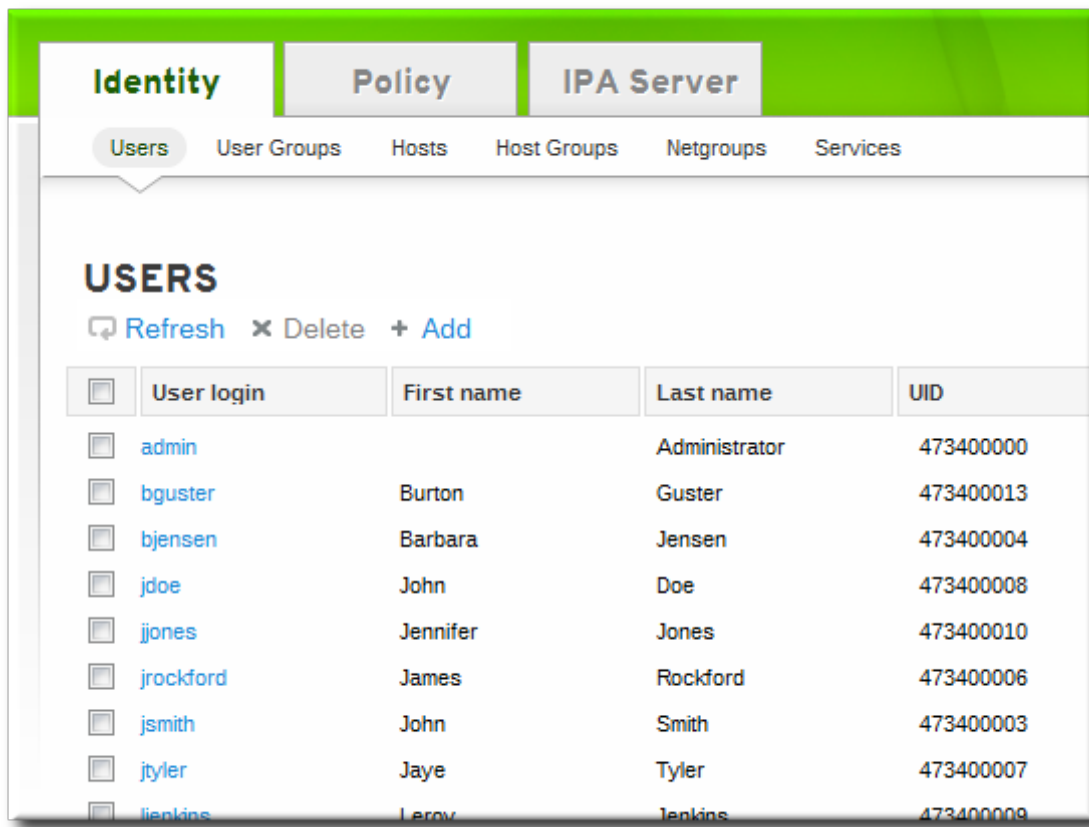
To upload multiple keys, pass a comma-separated list of keys with a single `--sshpubkey` option:

```
--sshpubkey="12345abcde==, key2==, key3=="
```

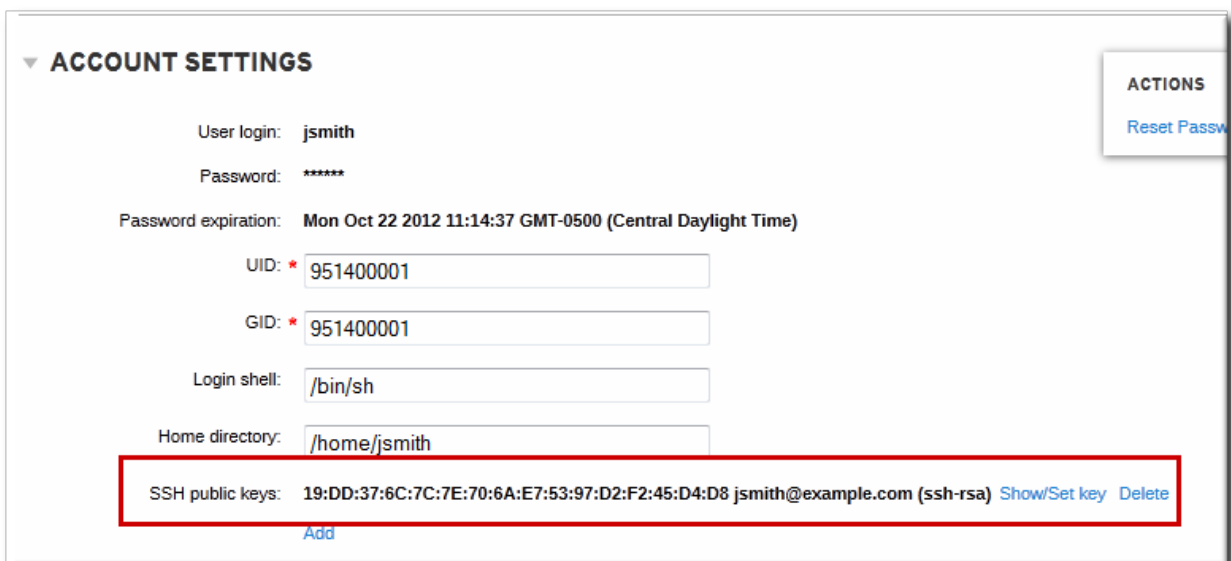
After uploading the user keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing user keys. This is covered in the [Red Hat Enterprise Linux Deployment Guide](#).

9.3.4. Deleting User Keys

1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user to edit.



3. Open the **Account Settings** area of the **Settings** tab.
4. Click the **Delete** link by the fingerprint of the key to remove.



5. Click the **Update** link at the top of the user's page to save the changes.

The command-line tools can be used to remove all keys. This is done by running **ipa user-mod** with the **--sshpubkey=** set to a blank value; this removes *all* public keys for the user. For example:

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa user-mod --sshpubkey= jsmith
```

9.4. Changing Passwords

Password policies ([Chapter 18, Policy: Defining Password Policies](#)) and minimal access restrictions can be applied to a password change operation:

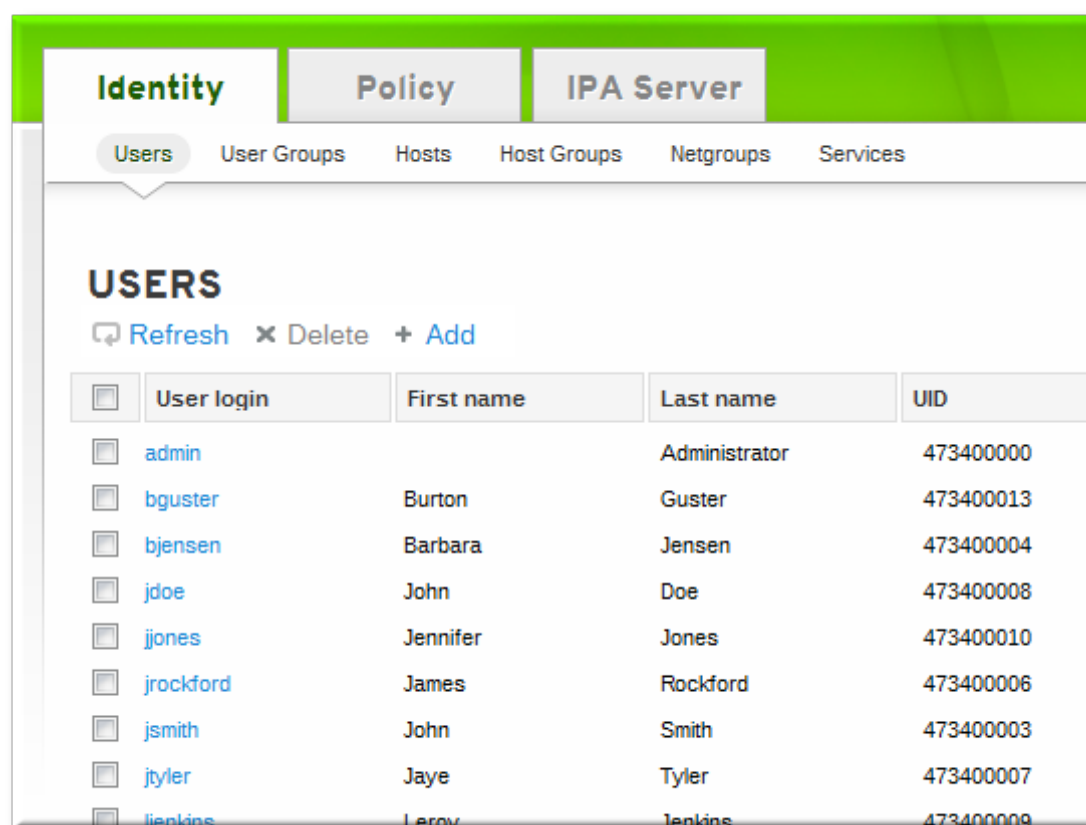
- Regular, non-administrative users can change only their personal passwords, and all passwords are constrained by the IdM password policies.

This allows administrators to create intro passwords or to reset passwords easily, while still keeping the final password confidential. Since any password sent by an administrator to the user is temporary, there is little security risk.

- Changing a password as the IdM admin user overrides any IdM password policies, but the password expires immediately. This requires the user to change the password at the next login. Similarly, any user who has password change rights can change a password and no password policies are applied, but the other user must reset the password at the next log in.
- Changing a password as the LDAP Directory Manager user, *using LDAP tools*, overrides any IdM password policies.

9.4.1. From the Web UI

1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user for whom to reset the password. All users can change their own password; only administrators or users with delegated permissions can change other user's passwords.



3. Scroll to the **Account Settings** area.
4. Click the **Reset Password** link.

► **IDENTITY SETTINGS**

▼ **ACCOUNT SETTINGS**

Status: **Enabled:** [Click to Disable](#)

User login: **jsmith**

Password: [Reset Password](#)

UID: *

GID: *

Login shell:

Home directory: *

► **PASSWORD POLICY**

5. In the pop-up box, enter and confirm the new password.

Reset Password [X]

New Password:

Verify Password:

9.4.2. From the Command Line

Changing a password — your own or another user's — is done using the **user-mod** command, as with other user account changes.

```
[bjensen@ipaserver ~]$ kinit admin
[bjensen@ipaserver ~]$ ipa user-mod jsmith --password
```

9.5. Activating and Deactivating User Accounts

User accounts can be *deactivated*. A deactivated user cannot log into IdM or its related services (like Kerberos) and he cannot perform any tasks. However, the user account still exists within Identity Management and all of the associated information remains unchanged.

**NOTE**

Any existing connections remain valid until the Kerberos TGT and other tickets expire. Once the ticket expires, the user cannot renew the ticket.

9.5.1. From the Web UI

1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user for whom to deactivate or activate.

The screenshot shows the Red Hat Identity Management Web UI. The top navigation bar has three tabs: **Identity**, **Policy**, and **IPA Server**. Under the **Identity** tab, there are subtabs: **Users**, **User Groups**, **Hosts**, **Host Groups**, **Netgroups**, and **Services**. The **Users** subtab is selected, displaying a table of users. Above the table are links for **Refresh**, **Delete**, and **Add**. The table has columns for **User login**, **First name**, **Last name**, and **UID**. Each row has a checkbox in the **User login** column.

☐	User login	First name	Last name	UID
☐	admin		Administrator	473400000
☐	bguster	Burton	Guster	473400013
☐	bjensen	Barbara	Jensen	473400004
☐	jdoe	John	Doe	473400008
☐	jjones	Jennifer	Jones	473400010
☐	jrockford	James	Rockford	473400006
☐	jsmith	John	Smith	473400003
☐	jtyler	Jaye	Tyler	473400007
☐	ljenkins	Leroy	Jenkins	473400009

3. Scroll to the **Account Settings** area.
4. Click the **Deactivate** link.

► **IDENTITY SETTINGS**

▼ **ACCOUNT SETTINGS**

Status: **Enabled:** [Click to Disable](#)

User login: **jsmith**

Password: [Reset Password](#)

UID: *

GID: *

Login shell:

Home directory: *

► **PASSWORD POLICY**

5. Click the **Update** link at the top of the page.

9.5.2. From the Command Line

Users are activated and disabled using **user-enable** and **user-disable** commands. All that is required is the user login. For example:

```
$ ipa user-disable jsmith
```

9.6. Unlocking User Accounts After Password Failures

If a user attempts to log in and uses the wrong password a certain number of times, then that user account is locked. The exact number of failed attempts that locks an account and the duration of the lockout is defined as part of the password policy ([Section 18.6, “Setting Account Lockout Policies”](#)).

A password policy can implicitly define a reset period, where the account unlocks naturally after a certain amount of time lapses. However, if the duration is fairly long or if the deployment requires stronger security checks before unlocking an account, then an administrator can unlock an account manually.

An account is unlocked using the **user-unlock** command. For example:

```
[bjensen@ipaserver ~]$ kinit admin
[bjensen@ipaserver ~]$ ipa user-unlock jsmith
```

9.7. Managing User Private Groups

On Red Hat Enterprise Linux systems, every time a user is created, a corresponding, secret user group is automatically created with that new user as its only member. This is a *user private group*. Using user private groups makes it simpler and safer to manage file and directory permissions because **umask** defaults only have to restrict user access, not group access.

When a new user is created in the IdM domain, it is also created with a corresponding private group, following the Red Hat Enterprise Linux convention. For most environments, this is an acceptable default behavior, but there may be certain users or types of users which do not require a private group or the environment may already have those GIDs ^[3] assigned to NIS groups or other system groups.

9.7.1. Disabling Private Groups for a Specific User

Private group creation can be disabled when the user is created by using the **--noprivate** option.

```
[jsmith@server ~]$ ipa user-add jsmith --first=John --last=Smith --noprivate
```

9.7.2. Disabling Private Groups Globally

User private groups are managed through the Managed Entries Plug-in in 389 Directory Server. This plug-in can be disabled, which effectively disables private group creation for all new users.

This is done using the **ipa-managed-entries** command.

1. Use the **ipa-managed-entries** command to list possible Managed Entries Plug-in definitions. By default, there are two, one for new users (UPG) and one for netgroups (NGP).

```
[root@ipaserver ~]# ipa-managed-entries --list -p DMpassword
Available Managed Entry Definitions:
UPG Definition
NGP Definition
```

2. Disable the desired Managed Entries Plug-in instance. For example:

```
[root@ipaserver ~]# ipa-managed-entries -e "UPG Definition" -p DMpassword
disable
Disabling Plugin
```

3. Restart the 389 Directory Server to load the new plug-in configuration.

```
[root@ipaserver ~]# service dirsrv restart
```

Managed Entries Plug-in instances can be re-enabled with the **enable** option.

9.8. Managing Unique UID and GID Number Assignments

An IdM server must generate random UID and GID values and simultaneously ensure that replicas never generate the same UID or GID value. The need for unique UID and GID numbers might even cross IdM domains, if a single organization has multiple disparate domains.

9.8.1. About ID Number Ranges

The UID and GID numbers are divided into *ranges*. By keeping separate numeric ranges for individual servers and replicas, the chances are minimal that any numbers issued by one server or replica will duplicate those from another. Ranges are updated and shared intelligently between servers and replicas through the Dynamic Numeric Assignment (DNA) Plug-in, as part of the backend 389 Directory Server instance for the domain. The same range is used for user IDs (**uidNumber**) and group IDs (**gidNumber**). A user and a group may have the same ID, but since the ID is set in different attributes, there is no conflict. Using the same ID number for both a user and a group also allows an administrator to configure user private groups, where a unique system group is created for each user and the ID number is the same for both the user and the group.

When a user is created interactively or without specifying a UID or GID number, then the user account is created with an ID number that is next available in the server or replica range. This means that a user always has a unique number for its UID number and, if configured, for its private group.



IMPORTANT

If a number is *manually* assigned to a user entry, the server does not validate that the **uidNumber** is unique. It will allow duplicate IDs; this is expected (though discouraged) behavior for POSIX entries. The same is true for group entries: a duplicate **gidNumber** can be manually assigned to the entry.

If two entries are assigned the same ID number, only the first entry is returned in a search for that ID number. However, both entries will be returned in searches for other attributes or with **ipa user-find --all**.

9.8.2. About ID Range Assignments During Installation

The IdM administrator can initially define a range during server installation, using the **--idstart** and **--idmax** options with **ipa-server-install**. These options are not required, so the setup script can assign random ranges during installation.

If no range is set manually when the first IdM server is installed, a range of 200,000 IDs is randomly selected. There are 10,000 possible ranges. Selecting a random range from that number provides a high probability of non-conflicting IDs if two separate IdM domains are ever merged in the future.

With a single IdM server, IDs are assigned to entries in order through the range. With replicas, the initial server ID range is split and distributed.

When a replica is installed, it is configured with an invalid range. It also has a directory entry (that is shared among replicas) that instructs the replica where it can request a valid range. When the replica starts, or as its current range is depleted so that less than 100 IDs are available, it can contact one of the available servers for a new range allotment. A special extended operation splits the range in two, so that the original server and the replica each have half of the available range.

9.8.3. A Note on Bad ID Ranges

It is possible for an administrator to define an ID number range — which means that it is possible for an administrator to define a *bad* range.

Red Hat Enterprise Linux reserves all UID/GID numbers below 500 for system use. SSSD treats all UID/GID numbers below 1000 as local system accounts. If an administrator sets the ID range to start at 500 to interact with a legacy application (for example), then user accounts assigned an ID number below 1000 will be unable to log in, because their user account is not recognized by SSSD.

9.8.4. Adding New Ranges

If the range for the entire domain is close to depletion, a new range can be manually selected and assigned to one of the master servers. All replicas then request ID ranges from the master as necessary.

The changes to the range are done by editing the 389 Directory Server configuration to change the DNA Plug-in instance. The range is defined in the **dnaNextRange** parameter. For example:

```
ldapmodify -x -D "cn=Directory Manager" -W -h server.example.com -p 389
Enter LDAP Password: *****
dn: cn=Posix IDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
changetype: modify
add: dnaNextRange
dnaNextRange: 123400000-123500000
```



NOTE

This command only adds the specified range of values; it does not check that the values in that range are actually available. This check is performed when an attempt is made to allocate those values. If a range is added that contains mostly values that were already allocated, the system will cycle through the entire range searching for unallocated values, and then the operation ultimately fails if none are available.

9.8.5. Repairing Changed UID and GID Numbers

When a user is created, the user is automatically assigned a user ID number and a group ID number.

When the user logs into an IdM system or service, SSSD on that system caches that username with the associated UID/GID numbers. The UID number is then used as the identifying key for the user. If a user with the same name but a different UID attempts to log into the system, then SSSD treats it as two different users with a name collision.

What this means is that SSSD does not recognize UID number changes. It interprets it as a different and new user, not an existing user with a different UID number. If an existing user changes the UID number, that user is prevented from logging into SSSD and associated services and domains. This also has an impact on any client applications which use SSSD for identity information; the user with the conflict will not be found or accessible to those applications.



Important

UID/GID changes are not supported in Identity Management or in SSSD.

If a user for some reason has a changed UID/GID number, then the SSSD cache must be cleared for that user before that user can log in again. For example:

```
[root@server ~]# sss_cache -u jsmith
```

9.9. Managing User and Group Schema

When a user entry is created, it is automatically assigned certain LDAP object classes which, in turn, make available certain attributes. LDAP attributes are the way that information is stored in the directory. (This is discussed in detail in the *Directory Server Deployment Guide* and the *Directory Server Schema Reference*.)

Table 9.1. Default Identity Management User Object Classes

Description	Object Classes
IdM object classes	ipaobject
Person object classes	person organizationalperson inetorgperson inetuser posixaccount
Kerberos object classes	krbprincipalaux krbticketpolicyaux
Managed entries (template) object classes	mepOriginEntry

A number of attributes are available to user entries. Some are set manually and some are set based on defaults if a specific value is not set. There is also an option to add any attributes available in the object classes in [Table 9.1, “Default Identity Management User Object Classes”](#), even if there is not a UI or command-line argument for that attribute. Additionally, the values generated or used by the default attributes can be configured, as in [Section 9.9.4, “Specifying Default User and Group Settings”](#).

Table 9.2. Default Identity Management User Attributes

UI Field	Command-Line Option	Required, Optional, or Default ^[a]
User login	<i>username</i>	Required
First name	--first	Required
Last name	--last	Required
Full name	--cn	Optional
Display name	--displayname	Optional
Initials	--initials	Default
Home directory	--homedir	Default
GECOS field	--gecos	Default
Shell	--shell	Default
Kerberos principal	--principal	Default
Email address	--email	Optional
Password	--password Unlike the other options, this accepts no value. The script prompts for the new password.	Optional
User ID number ^[b]	--uid	Default
Group ID number ^[b]	--gidnumber	Default
Street address	--street	Optional
City	--city	Optional
State/Province	--state	Optional
Zip code	--postalcode	Optional
Telephone number	--phone	Optional
Mobile telephone number	--mobile	Optional
Pager number	--pager	Optional
Fax number	--fax	Optional
Organizational unit	--orgunit	Optional
Job title	--title	Optional
Manager	--manager	Optional
Car license	--carlicense	Optional
	--noprivat	Optional
SSH Keys	--sshpubkey	Optional
Additional attributes	--addattr	Optional

^[a] Required attributes must be set for every entry. Optional attributes may be set, while default attributes are automatically added with a pre-defined value unless a specific value is given.

^[b] When a user is created without specifying a UID number, then the user account is automatically assigned an ID number that is next available in the server or replica range. (Number ranges are described more in [Section 9.8, "Managing Unique UID and GID Number Assignments"](#).) This means that a user always has a unique number for its UID number and, if configured, for its private group.

If a number is *manually* assigned to a user entry, the server does not validate that the **uidNumber** is unique. It will allow duplicate IDs; this is expected (though discouraged) behavior for POSIX entries.

If two entries are assigned the same ID number, only the first entry is returned in a search for that ID number. However, both entries will be returned in searches for other attributes or with **ipa user-find --all**.

9.9.1. About Changing the Default User and Group Schema

It is possible to add or change the object classes and attributes used for user and group entries ([Section 9.9, “Managing User and Group Schema”](#)).

The IdM configuration provides some validation when object classes are changed:

- » All of the object classes and their specified attributes must be known to the LDAP server.
- » All default attributes that are configured for the entry must be supported by the configured object classes.

There are limits to the IdM schema validation, however. Most important, the IdM server does not check that the defined user or group object classes contain all of the required object classes for IdM entries. For example, all IdM entries require the **ipaobject** object class. However, when the user or group schema is changed, the server does not check to make sure that this object class is included; if the object class is accidentally deleted, then future entry add operations will fail.

Also, all object class changes are atomic, not incremental. The entire list of default object classes has to be defined every time there is a change. For example, a company may create a custom object class to store employee information like birthdays and employment start dates. The administrator cannot simply add the custom object class to the list; he must set the entire list of current default object classes *plus* the new object class. The *existing* default object classes must always be included when the configuration is updated. Otherwise, the current settings will be overwritten, which causes serious performance problems.

9.9.2. Applying Custom Object Classes to New User Entries

User and group accounts are created with a pre-defined set of LDAP object classes applied to the entry. Any attributes which belong to the object class can be added to the user entry.

While the standard and IdM-specific LDAP object classes will cover most deployment scenarios, administrators may have custom object classes with custom attributes which should be applied to user entries.

9.9.2.1. From the Web UI

1. Add all of the custom schema elements to the 389 Directory Server instance used by Identity Management. Adding schema elements is described in [the schema chapter of the Directory Server Administrator's Guide](#).
2. Open the **IPA Server** tab.
3. Select the **Configuration** subtab.
4. Scroll to the **User Options** area.

CONFIGURATION

 Reset  Update

▼ SEARCH OPTIONS

Search size limit:

100

Search time limit:

2

▼ USER OPTIONS

User search fields:

uid,givenname,sn,telephonenumber,ou

Default e-mail domain for new users

example.com

Default users group:

ipausers ▼

Home directory base:

/home

Max. username length:

32

Password Expiration Notification (days)

4

Enable migration mode:

☐

Default user objectclasses:

top

Delete

- At the bottom of the users area, click the **Add** link to add a new field for another object class.



IMPORTANT

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

Default user objectclasses:

top	Delete
person	Delete
organizationalperson	Delete
inetorgperson	Delete
inetuser	Delete
posixaccount	Delete
krbprincipalaux	Delete
krbticketpolicyaux	Delete
ipaobject	Delete
Add	

6. When the changes are complete, click the **Update** link at the top of the **Configuration** page.

9.9.2.2. From the Command Line

1. Add all of the custom schema elements to the 389 Directory Server instance used by Identity Management. Adding schema elements is described in [the schema chapter of the Directory Server Administrator's Guide](#).
2. Add the new object class to the list of object classes added to entries. The option for user object classes is **--userobjectclasses**.



IMPORTANT

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

For example:

```
$ ipa config-mod --
userobjectclasses=top,person,organizationalperson,inetorgperson,inetuser,po
saccount, krbprincipalaux,krbticketpolicyaux,ipaobject,employeeinfo
```

9.9.3. Applying Custom Object Classes to New Group Entries

As with user entries, administrators may have custom object classes with custom attributes which should be applied to group entries. These can be added automatically by adding the object classes to the IdM server configuration.

9.9.3.1. From the Web UI

1. Add all of the custom schema elements to the 389 Directory Server instance used by Identity Management. Adding schema elements is described in [the schema chapter of the Directory Server Administrator's Guide](#).
2. Open the **IPA Server** tab.
3. Select the **Configuration** subtab.
4. Scroll to the **Group Options** area.

The screenshot shows the IPA Server Configuration page. At the top, there are tabs for Identity, Policy, and IPA Server. Under the IPA Server tab, there are subtabs: Role Based Access Control, Self Service Permissions, Delegations, Entitlements, and Configuration. The Configuration subtab is active. Below the subtabs, there is a 'CONFIGURATION' header with links for Refresh, Reset, and Update. The main content area is divided into sections: SEARCH OPTIONS, USER OPTIONS, and GROUP OPTIONS. The GROUP OPTIONS section is expanded, showing a 'Group search fields' input with the value 'cn,description'. Below this, there is a 'Default group objectclasses' section with a list of object classes: top, groupofnames, nestedgroup, ipausergroup, and ipaobject. Each object class has a 'Delete' link next to it. At the bottom of the list is an 'Add' link.

5. Click the **Add** link to add a new field for another object class.



IMPORTANT

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

6. When the changes are complete, click the **Update** link at the top of the **Configuration** page.

9.9.3.2. From the Command Line

1. Add all of the custom schema elements to the 389 Directory Server instance used by Identity Management. Adding schema elements is described in [the schema chapter of the Directory Server Administrator's Guide](#).
2. Add the new object class to the list of object classes added to entries. The option for group object classes is **--groupobjectclasses**.



IMPORTANT

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

For example:

```
$ ipa config-mod --  
groupobjectclasses=top,groupofnames,nestedgroup,ipausergroup,ipaobject,empl  
oyeegroup
```

9.9.4. Specifying Default User and Group Settings

Identity Management uses a template when it creates new entries.

For users, the template is very specific. Identity Management uses default values for several core attributes for IdM user accounts. These defaults can define actual values for user account attributes (such as the home directory location) or it can define the format of attribute values, such as the username length. These settings also define the object classes assigned to users.

For groups, the template only defines the assigned object classes.

These default definitions are all contained in a single configuration entry for the IdM server, **cn=ipaconfig,cn=etc,dc=example,dc=com**.

The configuration can be changed using the **ipa config-mod** command.

Table 9.3. Default User Parameters

Field	Command-Line Option	Descriptions
Maximum username length	--maxusername	Sets the maximum number of characters for usernames. The default value is eight.
Root for home directories	--homedirectory	Sets the default directory to use for user home directories. The default value is /home .
Default shell	--defaultshell	Sets the default shell to use for users. The default value is /bin/sh .
Default user group	--defaultgroup	Sets the default group to which all newly created accounts are added. The default value is ipausers , which is automatically created during the IdM server installation process.
Default e-mail domain	--emaildomain	Sets the email domain to use to create email addresses based on the new accounts. The default is the IdM server domain.
Search time limit	--searchtimelimit	Sets the maximum amount of time, in seconds, to spend on a search before the server returns results.
Search size limit	--searchrecordslimit	Sets the maximum number of records to return in a search.
User search fields	--usersearch	Sets the fields in a user entry that can be used as a search string. Any attribute listed has an index kept for that attribute, so setting too many attributes could affect server performance.
Group search fields	--groupsearch	Sets the fields in a group entry that can be used as a search string.
Certificate subject base		Sets the base DN to use when creating subject DN's for client certificates. This is configured when the server is set up.
Default user object classes	--userobjectclasses	Sets a list of object classes that are used to create IdM user accounts.
Default group object classes	--groupobjectclasses	Sets a list of object classes that are used to create IdM group accounts.
Password expiration notification	--pwdexpnotify	Sets how long, in days, before a password expires for the server to send a notification.

Password plug-in features

Sets the format of passwords that are allowed for users.

9.9.4.1. Viewing Settings from the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. The complete configuration entry is shown in three sections, one for all search limits, one for user templates, and one for group templates.

The screenshot displays the web interface for configuring the IPA Server. At the top, there are three main tabs: 'Identity', 'Policy', and 'IPA Server'. The 'IPA Server' tab is currently selected. Below these, there are several sub-tabs: 'Role Based Access Control', 'Self Service Permissions', 'Delegations', 'Entitlements', and 'Configuration'. The 'Configuration' sub-tab is active. The main content area is titled 'CONFIGURATION' and includes three buttons: 'Refresh', 'Reset', and 'Update'. Below this, there are three expandable sections: 'SEARCH OPTIONS' (expanded), 'USER OPTIONS', and 'GROUP OPTIONS'. The 'SEARCH OPTIONS' section contains two input fields: 'Search size limit' with a value of 100, and 'Search time limit' with a value of 2.

9.9.4.2. Viewing Settings from the Command Line

The **config-show** command shows the current configuration which applies to all new user accounts. By default, only the most common attributes are displayed; use the **--all** option to show the complete configuration.

```
# ipa config-show --all

dn: cn=ipaconfig,cn=etc,dc=example,dc=com
Max. username length: 32
Home directory base: /home
Default shell: /bin/sh
Default users group: ipausers
Default e-mail domain for new users: example.com
Search time limit: 2
Search size limit: 100
User search fields: uid,givenname,sn,telephonenumber,ou,title
Group search fields: cn,description
Enable migration mode: FALSE
Certificate Subject base: O=EXAMPLE.COM
Default group objectclasses: top, groupofnames, nestedgroup, ipausergroup,
ipaobject
Default user objectclasses: top, person, organizationalperson, inetorgperson,
inetuser, posixaccount,
krbprincipalaux, krbticketpolicyaux, ipaobject
Password Expiration Notification (days): 4
Password plugin features: AllowNThash
cn: ipaConfig
objectclass: nsContainer, top, ipaGuiConfig, ipaConfigObject
```

9.10. Managing User Groups

User groups are a way of centralizing control over important management tasks, particularly access control and password policies. Three groups are created during the installation, specifically for use by IdM operations:

- `ipausers`, which contains all users.
- `admins`, which contains administrative users. The initial **admin** user belongs to this group.
- `editors`, which is a special group for users working through the web UI. This group allows users to *edit* other users' entries, though without all of the rights of the admin user.

All groups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, IdM allows *nested groups*, where a group is a member of another group. In that case, all of the group members of the member group automatically belong to the parent group, as well.

Because groups are easy to create, it is possible to be very flexible in what groups to create and how they are organized. Groups can be defined around organizational divisions like departments, physical locations, or IdM or infrastructure usage guidelines for access controls.



NOTE

Some operating systems limit the number of groups that can be assigned to system users. For example, Solaris and AIX systems both limit users to 16 groups per user. This can be an issue when using nested groups, when a user may be automatically added to multiple groups.

When a group entry is created, it is automatically assigned certain LDAP object classes. (LDAP object classes and attributes are discussed in detail in the *Directory Server Deployment Guide* and the *Directory Server Schema Reference*.) For groups, only two attributes truly matter: the name and the

description.

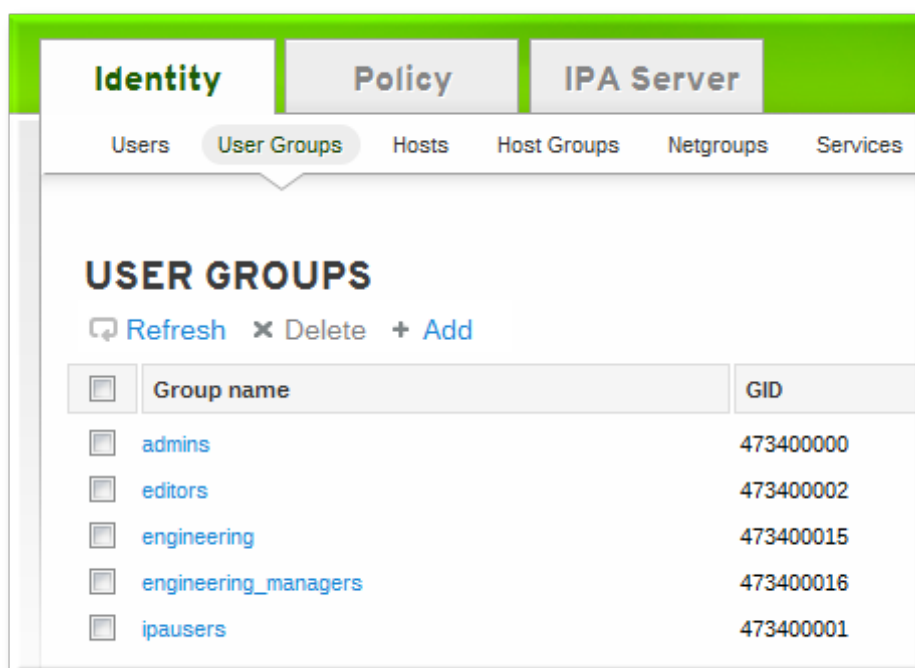
Table 9.4. Default Identity Management Group Object Classes

Description	Object Classes
IdM object classes	ipaobject
	ipausergroup
	nestedgroup
Group object classes	groupofnames
	posixgroup

9.10.1. Creating User Groups

9.10.1.1. With the Web UI

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Click the **Add** link at the top of the groups list.



3. Enter all of the information for the group.

- A unique name. This is the identifier used for the group in the IdM domain, and it cannot be changed after it is created. The name cannot contain spaces, but other separators like an underscore (`_`) are allowed.
- A text description of the group.
- Whether the group is a Posix group, which adds Linux-specific information to the entry. By default, all groups are Posix groups unless they are explicitly configured not to be. Non-Posix groups can be created for interoperability with Windows or Samba.
- Optionally, the GID number for the group. All Posix groups require a GID number, but IdM automatically assigns the GID number.

Setting a GID number is not necessary because of the risk of collisions. If a GID number is given manually, IdM will not override the specified GID number, even if it is not unique.

4. Click the **Add and Edit** button to go immediately to the member selection page.
5. Select the members, as described in [Section 9.10.2.1, “With the Web UI \(Group Page\)”](#).

9.10.1.2. With the Command Line

New groups are created using the **group-add** command. (This adds only the group; members are added separately.)

Two attributes are always required: the group name and the group description. If those attributes are not given as arguments, then the script prompts for them.

```
$ ipa group-add groupName --desc="description" [--nonposix]
```

Additionally, there is one other configuration option, **--nonposix**. (By default, all groups are created as POSIX groups.) To enable interoperability with Windows users and groups and programs like Samba, it is possible to create non-POSIX groups by using the **--nonposix** option. This option tells the script not to add the **posixGroup** object class to the entry.

For example:

```
$ ipa group-add examplegroup --desc="for examples" --nonposix

-----
Added group "examplegroup"
-----
  Group name: examplegroup
  Description: for examples
  GID: 855800010
```

When no arguments are used, the command prompts for the required group account information:

```
$ ipa group-add
Group name: engineering
Description: for engineers
-----
Added group "engineering"
-----
  Group name: engineering
  Description: for engineers
  GID: 387115842
```



IMPORTANT

When a group is created without specifying a GID number, then the group entry is assigned the ID number that is next available in the server or replica range. (Number ranges are described more in [Section 9.8, “Managing Unique UID and GID Number Assignments”](#).) This means that a group always has a unique number for its GID number.

If a number is *manually* assigned to a group entry, the server does not validate that the **gidNumber** is unique. It will allow duplicate IDs; this is expected (though discouraged) behavior for POSIX entries.

If two entries are assigned the same ID number, only the first entry is returned in a search for that ID number. However, both entries will be returned in searches for other attributes or with **ipa group-find --all**.



NOTE

You cannot edit the group name. The group name is the primary key, so changing it is the equivalent of deleting the group and creating a new one.

9.10.2. Adding Group Members

9.10.2.1. With the Web UI (Group Page)

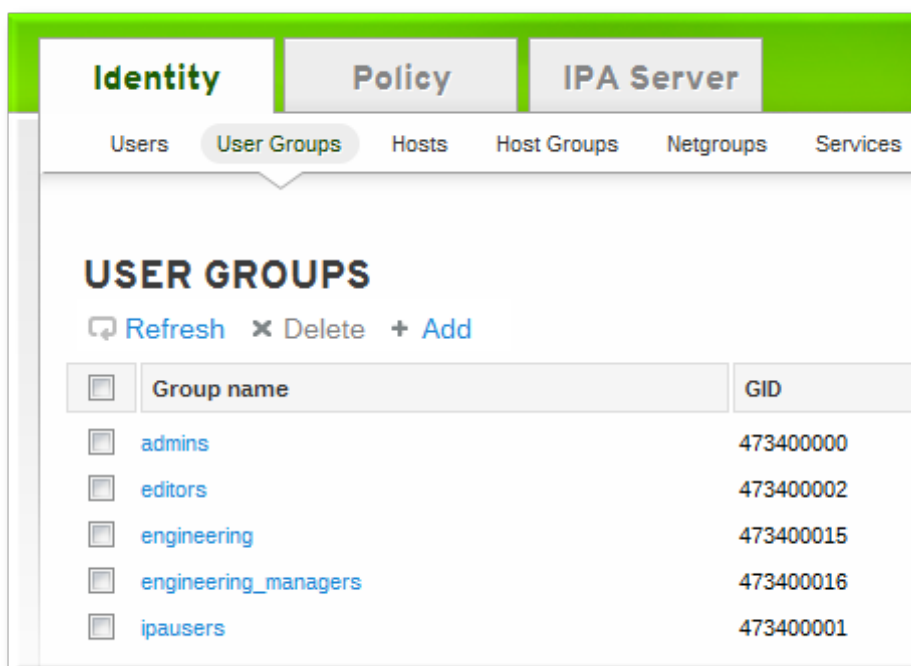
NOTE

This procedure adds a user to a group. User groups can contain other user groups as their members. These are *nested* groups.

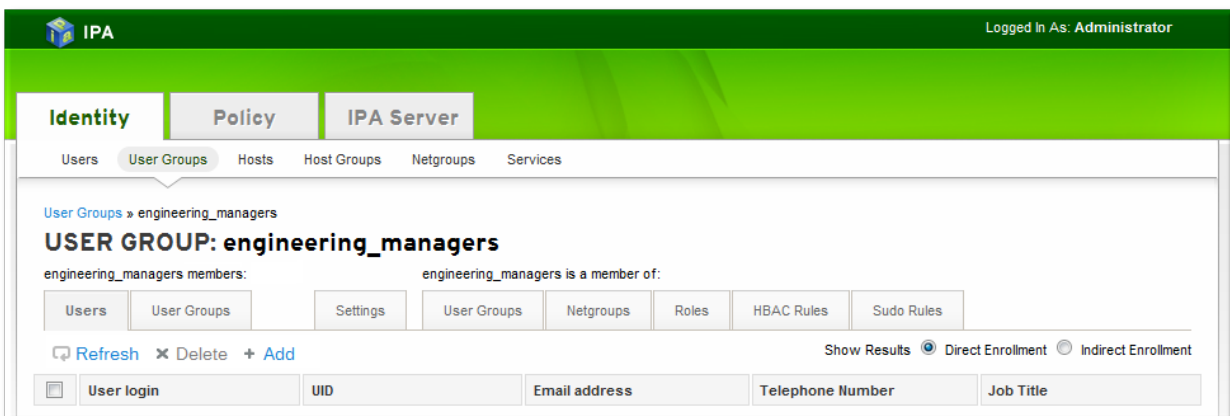
It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

When creating nested groups, be careful not to create *recursive* groups. For example, if GroupA is a member of GroupB, do not add GroupB as a member of GroupA. Recursive groups are not supported and can cause unpredictable behavior.

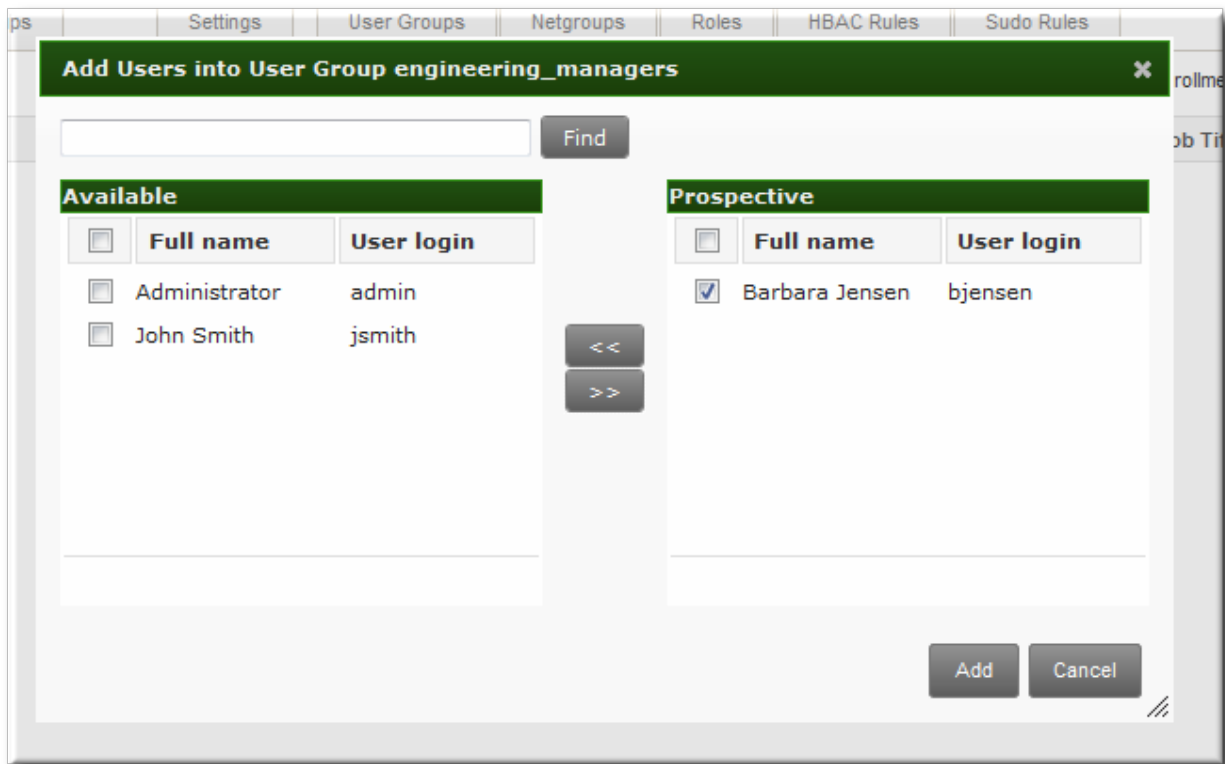
1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Click the name of the group to which to add members.



3. Click the **Add** link at the top of the task area.



4. Click the checkbox by the names of the users to add, and click the right arrows button, >>, to move the names to the selection box.



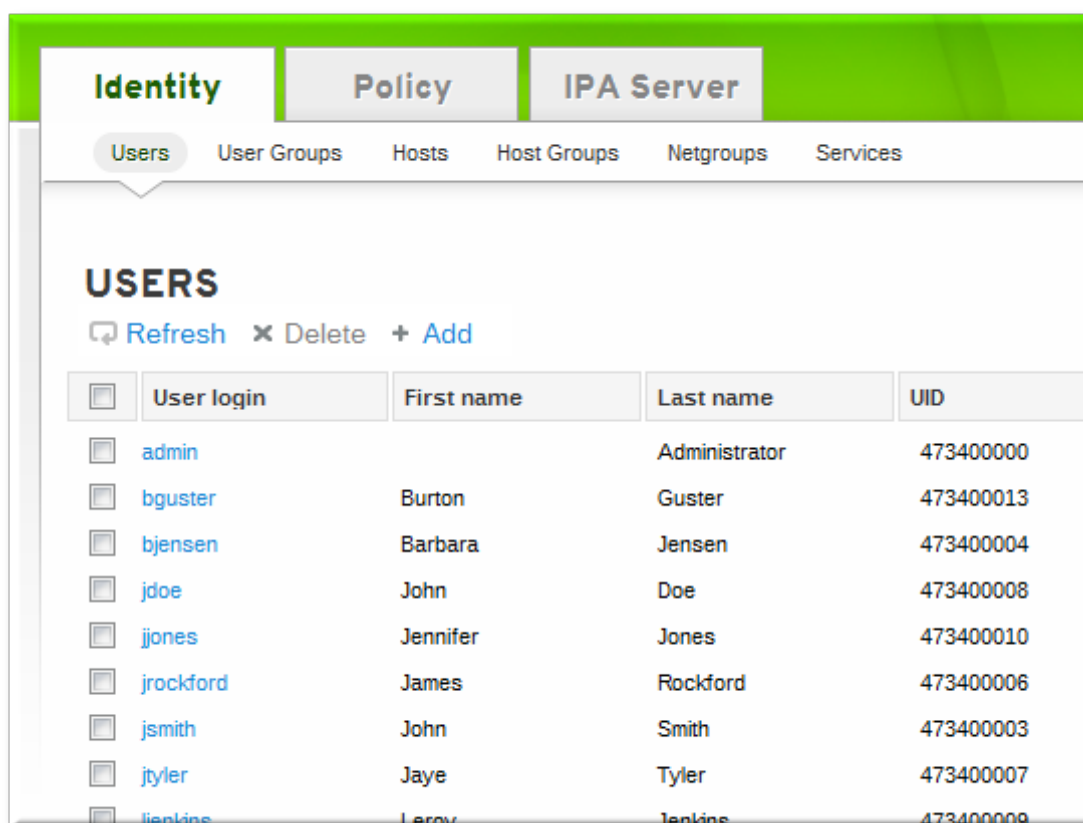
5. Click the **Add** button.

Group members can be users or other user groups. It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

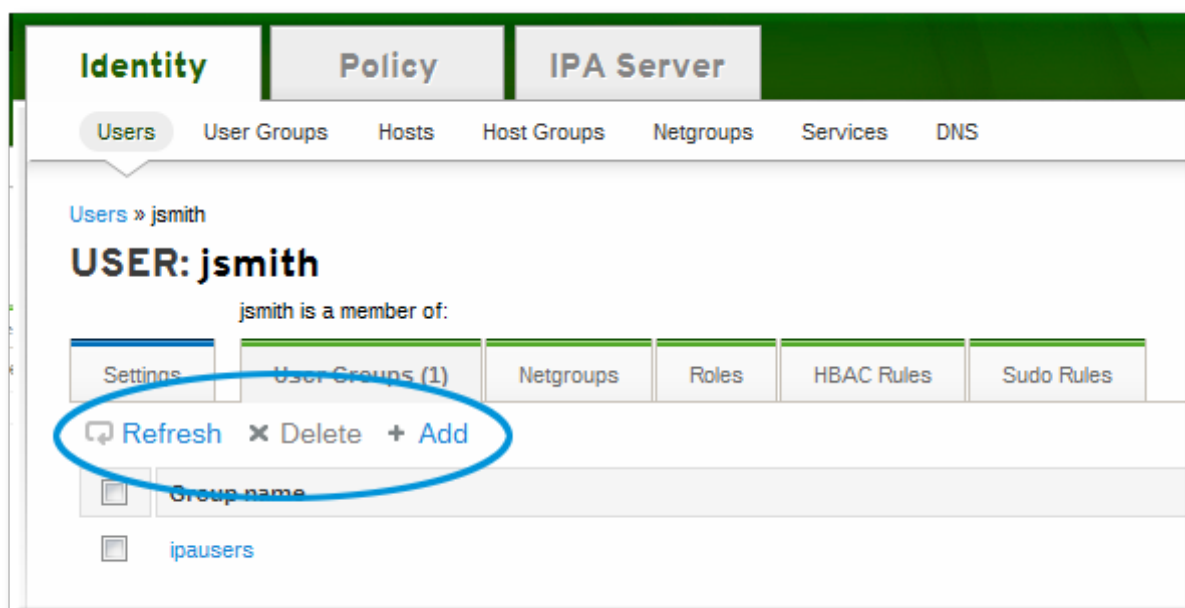
9.10.2.2. With the Web UI (User's Page)

Users can also be added to a group through the user's page.

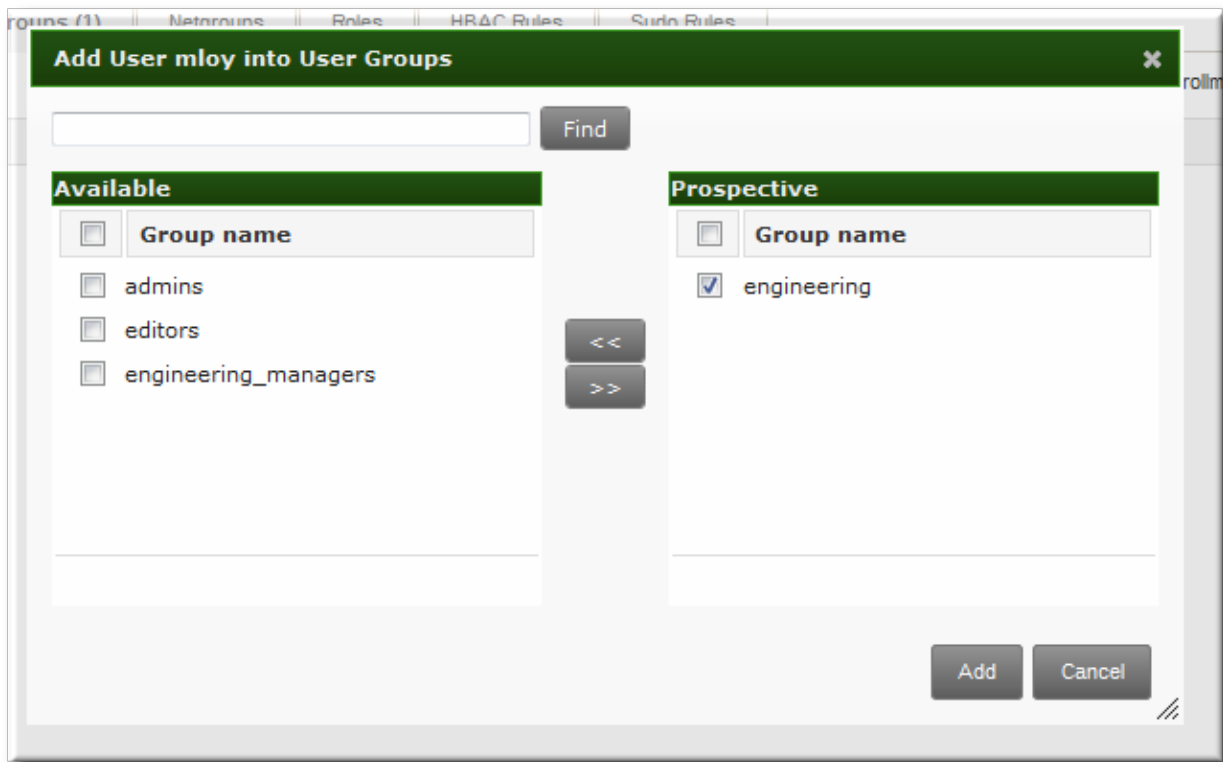
1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user to edit.



3. Open the **User Groups** tab on the user entry page.
4. Click the **Add** link at the top of the task area.



5. Click the checkbox by the names of the groups for the user to join, and click the right arrows button, **>>**, to move the groups to the selection box.



6. Click the **Add** button.

9.10.2.3. With the Command Line

Members are added to a group using the **group-add-member** command. This command can add both users as group members and other groups as group members.

The syntax of the **group-add-member** command requires only the group name and a comma-separated list of users to add:

```
$ ipa group-add-member groupName [--users=list] [--groups=list]
```

For example, this adds three users to the **engineering** group:

```
$ ipa group-add-member engineering --users=jsmith,bjensen,mreynolds
Group name: engineering
Description: for engineers
GID: 387115842
Member users: jsmith,bjensen,mreynolds
-----
Number of members added 3
-----
```

Likewise, other groups can be added as members, which creates nested groups:

```
$ ipa group-add-member engineering --groups=dev,qe1,dev2
Group name: engineering
Description: for engineers
GID: 387115842
Member groups: dev,qe1,dev2
-----
Number of members added 3
-----
```

When displaying nested groups, members are listed as members and the members of any member groups are listed as indirect members. For example:

```
$ ipa group-show examplegroup
Group name: examplegroup
Description: for examples
GID: 932000002
Member users: jsmith,bjensen,mreynolds
Member groups: californiausers
Indirect Member users: sbeckett,acalavicci
```

It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.



NOTE

When creating nested groups, be careful not to create *recursive* groups. For example, if GroupA is a member of GroupB, do not add GroupB as a member of GroupA. Recursive groups are not supported and can cause unpredictable behavior.

A group member is removed using the **group-remove-member** command.

```
$ ipa group-remove-member engineering --users=jsmith

Group name: engineering
Description: for engineers
GID: 855800009
Member users: bjensen,mreynolds
-----
Number of members removed 1
-----
```

9.10.2.4. Viewing Direct and Indirect Members of a Group

User groups can contain other user groups as members. This is called a *nested group*. This also means that a group has two types of members:

- *Direct members*, which are added explicitly to the group
- *Indirect members*, which are members of the group because they are members of another user group which is a member of the group

The IdM web UI has an easy way to view direct and indirect members of a group. The members list is filtered by member type, and this can be toggled by selecting the **Direct** and **Indirect** radio buttons at the top right corner of the members list.

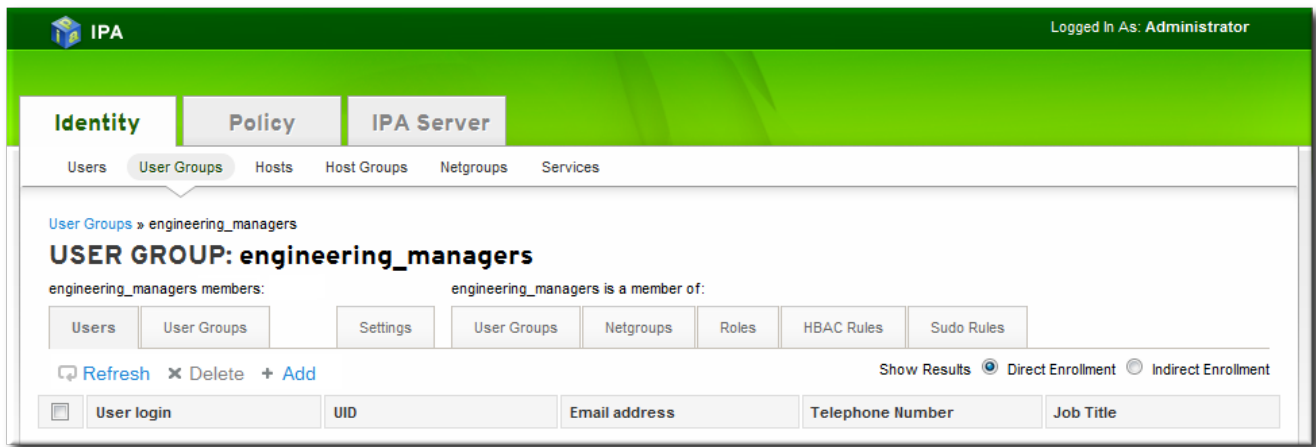


Figure 9.2. Indirect and Direct Members

Being able to track indirect members makes it easier to assign group membership properly, without duplicating membership.

9.10.3. Deleting User Groups

When a user group is deleted, only the group is removed. The user accounts of group members (including nested groups) are not affected. Additionally, any access control delegations that apply to that group are removed.

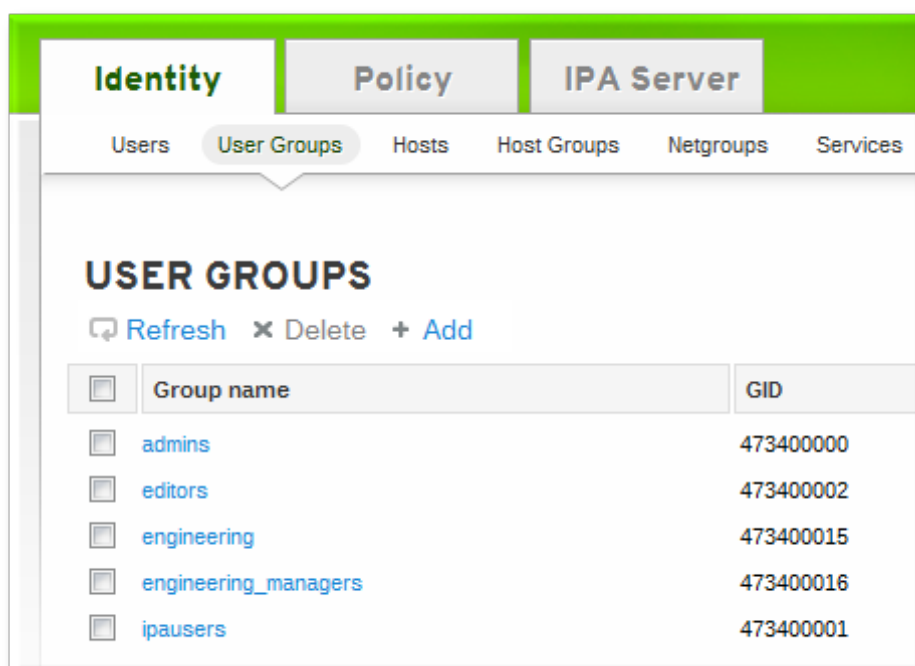


WARNING

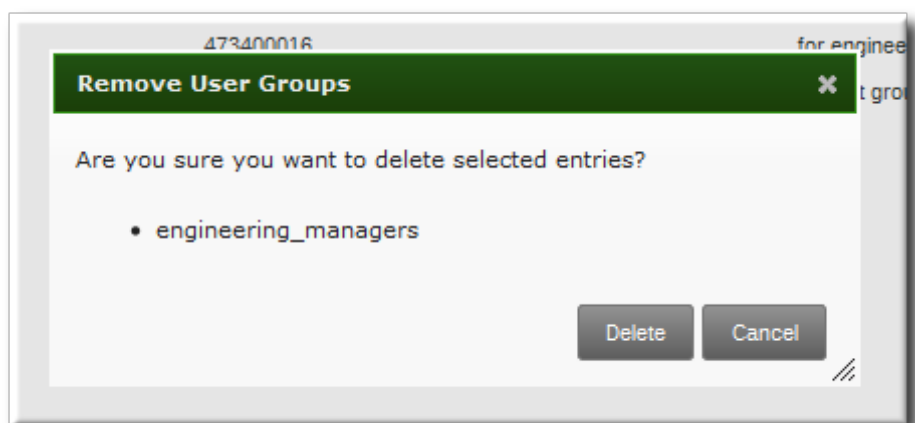
Deleting a group is immediate and permanent. If any group configuration (such as delegations) is required, it must be assigned to another group or a new group created.

9.10.3.1. With the Web UI

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Select the checkbox by the name of the group to delete.



3. Click the **Delete** link at the top of the task area.
4. When prompted, confirm the delete action.



9.10.3.2. With the Command Line

The **group-del** command to deletes the specified group. For example:

```
$ ipa group-del examplegroup
```

9.11. Searching for Users and Groups

The user searches in IdM can be run against simple (full word) or partial search strings. The range of attributes that are searched is configured as part of the default IdM configuration, as in [Section 9.9.4, “Specifying Default User and Group Settings”](#).

9.11.1. Setting Search Limits

9.11.1.1. Types of Search Limits and Where They Apply

Some searches can result in a large number of entries being returned, possibly even all entries. Search limits improve overall server performance by limiting how long the server spends in a search and how

many entries are returned.

Search limits have a dual purpose to improve server performance by reducing the search load and to improve usability by returning a smaller — and therefore easier to browse — set of entries.

The IdM server has several different limits imposed on searches:

- *The search limit configuration for the IdM server.* This is a setting for the IdM server itself, which is applied to all requests sent to the server from all IdM clients, the IdM CLI tools, and the IdM web UI for normal page display.
By default, this limit is 100 entries.
- *The time limit configuration for the IdM server.* Much like the search size limit, the time limit sets a maximum amount of time that the IdM server, itself, waits for searches to run. Once it reaches that limit, the server stops the search and returns whatever entries were returned in that time.
By default, this limit is two seconds.
- *The page size limit.* Although not strictly a search limit, the page size limit does limit how many entries are returned per page. The server returns the set of entries, up to the search limit, and then randomly selects up to 20 entries per page for display. Paging results makes the results more understandable and more viewable.
This is hard-coded to 20 for all searches.
- *The LDAP search limit (--pkey option).* All searches performed in the UI, and CLI searches which use the **--pkey** option, override the search limit set in the IdM server configuration and use the search limit set in the underlying LDAP directory.
By default, this limit is 2000 entries. It can be edited by editing the 389 Directory Server configuration.

9.11.1.2. Setting IdM Search Limits

Search limits set caps on the number of records returned or the time spent searching when querying the database for user or group entries. There are two types of search limits: time limits and size (number) limits.

With the default settings, users are limited to two-second searches and no more than 100 records returned per search.



IMPORTANT

Setting search size or time limits too high can negatively affect IdM server performance.

9.11.1.2.1. With the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Search Options** area.

4. Change the search limit settings.

- *Search size limit*, the maximum number of records to return in a search.
- *Search time limit*, the maximum amount of time, in seconds, to spend on a search before the server returns results.



TIP

Setting the time limit or size limit value to -1 means that there are no limits on searches.

5. When the changes are complete, click the **Update** link at the top of the **Configuration** page.

9.11.1.2.2. With the Command Line

The search limits can be changed using the **config-mod** command.

```
$ ipa config-mod --searchtimelimit=5 --searchrecordslimit=500

Max. username length: 32
Home directory base: /home
Default shell: /bin/sh
Default users group: ipausers
Default e-mail domain for new users: example.com
Search time limit: 5
Search size limit: 50
User search fields: uid,givenname,sn,telephonenumber,ou,title
Group search fields: cn,description
Enable migration mode: FALSE
Certificate Subject base: O=EXAMPLE.COM
Password Expiration Notification (days): 4
```

**TIP**

Setting the time limit or size limit value to -1 means that there are no limits on searches.

9.11.1.3. Overriding the Search Defaults

Part of the server configuration is setting global defaults for size and time limits on searches. While these limits are always enforced in the web UI, they can be overridden with any ***-find** command run through the command line.

The **--sizelimit** and **--timelimit** options set alternative size and time limits, respectively, for that specific command run. The limits can be higher or lower, depending on the kinds of results you need.

For example, if the default time limit is 60 seconds and a search is going to take longer, the time limit can be increased to 120 seconds:

```
[jsmith@ipaserver ~]$ ipa user-find smith --timelimit=120
```

9.11.2. Setting Search Attributes

A search for users or groups does not automatically search every possible attribute for that attribute. Rather, it searches a specific subset of attributes, and that list is configurable.

When adding attributes to the user or group search fields, make sure that there is a corresponding index within the LDAP directory for that attribute. Searches are performed based on indexes. Most standard LDAP attributes have indexes, but any custom attributes must have indexes created for them. Creating indexes is described in the [indexes chapter in the Directory Server Administrator's Guide](#).

9.11.2.1. Default Attributes Checked by Searches

By default, there are six attributes that are indexed for user searches and two that are indexed for group searches. These are listed in [Table 9.5, “Default Search Attributes”](#). All search attributes are searched in a user/group search.

Table 9.5. Default Search Attributes

User Search Attributes	
First name	Last name
Login ID	Job title
Organizational unit	Phone number
Group Search Attributes	
Name	Description

The attributes which are searched in user and group searches can be changed, as described in [Section 9.11.2, “Setting Search Attributes”](#) and [Section 9.11.2.3, “Changing Group Search Attributes”](#).

9.11.2.2. Changing User Search Attributes

9.11.2.2.1. From the Web UI

1. Open the **IPA Server** tab.

2. Select the **Configuration** subtab.
3. Scroll to the **User Options** area.

CONFIGURATION

Reset Update

▼ SEARCH OPTIONS

Search size limit:

Search time limit:

▼ USER OPTIONS

User search fields:

Default e-mail domain for new users:

Default users group:

Home directory base:

Max. username length:

Password Expiration Notification (days):

Enable migration mode: ☐

Default user objectclasses:

4. Add any additional search attributes, in a comma-separated list, in the **User search fields** field.
5. When the changes are complete, click the **Update** link at the top of the **Configuration** page.

9.11.2.2.2. From the Command Line

To change the search attributes, use the `--usersearch` option to set the attributes for user searches.

```
$ ipa config-mod --usersearch=uid,givenname,sn,telephonenumber,ou,title
```



NOTE

Always give the complete list of search attributes. Whatever values are passed with the configuration argument overwrite the previous settings.

9.11.2.3. Changing Group Search Attributes

A search for users or groups does not automatically search every possible attribute for that attribute.

Rather, it searches a specific subset of attributes, and that list is configurable.

When adding attributes to the user or group search fields, make sure that there is a corresponding index within the LDAP directory for that attribute. Searches are performed based on indexes. Most standard LDAP attributes have indexes, but any custom attributes must have indexes created for them. Creating indexes is described in the [indexes chapter in the Directory Server Administrator's Guide](#).

9.11.2.3.1. From the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Group Options** area.

The screenshot shows the IPA Server Configuration web interface. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Under 'IPA Server', there are subtabs: 'Role Based Access Control', 'Self Service Permissions', 'Delegations', 'Entitlements', and 'Configuration'. The 'Configuration' subtab is active. Below the subtabs, the main heading is 'CONFIGURATION'. There are three links: 'Refresh', 'Reset', and 'Update'. Below this, there are three expandable sections: 'SEARCH OPTIONS', 'USER OPTIONS', and 'GROUP OPTIONS'. The 'GROUP OPTIONS' section is expanded, showing 'Group search fields' with a text input containing 'cn,description'. Below that, 'Default group objectclasses' are listed in a table-like structure with 'Delete' links for each: 'top', 'groupofnames', 'nestedgroup', 'ipausergroup', and 'ipaobject'. At the bottom of this section is an 'Add' link.

4. Add any additional search attributes, in a comma-separated list, in the **Group search fields** field.
5. When the changes are complete, click the **Update** link at the top of the **Configuration** page.

9.11.2.3.2. From the Command Line

To change the search attributes, use the **--groupsearch** options to set the attributes for group searches.

```
$ ipa config-mod --groupsearch=cn,description
```



NOTE

Always give the complete list of search attributes. Whatever values are passed with the configuration argument overwrite the previous settings.

9.11.2.4. Limits on Attributes Returned in Search Results

Searches can be performed on attributes that are not displayed in the UI. This means that entries can be returned in a search that do not appear to match the given filter. This is especially common if the search information is very short, which increases the likelihood of a match.

[2] The key type is determined automatically from the key itself, if it is not included in the uploaded key.

[3] See [Section 9.8](#), “Managing Unique UID and GID Number Assignments” for information on changing GID/UID assignment ranges.

Chapter 10. Identity: Managing Hosts

Both DNS and Kerberos are configured as part of the initial client configuration. This is required because these are the two services that bring the machine within the IdM domain and allow it to identify the IdM server it will connect with. After the initial configuration, IdM has tools to manage both of these services in response to changes in the domain services, changes to the IT environment, or changes on the machines themselves which affect Kerberos, certificate, and DNS services, like changing the client hostname.

This chapter describes how to manage identity services that relate directly to the client machine:

- DNS entries and settings
- Machine authentication
- Hostname changes (which affect domain services)

10.1. About Hosts, Services, and Machine Identity and Authentication

The basic function of an enrollment process is to create a *host* entry for the client machine in the IdM directory. This host entry is used to establish relationships between other hosts and even services within the domain. These relationships are part of *delegating* authorization and control to hosts within the domain.

A host entry contains all of the information about the client within IdM:

- Service entries associated with the host
- The host and service principal
- Access control rules
- Machine information, such as its physical location and operating system

Some services that run on a host can also belong to the IdM domain. Any service that can store a Kerberos principal or an SSL certificate (or both) can be configured as an IdM service. Adding a service to the IdM domain allows the service to request an SSL certificate or keytab from the domain. (Only the public key for the certificate is stored in the service record. The private key is local to the service.)

An IdM domain establishes a commonality between machines, with common identity information, common policies, and shared services. Any machine which belongs to a domain functions as a client of the domain, which means it uses the services that the domain provides. An IdM domain (as described in [Section 1.2, “Bringing Linux Services Together”](#)) provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

Machines are treated as another identity that is managed by IdM. Clients use DNS to identify IdM servers, services, and domain members — which, like user identities are stored in the 389 Directory Server instance for the IdM server. Like users, machines can be authenticated to the domain using Kerberos or certificates to verify the machine's identity.

From the machine perspective, there are several tasks that can be performed that access these domain services:

- Joining the DNS domain (*machine enrollment*)

- Managing DNS entries and zones
- Managing machine authentication

Authentication in IdM includes machines as well as users. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests. IdM supports three different approaches to machine authentication:

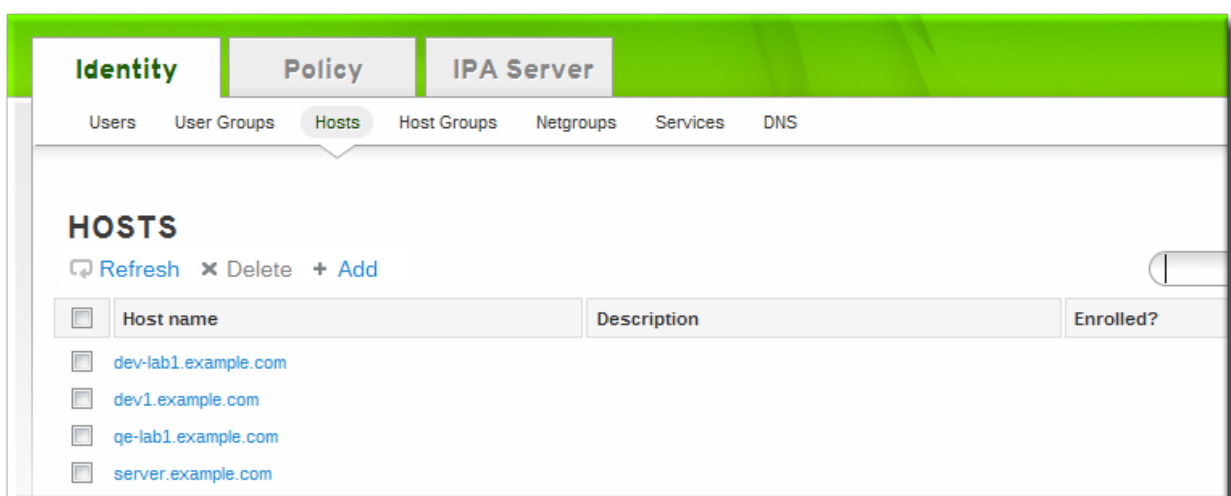
- SSH keys. The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in Identity Management. This is described in [Section 10.4, “Managing Public SSH Keys for Hosts”](#) and the [Red Hat Enterprise Linux Deployment Guide](#).
- Key tables (or *keytabs*, a symmetric key resembling to some extent a user password) and machine certificates. Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. Initially granting a Kerberos ticket, renewing the Kerberos credentials, and even destroying the Kerberos session are all handled by the IdM services. Managing Kerberos is covered in [Chapter 19, Policy: Managing the Kerberos Domain](#).
- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the IdM server's certificate authority and then stored in IdM's Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called *certmonger*, which is described in [Appendix B, Working with certmonger](#).

10.2. Adding Host Entries

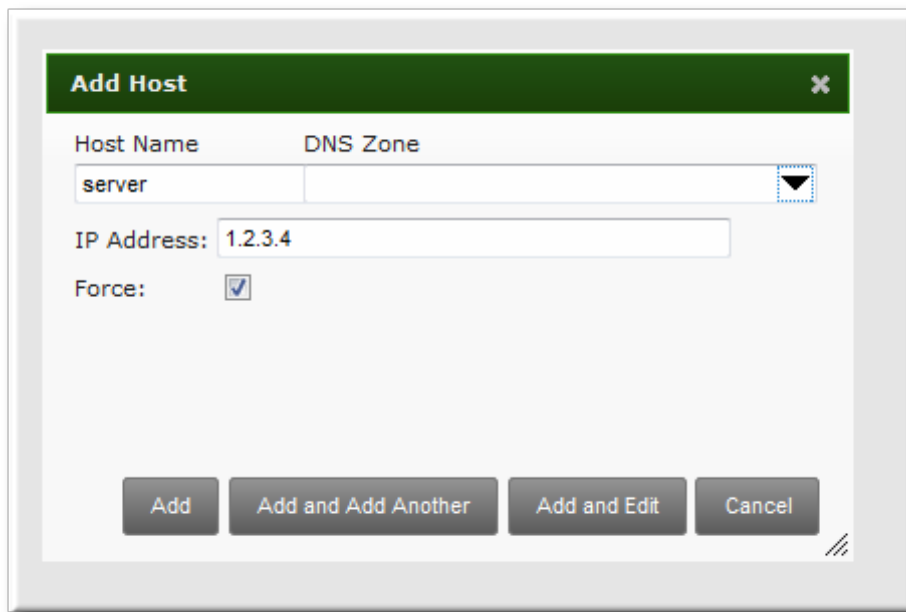
A host entry is always created when a client is configured. On Red Hat Enterprise Linux systems, this is done automatically with the `ipa-client-install` script. On other platforms, the host entry is created manually.

10.2.1. Adding Host Entries from the Web UI

1. Open the **Identity** tab, and select the **Hosts** subtab.
2. Click the **Add** link at the top of the hosts list.



3. Fill in the machine name and select the domain from the configured zones in the drop-down list. If the host has already been assigned a static IP address, then include that with the host entry so that the DNS entry is fully created.



DNS zones can be created in IdM, which is described in [Section 16.7.1, “Adding Forward DNS Zones”](#). If the IdM server does not manage the DNS server, the zone can be entered manually in the menu area, like a regular text field.



NOTE

Select the **Force** checkbox to add the host DNS record, even if the hostname cannot be resolved.

This is useful for hosts which use DHCP and do not have a static IP address. This essentially creates a placeholder entry in the IdM DNS service. When the DNS service dynamically updates its records, the host's current IP address is detected and its DNS record is updated.

4. Click the **Add and Edit** button to go directly to the expanded entry page and fill in more attribute information. Information about the host hardware and physical location can be included with the host entry.

The screenshot shows the Red Hat Identity Management web interface. The top navigation bar has tabs for 'Identity', 'Policy', and 'IPA Server'. Under 'Identity', there are sub-tabs for 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', 'Services', and 'DNS'. The 'Hosts' tab is selected, and the breadcrumb path is 'Hosts » server.example.com'. The main heading is 'HOST: server.example.com'. Below this, it states 'server.example.com is a member of these:' followed by buttons for 'Settings', 'Host Groups', 'Netgroups', 'Roles', 'HBAC Rules', 'Sudo Rules', and 'Hosts (1)'. To the right, it says 'server.example.com is managed by:' followed by an 'Expand All' link. Below the navigation bar, there are 'Reset' and 'Update' buttons. The main content area is titled 'HOST SETTINGS' and contains several input fields: 'Host name' (pre-filled with 'server.example.com'), 'Principal name' (pre-filled with 'host/server.example.com@EXAMPLE.COM'), 'Description', 'Locality', 'Location', 'Platform', and 'Operating system'. Below the settings section, there are two expandable sections: 'ENROLLMENT' and 'HOST CERTIFICATE'.

10.2.2. Adding Host Entries from the Command Line

Host entries are created using the **host-add** command. This command adds the host entry to the IdM Directory Server. The full list of options with **host-add** are listed in the **ipa host** manpage. At its most basic, an add operation only requires the client hostname to add the client to the Kerberos realm and to create an entry in the IdM LDAP server:

```
$ ipa host-add client1.example.com
```

If the IdM server is configured to manage DNS, then the host can also be added to the DNS resource records using the **--ip-address** and **--force** options.

Example 10.1. Creating Host Entries with Static IP Addresses

```
$ ipa host-add --force --ip-address=192.168.166.31 client1.example.com
```

Commonly, hosts may not have a static IP address or the IP address may not be known at the time the client is configured. For example, laptops may be preconfigured as Identity Management clients, but they do not have IP addresses at the time they're configured. Hosts which use DHCP can still be configured with a DNS entry by using **--force**. This essentially creates a placeholder entry in the IdM DNS service. When the DNS service dynamically updates its records, the host's current IP address is detected and its DNS record is updated.

Example 10.2. Creating Host Entries with DHCP

```
$ ipa host-add --force client1.example.com
```

Host records are deleted using the **host-del** command. If the IdM domain uses DNS, then the **--updatedns** option also removes the associated records of any kind for the host from the DNS.

```
$ ipa host-del --updatedns client1.example.com
```

10.3. Disabling and Re-enabling Host Entries

Active hosts can be accessed by other services, hosts, and users within the domain. There can be situations when it is necessary to remove a host from activity. However, deleting a host removes the entry and all the associated configuration, and it removes it permanently.

10.3.1. Disabling Host Entries

Disabling a host prevents domain users from access it without permanently removing it from the domain. This can be done by using the **host-disable** command.

For example:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa host-disable server.example.com
```



IMPORTANT

Disabling a host entry not only disables that host. It disables every configured service on that host as well.

10.3.2. Re-enabling Hosts

Disabling a host essentially kills its current, active keytabs. Removing the keytabs effectively removes the host from the IdM domain without otherwise touching its configuration entry.

To re-enable a host, simply use the **ipa-getkeytab** command. The **-s** option sets which IdM server to request the keytab, **-p** gives the principal name, and **-k** gives the file to which to save the keytab.

For example, requesting a new host keytab:

```
[jsmith@ipaserver ~]$ ipa-getkeytab -s ipaserver.example.com -p
host/server.example.com -k /etc/krb5.keytab -D
fqdn=server.example.com,cn=computers,cn=accounts,dc=example,dc=com -w password
```

If the **ipa-getkeytab** command is run on an active IdM client or server, then it can be run without any LDAP credentials (**-D** and **-w**). The IdM user uses Kerberos credentials to authenticate to the domain. To run the command directly on the disabled host, then supply LDAP credentials to authenticate to the IdM server. The credentials should correspond to the host or service which is being re-enabled.

10.4. Managing Public SSH Keys for Hosts

OpenSSH uses *public-private key pairs* to authenticate hosts. One machine attempts to access another machine and presents its key pair. The first time the host authenticates, the administrator on the target machine has to approve the request manually. The machine then stores the host's public key in a **known_hosts** file. Any time that the remote machine attempts to access the target machine again, the target machine simply checks its **known_hosts** file and then grants access automatically to approved hosts.

There are a few problems with this system:

- The **known_hosts** file stores host entries in a triplet of the host IP address, hostname, and key. This file can rapidly become out of date if the IP address changes (which is common in virtual environments and data centers) or if the key is updated.
- SSH keys have to be distributed manually and separately to all machines in an environment.
- Administrators have to approve host keys to add them to the configuration, but it is difficult to verify either the host or key issuer properly, which can create security problems.

On Red Hat Enterprise Linux, the System Security Services Daemon (SSSD) can be configured to cache and retrieve host SSH keys so that applications and services only have to look in one location for host keys. Because SSSD can use Identity Management as one of its identity information providers, Identity Management provides a universal and centralized repository of keys. Administrators do not need to worry about distributing, updating, or verifying host SSH keys.

10.4.1. About the SSH Key Format

When keys are uploaded to the IdM entry, the key format can be either an [OpenSSH-style key](#) or a raw [RFC 4253-style blob](#). Any RFC 4253-style key is automatically converted into an OpenSSH-style key before it is imported and saved into the IdM LDAP server.

The IdM server can identify the type of key, such as an RSA or DSA key, from the uploaded key blob. However, in a key file such as `~/.ssh/known_hosts`, a key entry is identified by the hostname and IP address of the server, its type, then lastly the key itself. For example:

```
host.example.com,1.2.3.4 ssh-rsa AAA...ZZZ==
```

This is slightly different than a user public key entry, which has the elements in the order *type key== comment*:

```
"ssh-rsa ABCD1234...== ipaclient.example.com"
```

All three parts from the key file can be uploaded to and viewed for the host entry. In that case, the host public key entry from the `~/.ssh/known_hosts` file needs to be reordered to match the format of a user key, *type key== comment*:

```
ssh-rsa AAA...ZZZ== host.example.com,1.2.3.4
```

The key type can be determined automatically from the content of the public key, and the comment is optional, to make identifying individual keys easier. The only required element is the public key blob itself.

10.4.2. About ipa-client-install and OpenSSH

The **ipa-client-install** script, by default, configures an OpenSSH server and client on the IdM client machine. It also configures SSSD to perform host and user key caching. Essentially, simply

configuring the client does all of the configuration necessary for the host to use SSSD, OpenSSH, and Identity Management for key caching and retrieval.



NOTE

Even if the machine is added as an IdM client using **ipa-client-install**, the client is not created with any SSH keys. These keys need to be created separately and added to the host account, as described in [Section 10.4.4, “Adding Host Keys from the Command Line”](#).

There is an additional client configuration option, **--ssh-trust-dns**, which can be run with **ipa-client-install** and automatically configures OpenSSH to trust the IdM DNS records, where the host keys are stored.

Alternatively, it is possible to disable OpenSSH at the time the client is installed, using the **--no-sshd** option. This prevents the install script from configuring the OpenSSH server.

Another option, **--no-dns-sshfp**, prevents the host from creating DNS SSHFP records with its own DNS entries. This can be used with or without the **--no-sshd** option.

10.4.3. Uploading Host SSH Keys Through the Web UI

1. The key for a host can probably be retrieved from a `~/ .ssh/known_hosts`. For example:

```
server.example.com,1.2.3.4 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAQvjbVsfSKtU0WQW4e0weeo0DZZ08F9Ud21x1Ly6F0hzwpxFG
IyxvXZ52+siHBHbbqGL5+14N7UvElruyslIHx9LYUR/pPKSMXCGyboLy5aTN150Q5EHwrhVnFDIKX
kvp45945R7SKYCUtRumm0Iw6wq0XD4o+ILeVbV3wmcB1bXs36ZvC/M6riefn9PcJmh6vNCvIsbMY
6S+FhkwUTTi0XJjUDYRLlwM273FfWhzHK+SSQXeBp/zIn1gFvJhSZMRi9HZpDoqxLbBB9QIdIw6U4
MIjNmKsSI/ASpkFm2GuQ7ZK9KuMItY2AoCuIRmRAdF8iYNHBTXNfFurGogXwRDjQ==
```

If necessary, generate a host key. When using the OpenSSH tools, make sure to use a blank passphrase and to save the key to a different location than the user's `~/ .ssh/` directory, so it will not overwrite any existing keys.

```
[jsmith@server ~]$ ssh-keygen -t rsa -C "server.example.com,1.2.3.4"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jsmith/.ssh/id_rsa):
/home/jsmith/.ssh/host_keys
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jsmith/.ssh/host_keys.
Your public key has been saved in /home/jsmith/.ssh/host_keys.pub.
The key fingerprint is:
4f:61:ee:2c:f7:d7:da:41:17:93:de:1d:19:ac:2e:c8 server.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           .. |
|           .+ |
|          o  .* |
|         o . . * |
|        S + . o+ |
|         E . . . |
|         . = . o |
|         o . .o |
|           .....|
+-----+

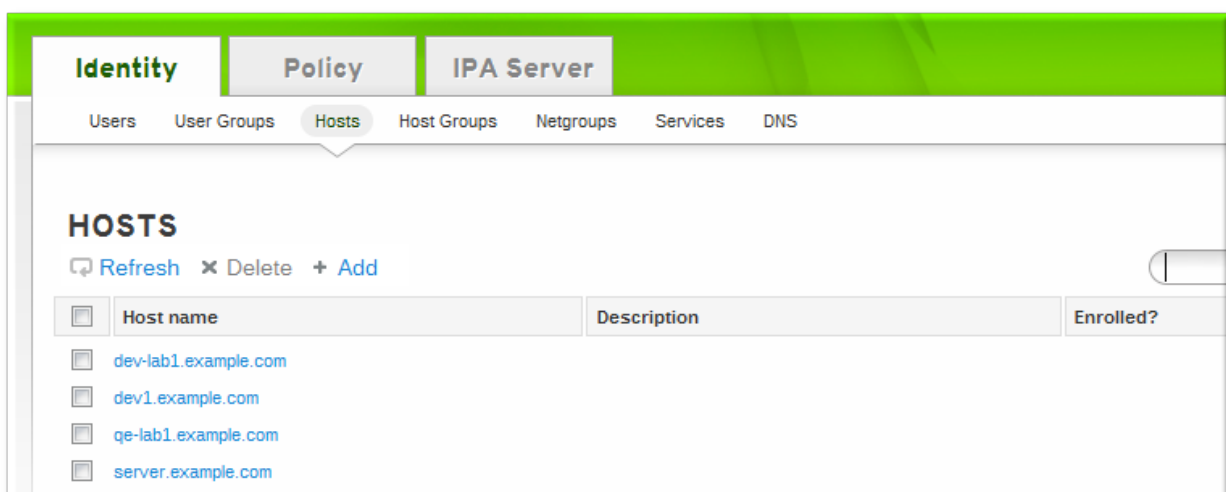
```

2. Copy the public key from the key file. The full key entry has the form *hostname,IP type key==*. Only the *key==* is required, but the entire entry can be stored. To use all elements in the entry, rearrange the entry so it has the order *type key== [hostname,IP]*

```
[jsmith@server ~]$ cat /home/jsmith/.ssh/host_keys.pub

ssh-rsa AAAAB3NzaC1yc2E...tJG1PK2Mq++wQ== server.example.com,1.2.3.4
```

3. Open the **Identity** tab, and select the **Hosts** subtab.
4. Click the name of the host to edit.



5. In the **Host Settings** area of the **Settings** tab, click the **SSH public keys: Add** link.

The screenshot shows the Red Hat Identity Management (IdM) web interface. The top navigation bar includes 'Identity', 'Policy', and 'IPA Server'. Below this, a sub-navigation bar shows 'Users', 'User Groups', 'Hosts' (selected), 'Host Groups', 'Netgroups', and 'Services'. The main content area is titled 'HOST: server.example.com' and includes a breadcrumb 'Hosts » server.example.com'. Below the title, there are tabs for 'Settings', 'Host Groups', 'Netgroups', 'Roles', 'HBAC Rules', 'Sudo Rules', and 'Hosts (1)'. The 'Settings' tab is active. Below the tabs are buttons for 'Refresh', 'Reset', and 'Update'. The 'HOST SETTINGS' section contains the following fields:

- Host name: server.example.com
- Principal name: host/server.example.com@RHTS.ENG.BOS.REDHAT.COM
- Description: (empty text area)
- Locality: (empty text field)
- Location: (empty text field)
- Platform: (empty text field)
- Operating system: (empty text field)
- SSH public keys: Add (highlighted with a red box)
- MAC address: Add

- The UI opens a new link, **New: key not set Show/Set key**. Click the **Show/Set key** link.

▼ HOST SETTINGS

Host name: **server.example.com**

Principal name: **host/server.example.com@RHTS.ENG.BOS.REDHAT.COM**

Description:

Locality:

Location:

Platform:

Operating system:

SSH public keys: **New: key not set** [Show/Set key](#) [undo](#)

[Add](#) [undo all](#)

MAC address: [Add](#)

7. Paste in the public key for the host, and click the **Set** button.

Set SSH key

SSH public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1+Rpdb1Y7UNTShs8xH2IrvF1vtse5ort4ziqay8i
9vH7+p1fyKJ6x5f20YAXgAbR/Q3nW4P0TQ0UsY3d5hNDCsIueIqBr461gJ97rv7FJ4jo
a
/bdLxV4ImHkLaz5PEd5JJGkJSukA4sugvUwzr7UOnhzma9E8H+7EiIM6JX2CqhajK0YT
2I9T9dYfRS
/VJ5dz2xkG1ZE+Syu3m4D5kJAQEjgKuaPgKYP3LSPdGT1KnSZwOolfav+buFMmwd6Smr
ThFGhz7/0F/HX5s_jhk2kF0r5cgdDjuaF0d
/Ve3+2hNIfb2txBE7T5HqUXekTbfusKcsUUbGrjtOkCPCyz4JCn+Q==
server.example.com
```

[Set](#) [Cancel](#)

The **SSH public keys** field now shows **New: key set**. Clicking the **Show/Set key** link opens the submitted key.

8. To upload multiple keys, click the **Add** link below the list of public keys, and upload the other keys.
9. When all the keys have been submitted, click the **Update** link at the top of the host's page to save the changes.

When the public key is saved, the entry is displayed as the key fingerprint, the comment (if one was included), and the key type ^[4].

The screenshot shows the 'HOST SETTINGS' page for a host named 'server.example.com'. The principal name is 'host/server.example.com@RHTS.ENG.BOS.REDHAT.COM'. There is a large text area for 'Description'. Below it are input fields for 'Locality', 'Location', 'Platform', and 'Operating system'. At the bottom, there is a section for 'SSH public keys' which contains one entry: 'BC:BD:BF:81:51:A5:74:07:C2:D5:EE:11:8C:95:48:3C server.example.com (ssh-rsa)'. This entry is highlighted with a red rectangular box. To the right of the key are links for 'Show/Set key' and 'Delete'. Below the highlighted entry is an 'Add' link. At the very bottom, there is a 'MAC address' field with an 'Add' link.

Figure 10.1. Saved Public Key

After uploading the host keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing host keys. This is covered in the [Red Hat Enterprise Linux Deployment Guide](#).

10.4.4. Adding Host Keys from the Command Line

Host SSH keys are added to host entries in IdM, either when the host is created using **host-add** or by modifying the entry later.



NOTE

Host keys are not created by the **ipa-client-install** command.

1. Run the **host-mod** command with the **--sshpubkey** option to upload the 64 bit-encoded public key to the host entry.
Adding a host key also changes the DNS SSHFP entry for the host, so also use the **--updatedns** option to update the host's DNS entry.
For example:

```
[jsmith@server ~]$ ipa host-mod --sshpubkey="ssh-rsa 12345abcde==
ipaclient.example.com" --updatedns host1.example.com
```

With a real key, the key is longer and usually ends with an equals sign (=).

To upload multiple keys, pass a comma-separated list of keys with a single **--sshpubkey** option:

```
--sshpubkey="12345abcde==, key2==, key3=="
```



TIP

A host can have multiple public keys.

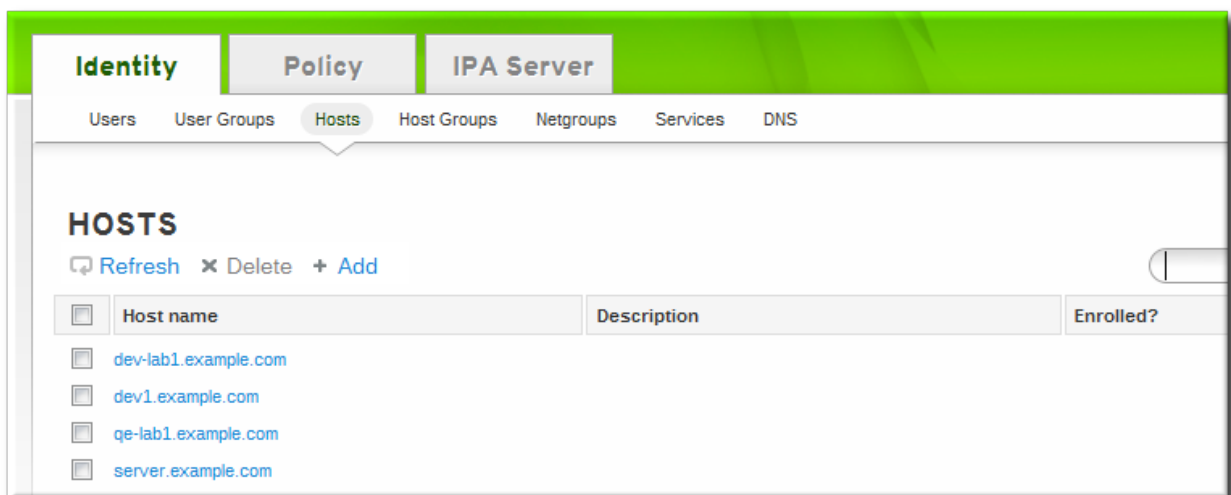
2. After uploading the host keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing host keys. This is covered in the [Red Hat Enterprise Linux Deployment Guide](#).

10.4.5. Removing Host Keys

Host keys can be removed once they expire or are no longer valid.

To remove an individual host key, it is easiest to remove the key through the web UI:

1. Open the **Identity** tab, and select the **Hosts** subtab.
2. Click the name of the host to edit.



3. Open the **Host Settings** area of the **Settings** tab.
4. Click the **Delete** link by the fingerprint of the key to remove.

▼ HOST SETTINGS

Host name: **server.example.com**

Principal name: **host/server.example.com@RHTS.ENG.BOS.REDHAT.COM**

Description:

Locality:

Location:

Platform:

Operating system:

SSH public keys: **BC:BD:BF:81:51:A5:74:07:C2:D5:EE:11:8C:95:48:3C server.example.com (ssh-rsa)** [Show/Set key](#) [Delete](#)

[Add](#)

MAC address: [Add](#)

5. Click the **Update** link at the top of the host's page to save the changes.

The command-line tools can be used to remove all keys. This is done by running **ipa host-mod** with the **--sshpubkey=** set to a blank value; this removes *all* public keys for the host. Also, use the **--updatedns** option to update the host's DNS entry. For example:

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa host-mod --sshpubkey= --updatedns host1.example.com
```

10.5. Renaming Machines and Reconfiguring IdM Client Configuration

The hostname of a system is critical for the correct operation of Kerberos and SSL. Both of these security mechanisms rely on the hostname to ensure that communication is occurring between the specified hosts. Infrastructures which use virtual machines or clustered servers will commonly have hosts which are renamed because systems are copied, moved, or renamed.

Red Hat Enterprise Linux does not provide a simple rename command to facilitate the renaming of an IdM host. Renaming a host in an IdM domain involves deleting the entry in IdM, uninstalling the client software, changing the hostname, and re-enrolling using the new name. Additionally, part of renaming hosts requires regenerating service principals.

To reconfigure the client:

1. Identify which services are running on the machine. These need to be re-created when the machine is re-enrolled.

```
# ipa service-find server.example.com
```

Each host has a default service which does not appear in the list of services. This service can be referred to as the "host service". The service principal for the host service is **host/<hostname>**, such as **host/server.example.com**. This principal can also be referred

to as the *host principal*.

2. Identify all host groups to which the machine belongs.

```
# ipa hostgroup-find server.example.com
```

Identify which of the services have certificates associated with them. This can be done using the **ldapsearch** command to check the entries in the IdM LDAP database directly:

```
# ldapsearch -x -b "cn=accounts,dc=example,dc=com"
"(&(objectclass=ipaservice)(userCertificate=*))" krbPrincipalName
```

3. For any service principals (in addition to the host principal), determine the location of the corresponding keytabs on **server.example.com**. The keytab location is different for each service, and IdM does not store this information.

Each service on the client system has a Kerberos principal in the form *service name/hostname@REALM*, such as **ldap/server.example.com@EXAMPLE.COM**.

4. Unenroll the client machine from the IdM domain:

```
# ipa-client-install --uninstall
```

5. For each identified keytab other than **/etc/krb5.keytab**, remove the old principals:

```
# ipa-rmkeytab -k /path/to/keytab -r EXAMPLE.COM
```

6. On another IdM machine, as an IdM administrator, remove the host entry. This removes all services and revokes all certificates issued for that host:

```
# ipa host-del server.example.com
```

At this point, the host is completely removed from IdM.

7. Rename the machine.
8. Re-enroll the system with IdM:

```
# ipa-client-install
```

This generates a host principal for with the new hostname in **/etc/krb5.keytab**.

9. For every service that needs a new keytab, run the following command:

```
# ipa service-add serviceName/new-hostname
```

10. To generate certificates for services, use either **certmonger** or the IdM administration tools.
11. Re-add the host to any applicable host groups.

10.6. Managing Host Groups

Host groups are a way of centralizing control over important management tasks, particularly access control.

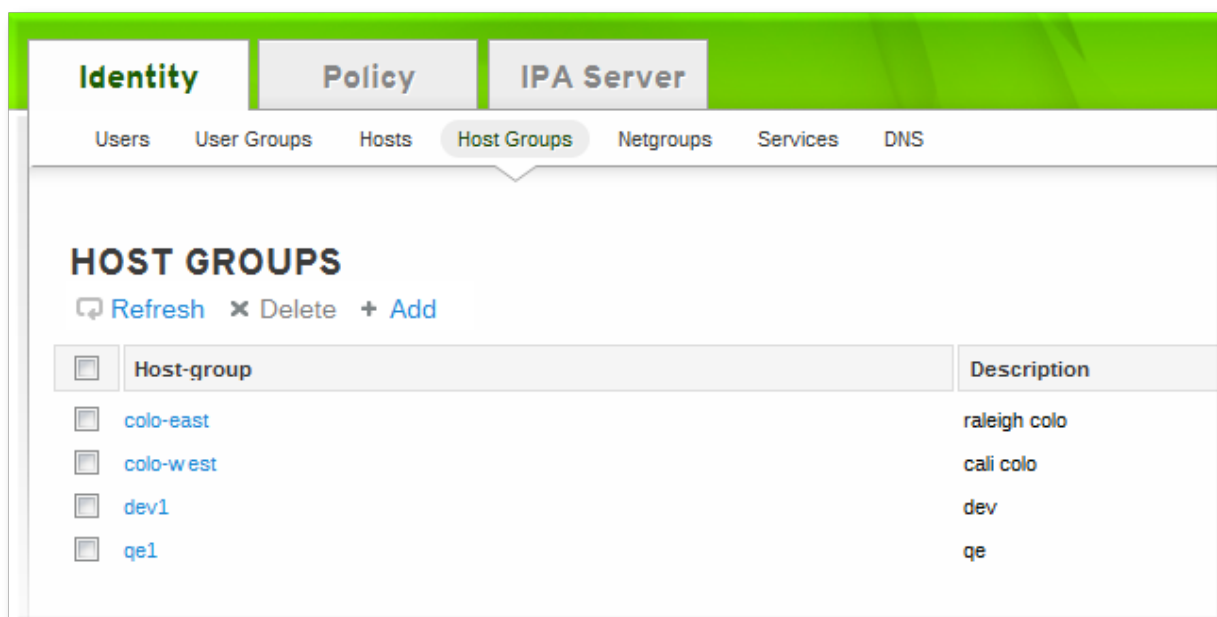
All groups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, IdM allows *nested groups*, where a group is a member of another group. In that case, all of the group members of the member group automatically belong to the parent group, as well.

Because groups are easy to create, it is possible to be very flexible in what groups to create and how they are organized. Groups can be defined around organizational divisions like departments, physical locations, or IdM or infrastructure usage guidelines for access controls.

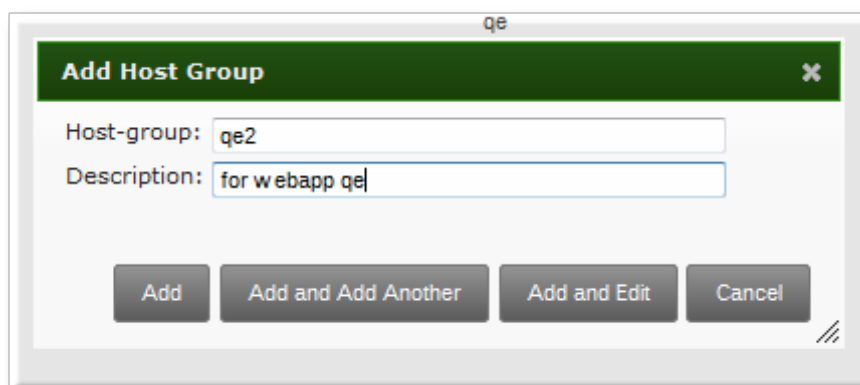
10.6.1. Creating Host Groups

10.6.1.1. Creating Host Groups from the Web UI

1. Open the **Identity** tab, and select the **Host Groups** subtab.
2. Click the **Add** link at the top of the groups list.



3. Enter the name and a description for the group.



4. Click the **Add and Edit** button to go immediately to the member selection page.
5. Select the members, as described in [Section 10.6.2.2, “Adding Group Members from the Web UI”](#).

10.6.1.2. Creating Host Groups from the Command Line

New groups are created using the **hostgroup-add** command. (This adds only the group; members are added separately.)

Two attributes are always required: the group name and the group description. If those attributes are not given as arguments, then the script prompts for them.

```
$ ipa hostgroup-add groupName --desc="description"
```

10.6.2. Adding Group Members

10.6.2.1. Showing and Changing Group Members

Members can be added to a group through the group configuration. There are tabs for all the member types which can belong to the group, and an administrator picks all of the marching entries and adds them as members.

However, it is also possible for an entity to be added to a group through its own configuration. Each entry has a list of tabs that displays group types that the entry can join. The list of all groups of that type are displayed, and the entity can be added to multiple groups at the same time.

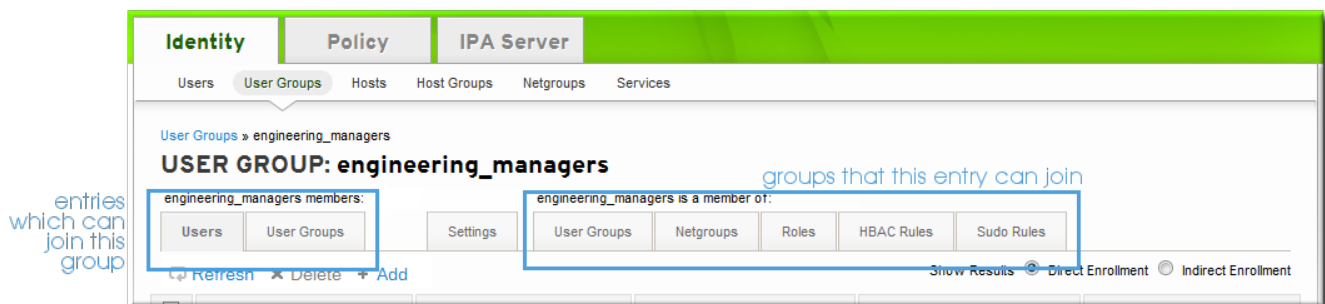
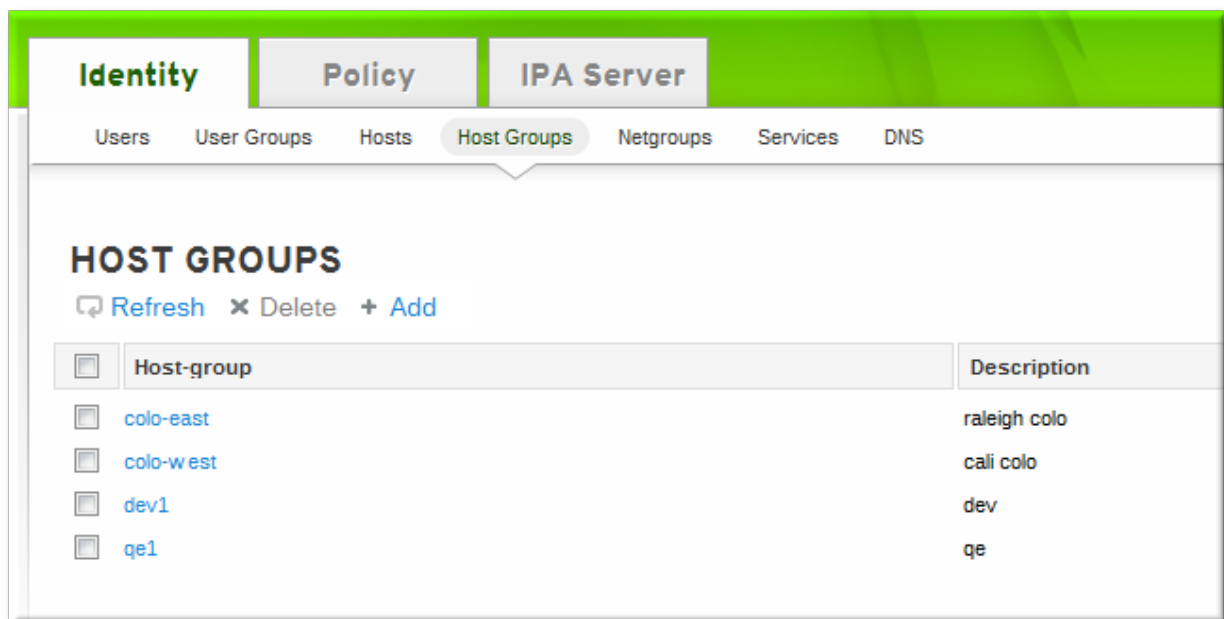


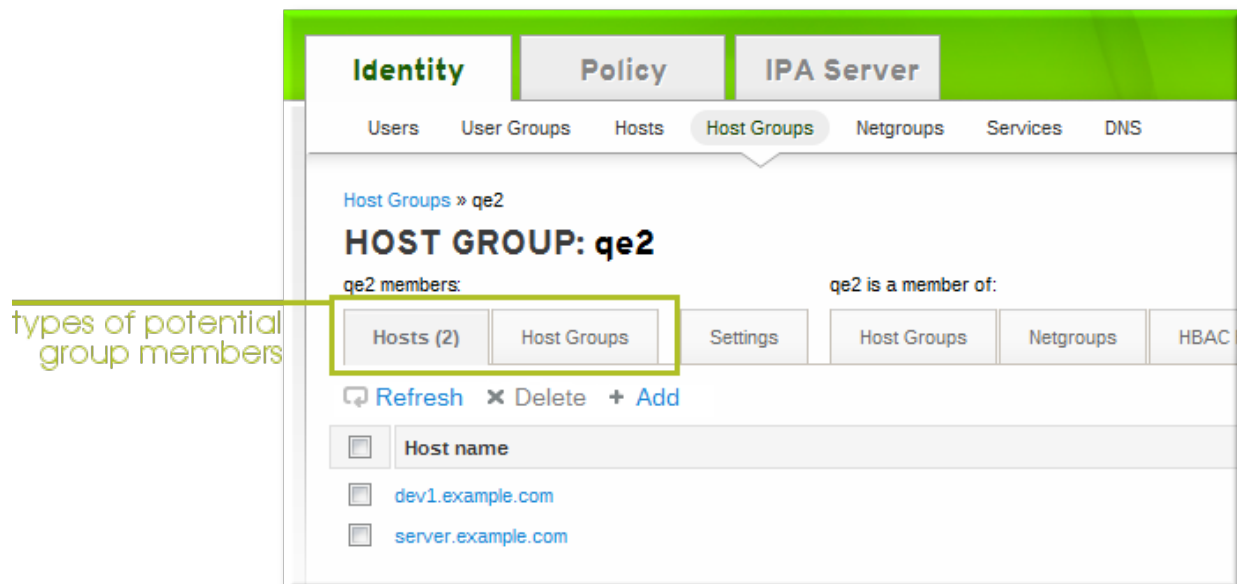
Figure 10.2. Member Of...

10.6.2.2. Adding Group Members from the Web UI

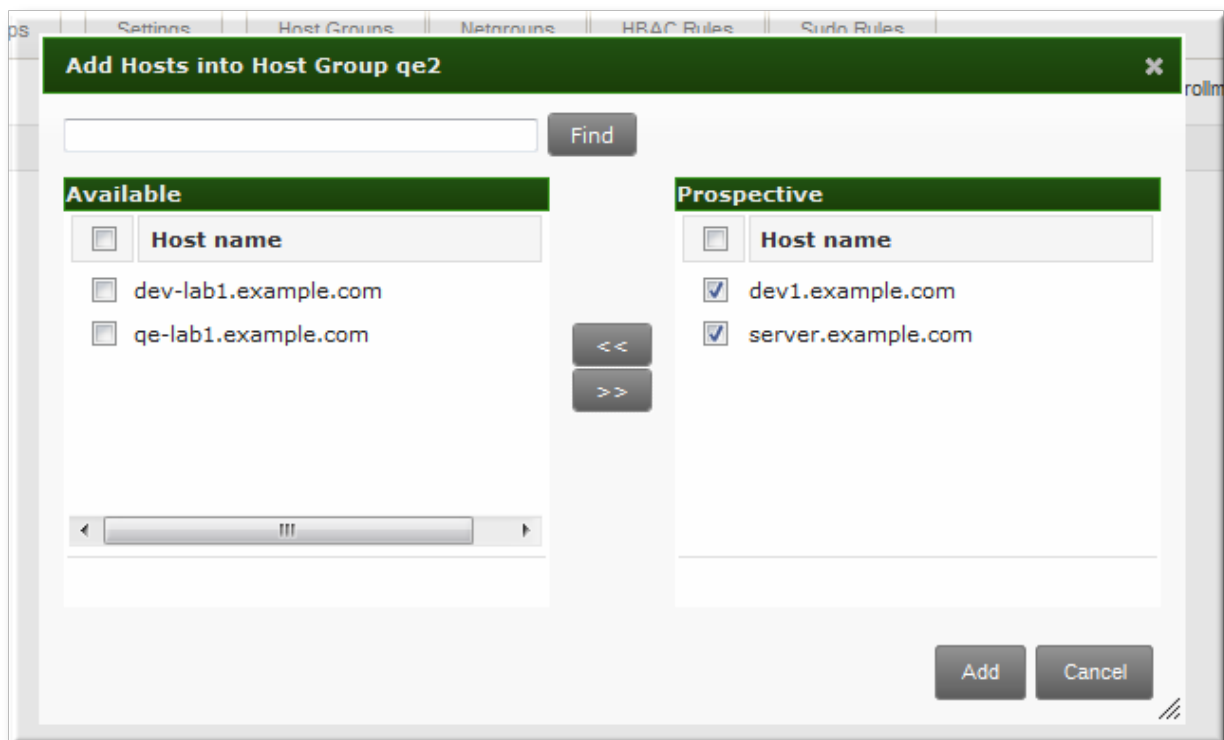
1. Open the **Identity** tab, and select the **Host Groups** subtab.
2. Click the name of the group to which to add members.



3. Click the **Add** link at the top of the task area.



4. Click the checkbox by the names of the hosts to add, and click the right arrows button, >>, to move the hosts to the selection box.



5. Click the **Add** button.

10.6.2.3. Adding Group Members from the Command Line

Members are added to a host group using the **hostgroup-add-member** command. This command can add both hosts as group members and other groups as group members.

The syntax of the **hostgroup-add-member** command requires only the group name and a comma-separated list of hosts to add:

```
$ ipa hostgroup-add-member groupName [--hosts=list] [--hostgroups=list]
```

For example, this adds three hosts to the **caligroup** group:


```
$ ipa hostgroup-add-member caligroup --  
hosts=ipaserver.example.com,client1.example.com,client2.example.com  
Group name: caligroup  
Description: for machines in california  
GID: 387115842  
Member hosts: ipaserver.example.com,client1.example.com,client2.example.com  
-----  
Number of members added 3  
-----
```

Likewise, other groups can be added as members, which creates nested groups:

```
$ ipa hostgroup-add-member caligroup --groups=mountainview,sandiego  
Group name: caligroup  
Description: for machines in california  
GID: 387115842  
Member groups: mountainview,sandiego  
-----  
Number of members added 2  
-----
```

[4] The key type is determined automatically from the key itself, if it is not included in the uploaded key.

Chapter 11. Identity: Managing Services

Some services that run on a host can also belong to the IdM domain. Any service that can store a Kerberos principal or an SSL certificate (or both) can be configured as an IdM service. Adding a service to the IdM domain allows the service to request an SSL certificate or keytab from the domain. (Only the public key for the certificate is stored in the service record. The private key is local to the service.)

An IdM domain establishes a commonality between machines, with common identity information, common policies, and shared services. Any machine which belongs to a domain functions as a client of the domain, which means it uses the services that the domain provides. An IdM domain (as described in [Section 1.2, “Bringing Linux Services Together”](#)) provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

11.1. Adding and Editing Service Entries and Keytabs

As with host entries, service entries for the host (and any other services on that host which will belong to the domain) must be added manually to the IdM domain. This is a two step process. First, the service entry must be created, and then a keytab must be created for that service which it will use to access the domain.

By default, Identity Management saves its HTTP keytab to `/etc/httpd/conf/ipa.keytab`.



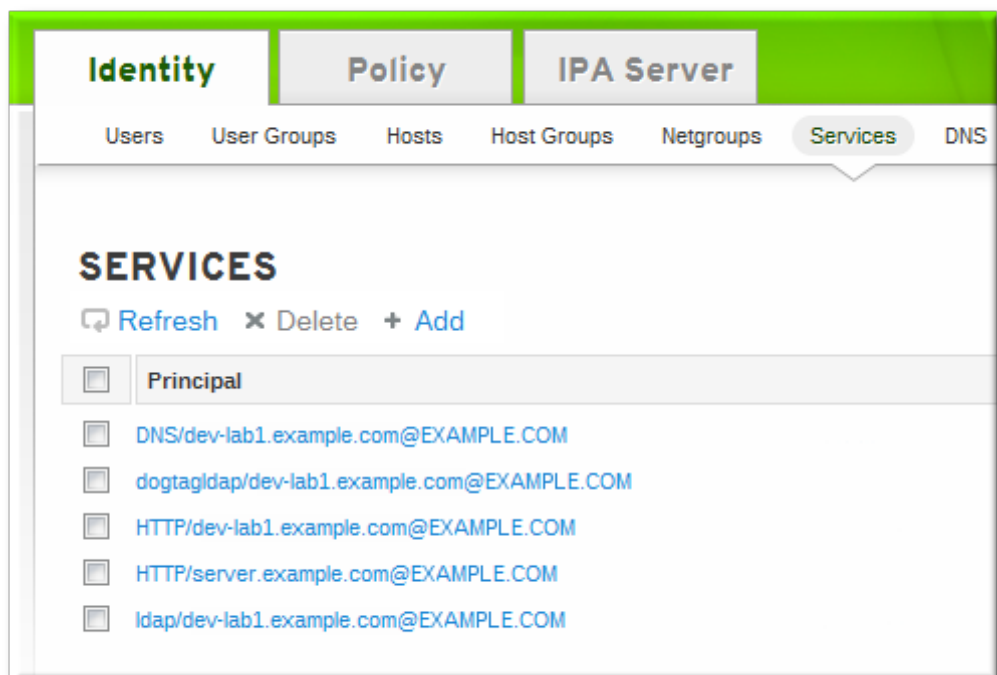
NOTE

This keytab is used for the web UI. If a key were stored in `ipa.keytab` and that keytab file is deleted, the IdM web UI will stop working, because the original key would also be deleted.

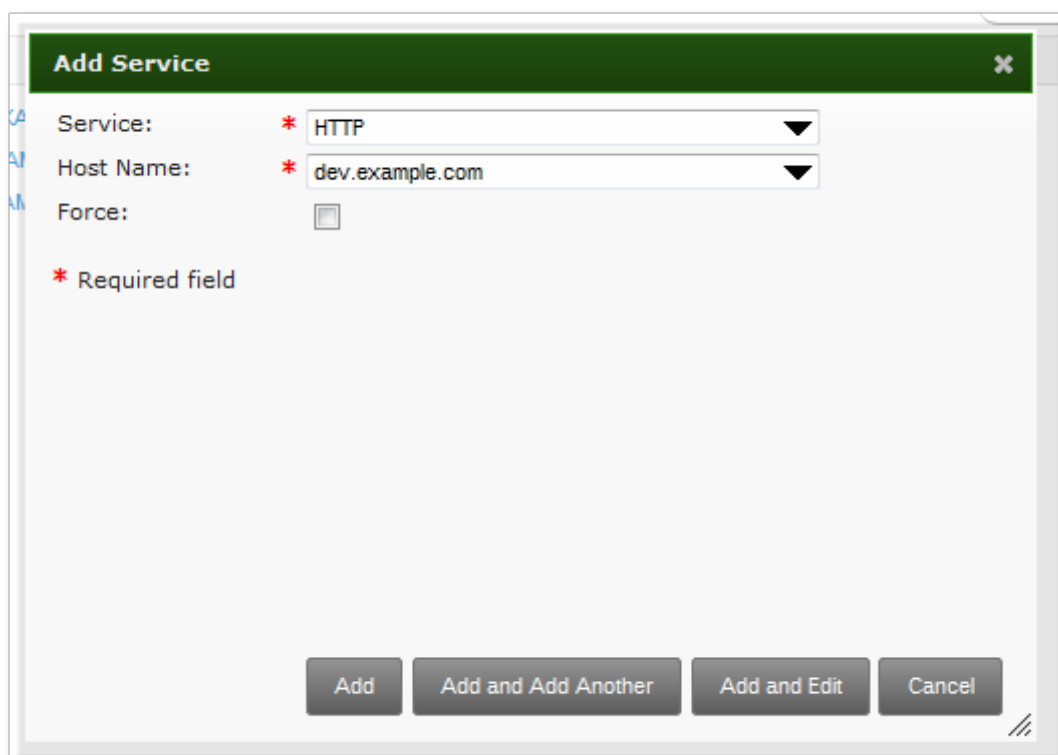
Similar locations can be specified for each service that needs to be made Kerberos aware. There is no specific location that must be used, but, when using `ipa-getkeytab`, you should avoid using `/etc/krb5.keytab`. This file should not contain service-specific keytabs; each service should have its keytab saved in a specific location and the access privileges (and possibly SELinux rules) should be configured so that only this service has access to the keytab.

11.1.1. Adding Services and Keytabs from the Web UI

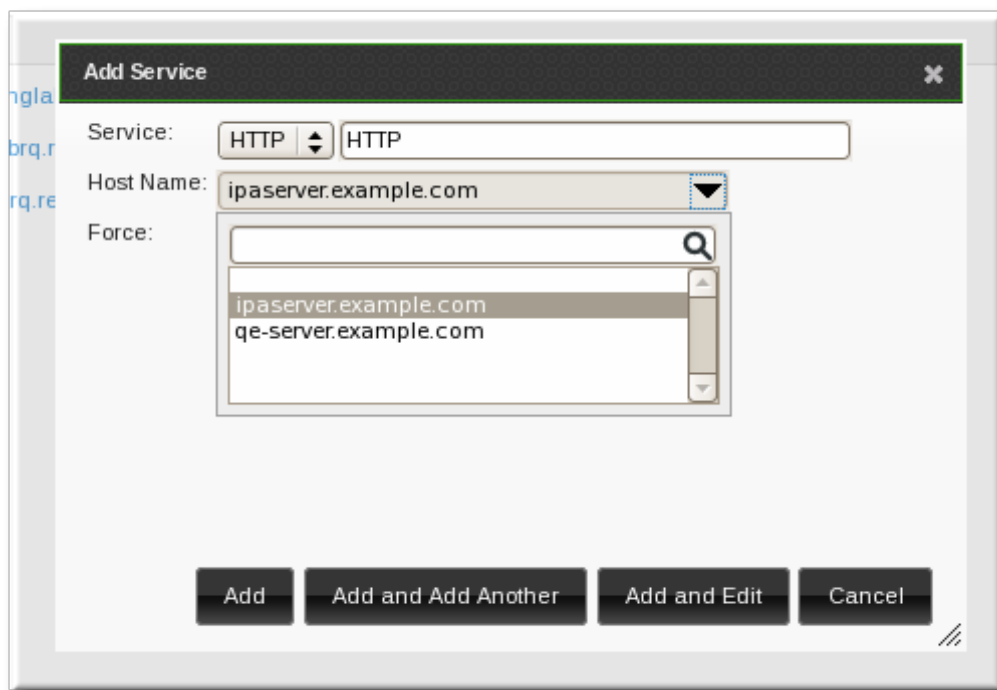
1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the **Add** link at the top of the services list.



3. Select the service type from the drop-down menu, and give it a name.



4. Select the hostname of the IdM host on which the service is running. The hostname is used to construct the full service principal name.



5. Click the **Add** button to save the new service principal.
6. Use the **ipa-getkeytab** command to generate and assign the new keytab for the service principal.

```
[root@ipaserver ~]# # ipa-getkeytab -s ipaserver.example.com -p
HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e des-cbc-crc
```

- The realm name is optional. The IdM server automatically appends the Kerberos realm for which it is configured. You cannot specify a different realm.
- The hostname must resolve to a DNS A record for it to work with Kerberos. You can use the **-force** flag to force the creation of a principal should this prove necessary.
- The **-e** argument can include a comma-separated list of encryption types to include in the keytab. This supersedes any default encryption type.



WARNING

Creating a new key resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

11.1.2. Adding Services and Keytabs from the Command Line

1. Create the service principal. The service is recognized through a name like *service/FQDN*:

```
# ipa service-add serviceName/hostname
```

For example:

```
$ ipa service-add HTTP/server.example.com
-----
Added service "HTTP/server.example.com@EXAMPLE.COM"
-----
Principal: HTTP/server.example.com@EXAMPLE.COM
Managed by: ipaserver.example.com
```

2. Create the service keytab file using the **ipa-getkeytab** command. This command is run on the client in the IdM domain. (Actually, it can be run on any IdM server or client, and then the keys copied to the appropriate machine. However, it is simplest to run the command on the machine with the service being created.)

The command requires the Kerberos service principal (**-p**), the IdM server name (**-s**), the file to write (**-k**), and the encryption method (**-e**). Be sure to copy the keytab to the appropriate directory for the service.

For example:

```
# ipa-getkeytab -s server.example.com -p HTTP/server.example.com -k
/etc/httpd/conf/krb5.keytab -e des-cbc-crc
```

- The realm name is optional. The IdM server automatically appends the Kerberos realm for which it is configured. You cannot specify a different realm.
- The hostname must resolve to a DNS A record for it to work with Kerberos. You can use the **-force** flag to force the creation of a principal should this prove necessary.
- The **-e** argument can include a comma-separated list of encryption types to include in the keytab. This supersedes any default encryption type.



WARNING

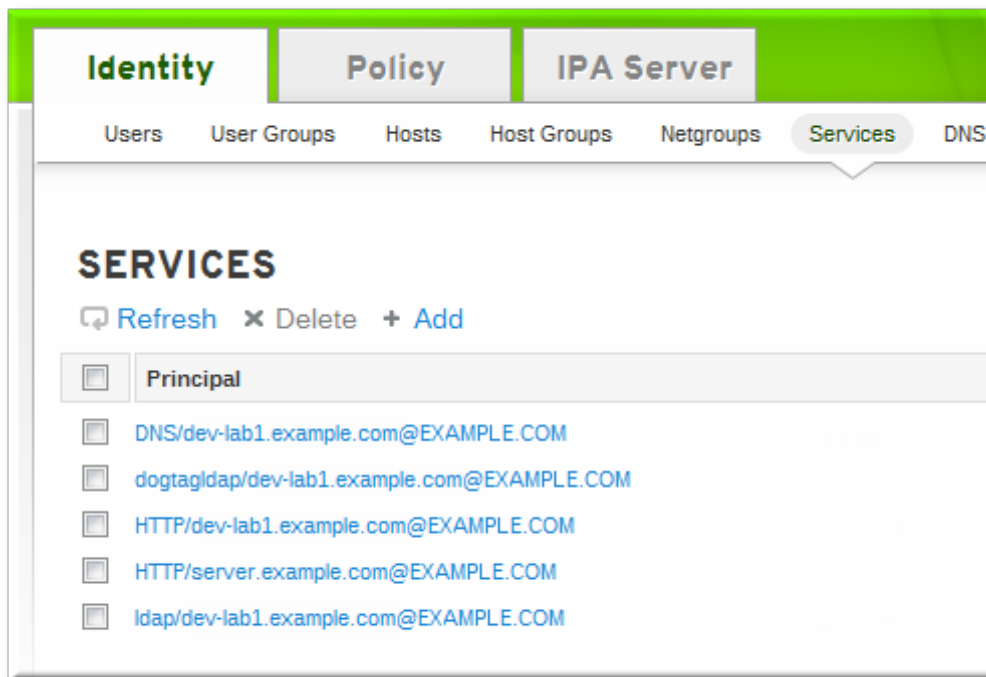
The **ipa-getkeytab** command resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

11.2. Adding Services and Certificates for Services

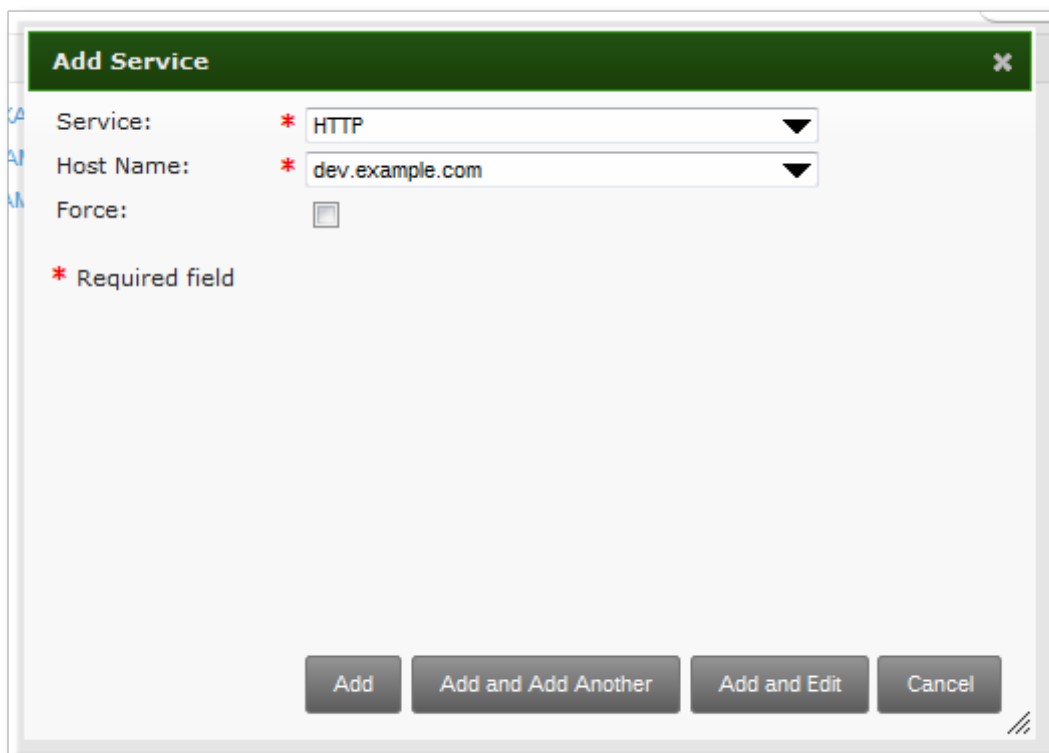
While services can use keytabs, some services require certificates for access. In that case, a service can be added (or modified) to include a certificate with its service entry.

11.2.1. Adding Services and Certificates from the Web UI

1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the **Add** link at the top of the services list.



3. Select the service type from the drop-down menu, and give it a name.



4. Select the hostname of the IdM host on which the service is running. The hostname is used to construct the full service principal name.

5. Click the **Add and Edit** button to go directly to the service entry page.
6. Scroll to the bottom of the page, to the **Service Certificate** section.
7. Click the **New Certificate** button to create the service certificate.

11.2.2. Adding Services and Certificates from the Command Line

1. Create the service principal. The service is recognized through a name like *service/FQDN*:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa service-add serviceName/hostname
```

For example:

```
$ ipa service-add HTTP/server.example.com

-----
Added service "HTTP/server.example.com@EXAMPLE.COM"
-----

Principal: HTTP/server.example.com@EXAMPLE.COM
Managed by: ipaserver.example.com
```

2. Create a certificate for the service. Be sure to copy the keytab to the appropriate directory for the service.

For example:

```
$ ipa cert-request --principal=HTTP/web.example.com example.csr
```

**TIP**

Use the **--add** option to create the service automatically when requesting the certificate.

Alternatively, use the **getcert** command, which creates and manages the certificate through **certmonger**. The options are described more in [Section B.1, “Requesting a Certificate with certmonger”](#).

```
$ ipa-getcert request -d /etc/httpd/alias -n Server-Cert -K
HTTP/client1.example.com -N 'CN=client1.example.com,O=EXAMPLE.COM'
```

11.3. Storing Certificates in NSS Databases

When services use certificates, the certificates and keys can be stored in NSS databases (which may also be used by the services themselves, as well as Identity Management).

1. Create the NSS databases.

```
$ certutil -N -d /path/to/database/dir
```

2. Request the certificate using **certutil**, an NSS tool.

```
$ certutil -R -s "CN=client1.example.com,O=EXAMPLE.COM" -d
/path/to/database/dir -a > example.csr
```

If the IdM domain is using Certificate System for its CA, only the CN of the subject name is used. With a self-signed CA, the subject must match the configured certificate subject base. The IdM server rejects requests with a subject base that differs from this value.

11.4. Configuring Clustered Services

The IdM server is not *cluster aware*. However, it is possible to configure a clustered service to be part of IdM by synchronizing Kerberos keys across all of the participating hosts and configuring services running on the hosts to respond to whatever names the clients use.

1. Enroll all of the hosts in the cluster into the IdM domain.
2. Create any service principals and generate the required keytabs.
3. Collect any keytabs that have been set up for services on the host, including the host keytab at **/etc/krb5.keytab**.
4. Use the **ktutil** command to produce a single keytab file that contains the contents of all of the keytab files.
 - a. For each file, use the **rkt** command to read the keys from that file.
 - b. Use the **wkt** command to write all of the keys which have been read to a new keytab file.
5. Replace the keytab files on each host with the newly-created combined keytab file.
6. At this point, each host in this cluster can now impersonate any other host.
7. Some services require additional configuration to accommodate cluster members which do not reset hostnames when taking over a failed service.

- For `sshd`, set **`GSSAPIStrictAcceptorCheck no`** in `/etc/ssh/sshd_config`.
- For `mod_auth_kerb`, set **`KrbServiceName Any`** in `/etc/httpd/conf.d/auth_kerb.conf`.



NOTE

For SSL servers, the subject name or a subject alternative name for the server's certificate must appear correct when a client connects to the clustered host. If possible, share the private key among all of the hosts.

If each cluster member contains a subject alternative name which includes the names of all the other cluster members, that satisfies any client connection requirements.

11.5. Using the Same Service Principal for Multiple Services

Within a cluster, the same service principal can be used for multiple services, spread across different machines.

1. Retrieve a service principal using the **`ipa-getkeytab`** command.

```
# ipa-getkeytab -s kdc.example.com -p HTTP/server.example.com -k
/etc/httpd/conf/krb5.keytab -e des-cbc-crc
```

2. Either direct multiple servers or services to use the same file, or copy the file to individual servers as required.

11.6. Disabling and Re-enabling Service Entries

Active services can be accessed by other services, hosts, and users within the domain. There can be situations when it is necessary to remove a host or a service from activity. However, deleting a service or a host removes the entry and all the associated configuration, and it removes it permanently.

11.6.1. Disabling Service Entries

Disabling a service prevents domain users from access it without permanently removing it from the domain. This can be done by using the **`service-disable`** command.

For a service, specify the principal for the service. For example:

```
[jsmith@ipaserver ~]$ kinit admin
$ ipa service-disable http/server.example.com
```



IMPORTANT

Disabling a host entry not only disables that host. It disables every configured service on that host as well.

11.6.2. Re-enabling and Services

Disabling a service essentially kills its current, active keytabs. Removing the keytabs effectively removes the service from the IdM domain without otherwise touching its configuration entry.

To re-enable a service, simply use the **ipa-getkeytab** command. The **-s** option sets which IdM server to request the keytab, **-p** gives the principal name, and **-k** gives the file to which to save the keytab.

For example, requesting a new HTTP keytab:

```
[root@ipaserver ~]# ipa-getkeytab -s ipaserver.example.com -p  
HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e des-cbc-crc
```

If the **ipa-getkeytab** command is run on an active IdM client or server, then it can be run without any LDAP credentials (**-D** and **-w**). The IdM user uses Kerberos credentials to authenticate to the domain. To run the command directly on a disabled host, then supply LDAP credentials to authenticate to the IdM server. The credentials should correspond to the host or service which is being re-enabled.

Chapter 12. Identity: Delegating Access to Hosts and Services

As discussed in [Section 1.3, “Relationships Between Servers and Clients”](#), within the IdM domain, *manage* means being able to retrieve a keytab and certificates for another host or service. Every host and service has a ***managedby*** entry which lists what hosts or services can manage it. By default, a host can manage itself and all of its services. It is also possible to allow a host to manage other hosts, or services on other hosts, by updating the appropriate delegations or providing a suitable ***managedby*** entry.

An IdM service can be managed from any IdM host, as long as that host has been granted, or *delegated*, permission to access the service. Likewise, hosts can be delegated permissions to other hosts within the domain.

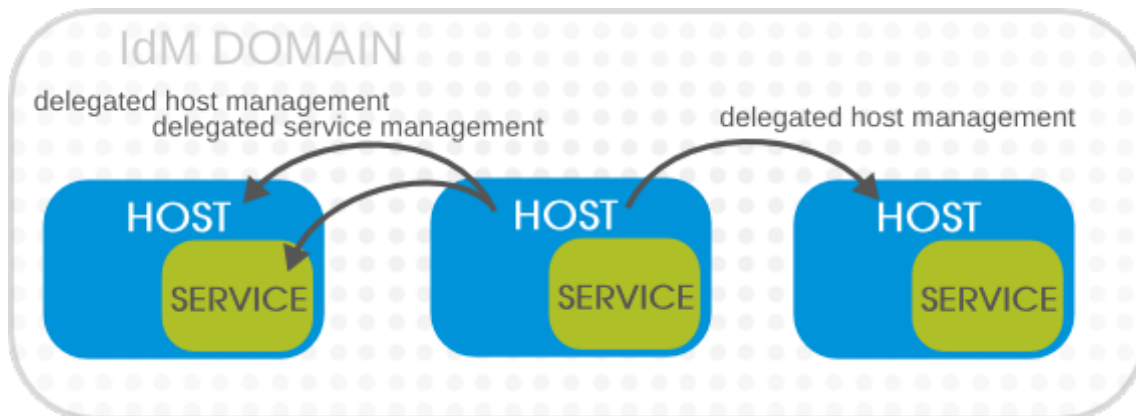


Figure 12.1. Host and Service Delegation



NOTE

If a host is delegated authority to another host through a ***managedBy*** entry, it does not mean that the host has also been delegated management for all services on that host. Each delegation has to be performed independently.

12.1. Delegating Service Management

A host is delegated control over a service using the **`service-add-host`** command. There are two parts to delegating the service: specifying the principal and identifying the hosts (in a comma-separated list) with control:

```
# ipa service-add-host principal --hosts=hostnames
```

For example:

```
# ipa service-add-host http/web.example.com --hosts=client1.example.com
```

Once the host is delegated authority, the host principal can be used to manage the service:

```
# kinit -kt /etc/krb5.keytab host/`hostname`
# ipa-getkeytab -s `hostname` -k /tmp/test.keytab -p http/web.example.com
Keytab successfully retrieved and stored in: /tmp/test.keytab
```

To create a ticket for this service, create a certificate request on the host with the delegated authority and use the **cert-request** command to create a service entry and load the certification information:

```
# ipa cert-request --add --principal=http/web.example.com web.csr
Certificate: MIICETCCAXqgA...[snip]
Subject: CN=web.example.com,O=EXAMPLE.COM
Issuer: CN=EXAMPLE.COM Certificate Authority
Not Before: Tue Feb 08 18:51:51 2011 UTC
Not After: Mon Feb 08 18:51:51 2016 UTC
Fingerprint (MD5): c1:46:8b:29:51:a6:4c:11:cd:81:cb:9d:7c:5e:84:d5
Fingerprint (SHA1):
01:43:bc:fa:b9:d8:30:35:ee:b6:54:dd:a4:e7:d2:11:b1:9d:bc:38
Serial number: 1005
```

12.2. Delegating Host Management

Hosts are delegated authority over other hosts through the **host-add-managedby** command. This creates a **managedby** entry. Once the **managedby** entry is created, then the host can retrieve a keytab for the host it has delegated authority over.

1. Log in as the admin user.

```
# kinit admin
```

2. Add the **managedby** entry. For example, this delegates authority over client2 to client1.

```
# ipa host-add-managedby client2.example.com --hosts=client1.example.com
```

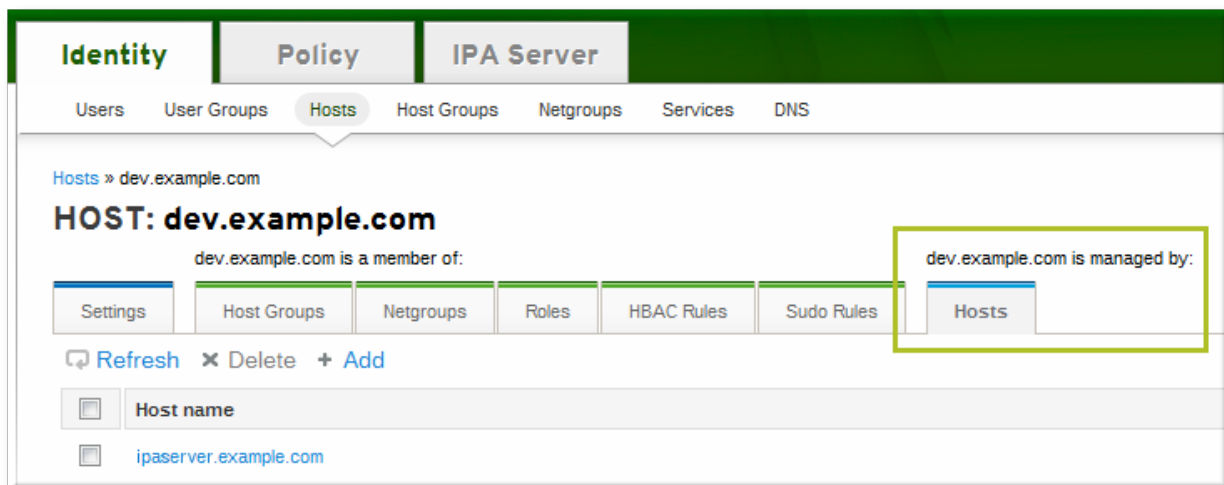
3. Obtain a ticket as the host **client1** and then retrieve a keytab for **client2**:

```
# kinit -kt /etc/krb5.keytab host/`hostname`
# ipa-getkeytab -s `hostname` -k /tmp/client2.keytab -p
host/client2.example.com
Keytab successfully retrieved and stored in: /tmp/client2.keytab
```

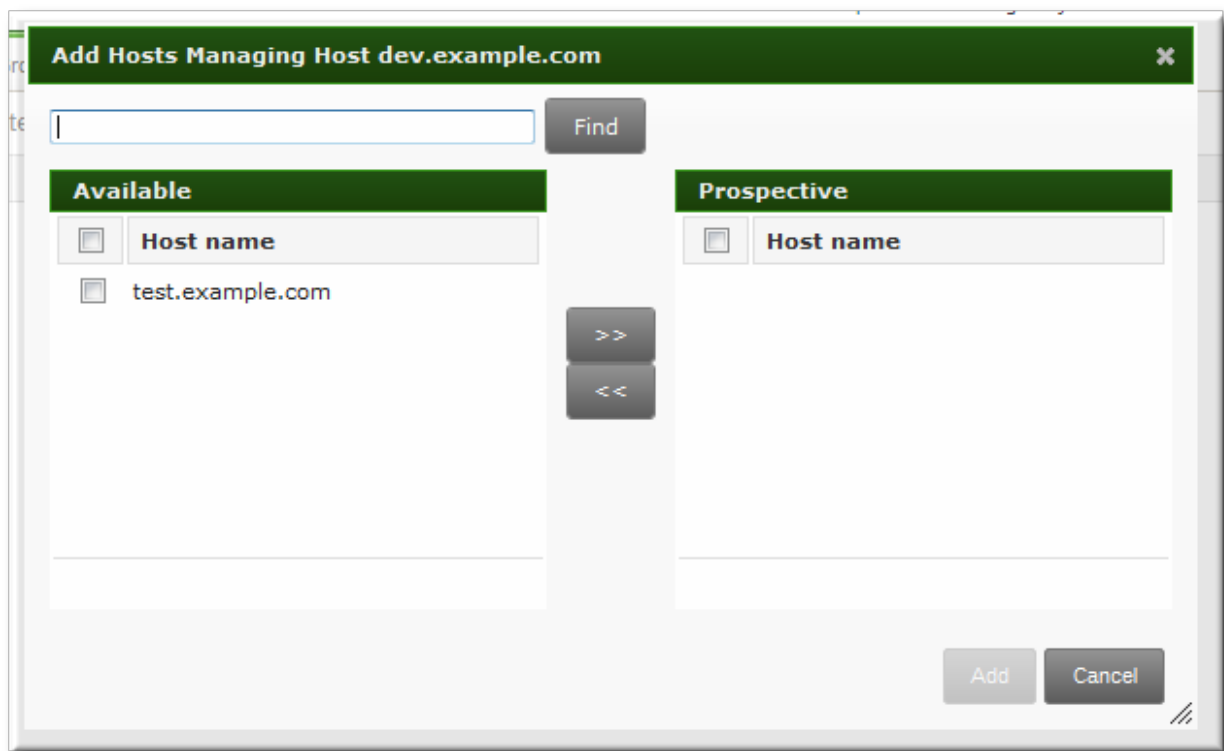
12.3. Delegating Host or Service Management in the Web UI

Each host and service entry has a configuration tab that indicates what hosts have been delegated management control over that host or service.

1. Open the **Identity** tab, and select the **Hosts** or **Services** subtab.
2. Click the name of the host or service *that you are going to grant delegated management to*.
3. Click the **Hosts** subtab on the far right of the host/service entry. This is the tab which lists hosts *which can manage* the selected host/service.



4. Click the **Add** link at the top of the list.
5. Click the checkbox by the names of the hosts to which to delegate management for the host/service. Click the right arrows button, >>, to move the hosts to the selection box.



6. Click the **Add** button to close the selection box and to save the delegation settings.

12.4. Accessing Delegated Services

For both services and hosts, if a client has delegated authority, it can obtain a keytab for that principal on the local machine. For services, this has the format *service/hostname@REALM*. For hosts, the *service* is **host**.

With **kinit**, use the **-k** option to load a keytab and the **-t** option to specify the keytab.

For example, to access a host:

```
# kinit -kt /etc/krb5.keytab host/ipa.example.com@EXAMPLE.COM
```

To access a service:

```
# kinit -kt /etc/httpd/conf/krb5.keytab http/ipa.example.com@EXAMPLE.COM
```

Chapter 13. Identity: Integrating with NIS Domains and Netgroups

Network information service (NIS) is one of the most common ways to manage identities and authentication on Unix networks. It is simple and easy to use, but it also has inherent security risks and a lack of flexibility that can make administering NIS domains problematic.

Identity Management supplies a way to integrate netgroups and other NIS data into the IdM domain, which incorporates the stronger security structure of IdM over the NIS configuration. Alternatively, administrators can simply migrate user and host identities from a NIS domain into the IdM domain.

13.1. About NIS and Identity Management

Network information service (NIS) centrally manages authentication and identity information such as users and passwords, hosts and IP addresses, and POSIX groups. This was originally called *Yellow Pages* (abbreviated YP) because of its simple focus on identity and authentication lookups.

NIS is considered too insecure for most modern network environments because it provides no host authentication mechanisms and it transmits all of its information over the network unencrypted, including password hashes. Still, while NIS has been falling out of favor with administrators, it is still actively used by many system clients. There are ways to work around those insecurities by integrating NIS with other protocols which offer enhanced security.

In Identity Management, NIS objects are integrated into IdM using the underlying LDAP directory. LDAP services offer support for NIS objects (as defined in [RFC 2307](#)), which Identity Management customizes to provide better integration with other domain identities. The NIS object is created inside the LDAP service and then a module like `nss_ldap` or `SSSD` fetches the object using an encrypted LDAP connection.

NIS entities are stored in *netgroups*. A netgroup allows nesting (groups inside groups), which standard Unix groups don't support. Also, netgroups provide a way to group hosts, which is also missing in Unix group.

NIS groups work by defining users and hosts as members of a larger domain. A netgroup sets a trio of information — host, user, domain. This is called a *triple*.

```
host, user, domain
```

A netgroup triple associates the user or the host with the domain; it does not associate the user and the host with each other. Therefore, a triple usually defines a host or a user for better clarity and management.

```
host.example.com, , nisdomain.example.com  
-, jsmith, nisdomain.example.com
```

NIS distributes more than just netgroup data. It stores information about users and passwords, groups, network data, and hosts, among other information. Identity Management can use a NIS listener to map passwords, groups, and netgroups to IdM entries.

In IdM LDAP entries, the users in a netgroup can be a single user or a group; both are identified by the *memberUser* parameter. Likewise, hosts can be either a single host or a host group; both are identified by the *memberHost* attribute.

```
dn: ipaUniqueID=d4453480-cc53-11dd-ad8b-0800200c9a66,cn=ng,cn=accounts,...
objectclass: top
objectclass: ipaAssociation
objectclass: ipaNISNetgroup
ipaUniqueID: d4453480-cc53-11dd-ad8b-0800200c9a66
cn: netgroup1
memberHost: fqdn=host1.example.com,cn=computers,cn=accounts,...
memberHost: cn=VirtGuests,cn=hostgroups,cn=accounts,...
memberUser: cn=jsmith,cn=users,cn=accounts,...
memberUser: cn=bjensen,cn=users,cn=accounts,...
memberUser: cn=Engineering,cn=groups,cn=accounts,...
nisDomainName: nisdomain.example.com
```

In Identity Management, these netgroup entries are handled using the **netgroup-*** commands, which show the basic LDAP entry:

```
# ipa netgroup-show netgroup1
Netgroup name: netgroup1
Description: my netgroup
NIS domain name: nisdomain
Member User: jsmith
Member User: bjensen
Member User: Engineering
Member Host: host1.example.com
Member Host: VirtGuests
```

When a client attempts to access the NIS netgroup, then Identity Management translates the LDAP entry into a traditional NIS map and sends it to a client over the NIS protocol (using a NIS plug-in) or it translates it into an LDAP format that is compliant with RFC 2307 or RFC 2307bis.

For more information on NIS, see the Berkeley lab manpages at <http://compute.cnr.berkeley.edu/cgi-bin/man-cgi?netgroup+4>.

13.2. Setting the NIS Port for Identity Management

The IdM server binds to its NIS services over a random port that is selected when the server starts. It sends that port assignment to the portmapper so that NIS clients know what port to use to contact the IdM server.

Administrators may need to open a firewall for NIS clients or may have other services that need to know the port number in advance and need that port number to remain the same. In that case, an administrator can specify the port to use.



NOTE

Any available port number below 1024 can be used for the NIS Plug-in setting.

The NIS configuration is in the NIS Plug-in in Identity Management's internal Directory Server instance. To specify the port:

1. Edit the plug-in configuration and add the port number as an argument. For example, to set the port to 514:


```
[root@ipaserver ~]# ldapmodify -x -D 'cn=directory manager' -w secret

dn: cn=NIS Server,cn=plugins,cn=config
changetype: modify
add: nsslapd-pluginarg0
nsslapd-pluginarg0: 514

modifying entry "cn=NIS Server,cn=plugins,cn=config"
```

- Restart the Directory Server to load the new plug-in configuration.

```
[root@ipaserver ~]# service dirsrv restart
```

13.3. Creating Netgroups

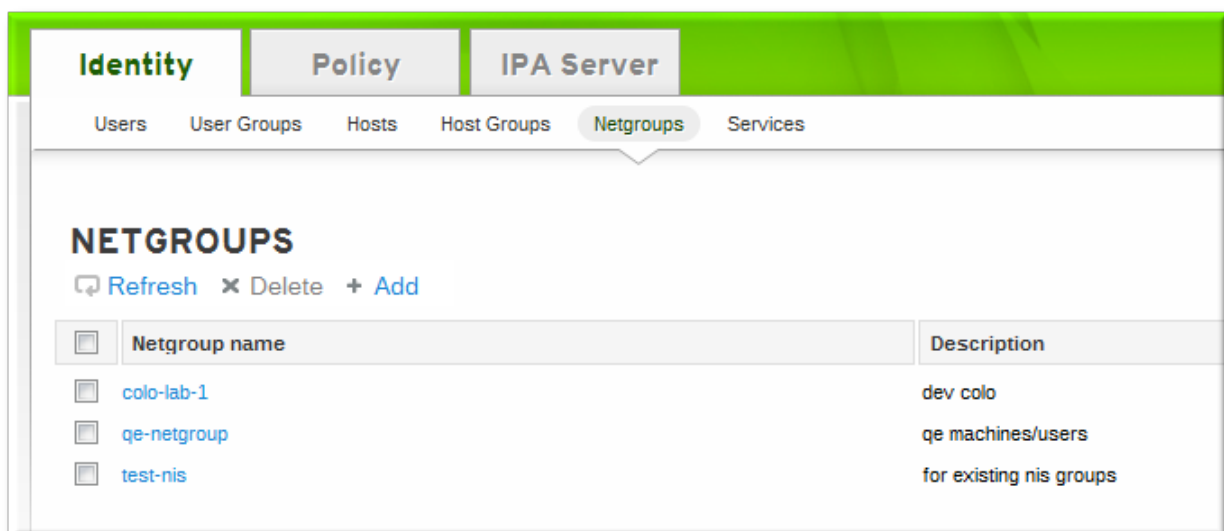
All netgroups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, IdM allows *nested groups*, where a group is a member of another group. In that case, all of the group members of the member group automatically belong to the parent group, as well.

Netgroups are added in two steps: the group itself is created, and then members are added to it.

13.3.1. Adding Netgroups

13.3.1.1. With the Web UI

- Open the **Identity** tab, and select the **Netgroups** subtab.
- Click the **Add** link at the top of the netgroups list.



- Enter both a unique name and a description for the netgroup. Both the name and description are required.

The group name is the identifier used for the netgroup in the IdM domain, and it cannot be changed after it is created. The name cannot contain spaces, but other separators like an underscore (_) are allowed.

4. Click the **Add and Edit** button to go immediately to the netgroup's edit pages.
5. Optionally, set the NIS domain for the netgroup. This defaults to the IdM domain, but it can be changed.
 - a. Click the **Settings** tab.
 - b. Enter the name of the alternate NIS domain in the **NIS domain name** field.

The **NIS domain name** field sets the domain that appears in the netgroup triple. It does *not* affect which NIS domain the Identity Management listener responds to.

6. Add members, as described in [Section 13.3.2.1, “With the Web UI”](#).

13.3.1.2. With the Command Line

New netgroups are added using the **netgroup-add** command. This adds only the group; members are added separately. Two attributes are always required: the group name and the group description. If those attributes are not given as arguments, then the script prompts for them. There is also an option to set the NIS domain name to use for the group; this defaults to the IdM domain, but it can be set to something different, depending on the network configuration.

```
$ ipa netgroup-add --desc="description" [--nisdomain=domainName] groupName
```

For example:

```
# ipa netgroup-add --desc="my new netgroup" example-netgroup
# ipa netgroup-add-member --hosts=ipa.example.com example-netgroup
# ypcat -d example.com -h ipa.example.com netgroup
(ipa.example.com, -, example.com)
```



NOTE

The **--nisdomain** option sets the domain that appears in the netgroup triple. It does *not* affect which NIS domain the Identity Management listener responds to.

13.3.2. Adding Netgroup Members



NOTE

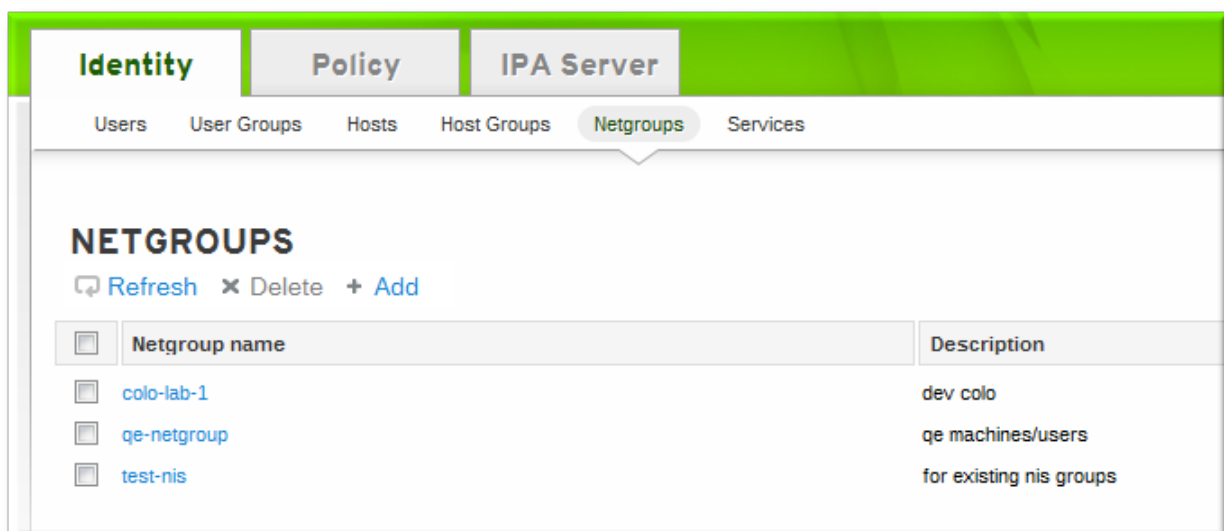
Netgroups can contain user groups, host groups, and other netgroups as their members. These are *nested* groups.

It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

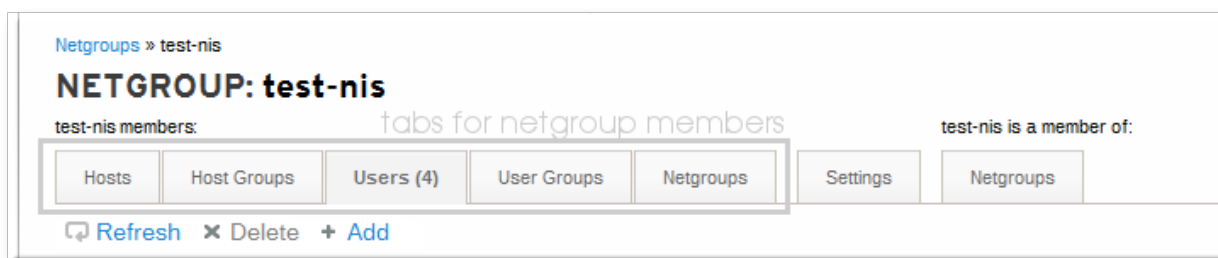
When creating nested groups, be careful not to create *recursive* groups. For example, if GroupA is a member of GroupB, do not add GroupB as a member of GroupA. Recursive groups are not supported and can cause unpredictable behavior.

13.3.2.1. With the Web UI

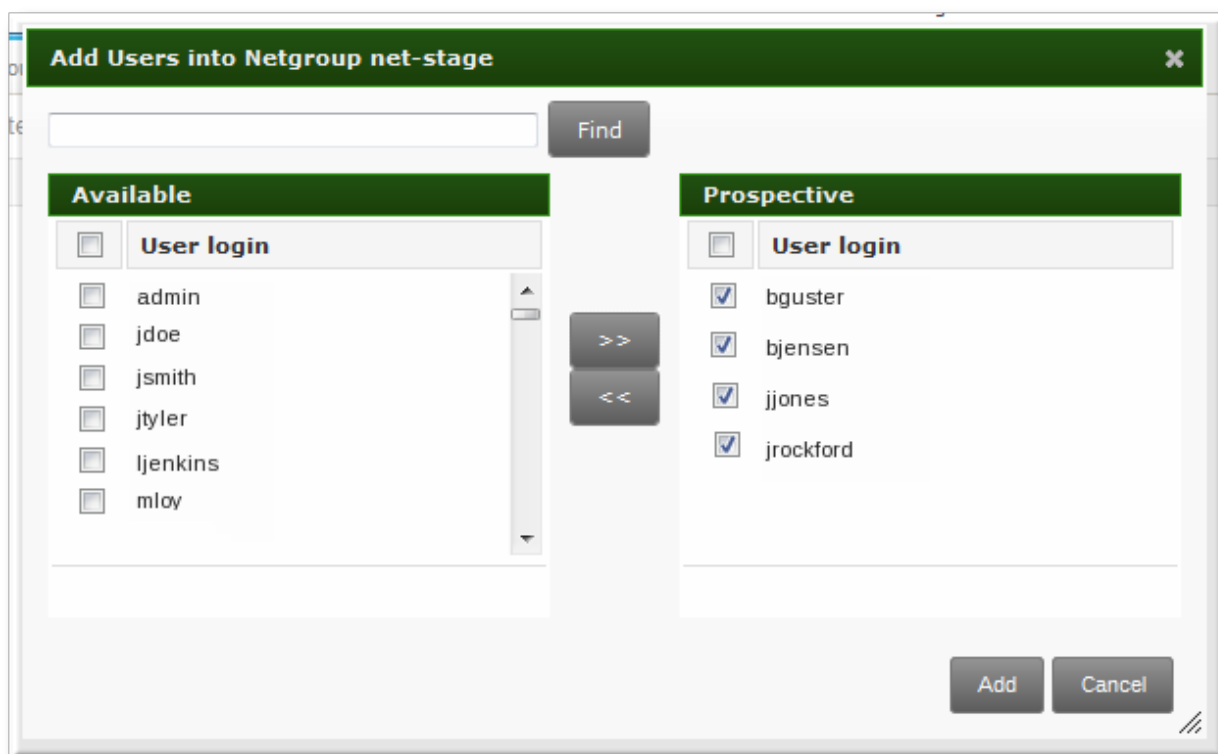
1. Open the **Identity** tab, and select the **Netgroups** subtab.
2. Click the name of the netgroup to which to add members.



3. Select the tab for the type of netgroup member to add. Netgroups can have users, user groups, hosts, host groups, and other netgroups as members.
4. Click the **Add** link at the top of the task area.



- Click the checkbox by the names of the users to add, and click the right arrows button, >>, to move the names to the selection box.



- Click the **Add** button.

13.3.2.2. With the Command Line

Once the group is configured, begin adding netgroup members with the **netgroup-add-member** command. Users, groups, hosts, host groups, and other netgroups can all be added to the netgroup entry. The entry name of the NIS group being edited usually comes at the end of the command:

```
# ipa netgroup-add-member --users=users --groups=groups --hosts=hosts --
hostgroups=hostGroups --netgroups=netgroups groupName
```

To set more than one member, use a comma-separated list with the option. For example, this sets two users and two hosts with the other configuration:

```
# ipa netgroup-add-member --users=jsmith,bjensen --groups=ITadmin --
hosts=host1.example.com,host2.example.com --hostgroups=EngDev --netgroups=nisgroup2
example-group
```

13.4. Exposing Automount Maps to NIS Clients

When the NIS service is enabled on a system, the IdM server is automatically configured to set the NIS domain to the IdM domain's name, and to include IdM users, groups, and netgroups as passwd, group,

and netgroup maps in the NIS domain.

If any automount maps are already defined, these maps need to be manually added to the NIS configuration in Identity Management for them to be exposed to NIS clients. The NIS server is managed by a special plug-in entry in the IdM LDAP directory; this is a container entry, and each NIS domain and map used by the NIS server is configured as a child entry beneath that container. The NIS domain entry must have the name of the NIS domain, the name of the NIS map, how to find the directory entries to use as the NIS map's contents, and which attributes to use as the NIS map's key and value. Most of these settings will be the same for every map.

The IdM server stores the automount maps, grouped by automount location, in the **cn=automount** branch of the IdM directory tree.

The NIS domain and map is added using LDAP tools, like **ldapadd**, and editing the directory directly. For example, this adds an automount map that is named **auto.example** in a location named **default** and for a server named **nissERVER**:

```
ldapadd -h nissERVER.example.com -x -D "cn=Directory Manager" -w secret

dn: nis-domain=example.com+nis-map=auto.example,cn=NIS Server,cn=plugins,cn=config
objectClass: extensibleObject
nis-domain: example.com
nis-map: auto.example
nis-filter: (objectclass=automount)
nis-key-format: %{automountKey}
nis-value-format: %{automountInformation}
nis-base: automountmapname=auto.example,cn=default,cn=automount,dc=example,dc=com
```

A similar add operation needs to be run for every map that is configured.

13.5. Migrating from NIS to IdM

There is no direct migration path from NIS to Identity Management. This is a manual process with three major steps: setting up netgroup entries in IdM, exporting the existing data from NIS, and importing that data into IdM. There are several options for how to set up the IdM environment and how to export data; the best option depends on the type of data and the overall network environment that you have.

13.5.1. Preparing Netgroup Entries in IdM

The first step is to identify what kinds of identities are being managed by NIS. Frequently, a NIS server is used for either user entries or host entries, but not for both, which can simplify the data migration process.

For user entries

Determine what applications are using the user information in the NIS server. While some clients (like **sudo**) require NIS netgroups, many clients can use Unix groups instead. If no netgroups are required, then simply create corresponding user accounts in IdM and delete the netgroups entirely. Otherwise, create the user entries in IdM and then create an IdM-managed netgroup and add those users as members. This is described in [Section 13.3, “Creating Netgroups”](#).

For host entries

Whenever a host group is created in IdM, a corresponding shadow NIS group is automatically created. These netgroups can then be managed using the **ipa-host-net-manage** command.

For a direct conversion

It may be necessary to have an exact conversion, with every NIS user and host having an exact corresponding entry in IdM. In that case, each entry can be created using the original NIS names:

1. Create an entry for every user referenced in a netgroup.
2. Create an entry for every host referenced in a netgroup.
3. Create a netgroup with the same name as the original netgroup.
4. Add the users and hosts as direct members of the netgroup. Alternatively, put add the users and hosts into IdM groups or other netgroups, and then add those groups as members to the netgroup.

13.5.2. Enabling the NIS Listener in Identity Management

The IdM Directory Server can function as a limited NIS server. The **slapi-nis** plug-in sets up a special NIS listener that receives incoming NIS requests and manages the NIS maps within the Directory Server. Identity Management uses three NIS maps:

- passwd
- group
- netgroup

Using IdM as an intermediate NIS server offers a reasonable way to handle NIS requests while migrating NIS clients and data.

The **slapi-nis** plug-in is not enabled by default. To enable NIS for Identity Management:

1. Obtain new Kerberos credentials as an IdM admin user.

```
[root@ipaserver ~]# kinit admin
```

2. Enable the NIS listener and compatibility plug-ins:

```
[root@ipaserver ~]# ipa-nis-manage enable
[root@ipaserver ~]# ipa-compat-manage enable
```

3. Restart the DNS and Directory Server service:

```
[root@server ~]# service rpcbind restart
[root@server ~]# service dirsrv restart
```

13.5.3. Exporting and Importing the Existing NIS Data

NIS can contain information for users, groups, DNS and hosts, netgroups, and automount maps. Any of these entry types can be migrated over to the IdM server.

Migration is performed by exporting the data using **ypcat** and then looping through that output and creating the IdM entries with the corresponding **ipa *-add** commands. While this could be done manually, it is easiest to script it. These examples use a shell script.

13.5.3.1. Importing User Entries

The **/etc/passwd** file contains all of the NIS user information. These entries can be used to create IdM user accounts with UID, GID, gecos, shell, home directory, and name attributes that mirror the NIS entries.

For example, this is **nis-user.sh**:

```
#!/bin/sh
# 1 is the nis domain, 2 is the nis master server
ypcat -d $1 -h $2 passwd > /dev/shm/nis-map.passwd 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.passwd); do
    IFS=' '
    username=$(echo $line|cut -f1 -d:)
    # Not collecting encrypted password because we need cleartext password to
    create kerberos key
    uid=$(echo $line|cut -f3 -d:)
    gid=$(echo $line|cut -f4 -d:)
    gecos=$(echo $line|cut -f5 -d:)
    homedir=$(echo $line|cut -f6 -d:)
    shell=$(echo $line|cut -f7 -d:)

    # Now create this entry
    echo passwd0rd1|ipa user-add $username --first=NIS --last=USER --password -
-gidnumber=$gid --uid=$uid --gecos=$gecos --homedir=$homedir --shell=$shell
    ipa user-show $username
done
```

This can be run for a given NIS domain:

```
[root@nis-server ~]# kinit admin
[root@nis-server ~]# ./nis-user.sh nisdomain nis-master.example.com
```



NOTE

This script does not migrate user passwords. Rather, it creates a temporary password which users are then prompted to change when they next log in.

13.5.3.2. Importing Group Entries

The **/etc/group** file contains all of the NIS group information. These entries can be used to create IdM user group accounts with the GID, gecos, shell, home directory, and name attributes that mirror the NIS entries.

For example, this is **nis-group.sh**:

```
#!/bin/sh
# 1 is the nis domain, 2 is the nis master server
ypcat -d $1 -h $2 group > /dev/shm/nis-map.group 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.group); do
    IFS=' '
    groupname=$(echo $line|cut -f1 -d:)
    # Not collecting encrypted password because we need cleartext password to
    create kerberos key
    gid=$(echo $line|cut -f3 -d:)
    members=$(echo $line|cut -f4 -d:)

    # Now create this entry
    ipa group-add $groupname --desc=NIS_GROUP_$groupname --gid=$gid
    if [ -n "$members" ]; then
        ipa group-add-member $groupname --users=$members
    fi
    ipa group-show $groupname
done
```

This can be run for a given NIS domain:

```
[root@nis-server ~]# kinit admin
[root@nis-server ~]# ./nis-group.sh nisdomain nis-master.example.com
```

13.5.3.3. Importing Host Entries

The **/etc/hosts** file contains all of the NIS host information. These entries can be used to create IdM host accounts that mirror the NIS entries.

For example, this is **nis-hosts.sh**:


```
#!/bin/sh
# 1 is the nis domain, 2 is the nis master server
ypcat -d $1 -h $2 hosts | egrep -v "localhost|127.0.0.1" > /dev/shm/nis-map.hosts
2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.hosts); do
    IFS=' '
    ipaddress=$(echo $line|awk '{print $1}')
    hostname=$(echo $line|awk '{print $2}')
    master=$(ipa env xmlrpc_uri |tr -d '[:space:]'|cut -f3 -d:|cut -f3 -d/)
    domain=$(ipa env domain|tr -d '[:space:]'|cut -f2 -d:)
    if [ $(echo $hostname|grep "\." |wc -l) -eq 0 ]; then
        hostname=$(echo $hostname.$domain)
    fi
    zone=$(echo $hostname|cut -f2- -d.)
    if [ $(ipa dnszone-show $zone 2>/dev/null | wc -l) -eq 0 ]; then
        ipa dnszone-add --name-server=$master --admin-email=root.$master
    fi
    ptrzone=$(echo $ipaddress|awk -F. '{print $3 "." $2 "." $1 ".in-addr.arpa."}')
    if [ $(ipa dnszone-show $ptrzone 2>/dev/null|wc -l) -eq 0 ]; then
        ipa dnszone-add $ptrzone --name-server=$master --admin-
email=root.$master
    fi
    # Now create this entry
    ipa host-add $hostname --ip-address=$ipaddress
    ipa host-show $hostname
done
```

This can be run for a given NIS domain:

```
[root@nis-server ~]# kinit admin
[root@nis-server ~]# ./nis-hosts.sh nisdomain nis-master.example.com
```



NOTE

This script example does not account for special host scenarios, such as using aliases.

13.5.3.4. Importing Netgroup Entries

The `/etc/netgroup` file contains all of the NIS netgroup information. These entries can be used to create IdM netgroup accounts that mirror the NIS entries.

For example, this is `nis-netgroup.sh`:

```
#!/bin/sh
# 1 is the nis domain, 2 is the nis master server
ypcat -k -d $1 -h $2 netgroup > /dev/shm/nis-map.netgroup 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.netgroup); do
    IFS=' '
    netgroupname=$(echo $line|awk '{print $1}')
    triples=$(echo $line|sed "s/^$netgroupname //")
    echo "ipa netgroup-add $netgroupname --desc=NIS_NG_$netgroupname"
    if [ $(echo $line|grep "(,|wc -l) -gt 0 ] ; then
        echo "ipa netgroup-mod $netgroupname --hostcat=all"
    fi
    if [ $(echo $line|grep ",,|wc -l) -gt 0 ] ; then
        echo "ipa netgroup-mod $netgroupname --usercat=all"
    fi
    for triple in $triples; do
        triple=$(echo $triple|sed -e 's/-//g' -e 's/(///' -e 's/)/\\')
        if [ $(echo $triple|grep ",.*,"|wc -l) -gt 0 ] ; then
            hostname=$(echo $triple|cut -f1 -d,)
            username=$(echo $triple|cut -f2 -d,)
            domain=$(echo $triple|cut -f3 -d,)
            hosts=""; users=""; doms="";
            [ -n "$hostname" ] && hosts="--hosts=$hostname"
            [ -n "$username" ] && users="--users=$username"
            [ -n "$domain" ] && doms="--nisdomain=$domain"
            echo "ipa netgroup-add-member $hosts $users $doms"
        else
            netgroup=$triple
            echo "ipa netgroup-add $netgroup --desc=NIS_NG_$netgroup"
        fi
    done
done
```

As explained briefly in [Section 13.1, “About NIS and Identity Management”](#), NIS entries exist in a set of three values, called a triple. The triple is *host,user,domain*, but not every component is required; commonly, a triple only defines a host and domain or user and domain. The way this script is written, the **ipa netgroup-add-member** command always adds a host, user, and domain triple to the netgroup.

```
if [ $(echo $triple|grep ",.*,"|wc -l) -gt 0 ] ; then
    hostname=$(echo $triple|cut -f1 -d,)
    username=$(echo $triple|cut -f2 -d,)
    domain=$(echo $triple|cut -f3 -d,)
    hosts=""; users=""; doms="";
    [ -n "$hostname" ] && hosts="--hosts=$hostname"
    [ -n "$username" ] && users="--users=$username"
    [ -n "$domain" ] && doms="--nisdomain=$domain"
    echo "ipa netgroup-add-member $hosts $users $doms"
```

Any missing element is added as a blank, so the triple is properly migrated. For example, for the triple **server,,domain** the options with the member add command are **--hosts=server --users="" --nisdomain=domain**.

This can be run for a given NIS domain by specifying the NIS domain and NIS server:

```
[root@nis-server ~]# kinit admin
[root@nis-server ~]# ./nis-hosts.sh nisdomain nis-master.example.com
```

13.5.3.5. Importing Automount Maps

Automount maps are actually a series of nested and inter-related entries that define the location (the parent entry), and then associated keys and maps.

While the data are the same in the NIS and IdM entries, the way that data are defined is different. The NIS information is exported and then used to construct an LDAP entry for the automount location and associated map; it then creates an entry for every configured key for the map.

Unlike the other NIS migration script examples, this script takes options to create an automount location and a map name, along with the migrated NIS domain and server.

```
#!/bin/sh
# 1 is for the automount entry in ipa

ipa automountlocation-add $1

# 2 is the nis domain, 3 is the nis master server, 4 is the map name
ypcat -k -d $2 -h $3 $4 > /dev/shm/nis-map.$4 2>&1

ipa automountmap-add $1 $4

basedn=$(ipa env basedn|tr -d '[:space:]'|cut -f2 -d:)
cat > /tmp/amap.ldif <<EOF
dn: nis-domain=nisdomain.example.com+nis-map=$4,cn=NIS Server,cn=plugins,cn=config
objectClass: extensibleObject
nis-domain: $3
nis-map: $4
nis-base: automountmapname=$4,cn=nis,cn=automount,$basedn
nis-filter: (objectclass=*)
nis-key-format: %{automountKey}
nis-value-format: %{automountInformation}
EOF
ldapadd -x -h $3 -D "cn=directory manager" -w secret -f /tmp/amap.ldif

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.$4); do
    IFS=" "
    key=$(echo "$line" | awk '{print $1}')
    info=$(echo "$line" | sed -e "s#^$key[ \t]*##")
    ipa automountkey-add nis $4 --key="$key" --info="$info"
done
```

This can be run for a given NIS domain:

```
[root@nis-server ~]# kinit admin
[root@nis-server ~]# ./nis-hosts.sh location nisdomain nis-master.example.com map
```

13.5.4. Setting Weak Password Encryption for NIS User Authentication to IdM

A NIS server can handle CRYPT password hashes. Once an existing NIS server is migrated to IdM (and its underlying LDAP database), it may still be necessary to preserve the NIS-supported CRYPT passwords. However, the LDAP server does not use CRYPT hashes by default. It uses SSHA or SSHA-256. If the 389 Directory Server password hash is not changed, then NIS users cannot authenticate to

the IdM domain, and **kinit** fails with password failures.

To set the underlying 389 Directory Server to use CRYPT as the password hash, change the **passwordStorageScheme** attribute using **ldapmodify**:

```
[root@server ~]# ldapmodify -D "cn=directory server" -w secret -p 389 -h  
ipaserver.example.com
```

```
dn: cn=config  
changetype: modify  
replace: passwordStorageScheme  
passwordStorageScheme: crypt
```



NOTE

Changing the password storage scheme only applies the scheme to new passwords; it does not retroactively change the encryption method used for existing passwords.

If weak crypto is required for password hashes, it is better to change the setting as early as possible so that more user passwords use the weaker password hash.

Chapter 14. Identity: Integrating with Active Directory Through Cross-Realm Kerberos Trusts (TECH PREVIEW)

Kerberos allows the configuration of *trusted realms*. Each realm has its own resources and users, yet the trust relationship allows users of any trusted realm to obtain tickets and connect to machines or services in a peer realm as if they were members of that peer realm.

Because of differences in the way that Windows and Linux domains implement LDAP services, DNS management, and even Kerberos realms, it is difficult to establish a direct trust between Active Directory and Linux domains manually. A trust relationship using IdM centrally defines and establishes the Kerberos trust and DNS mappings so that Active Directory users can access Linux hosts and services completely transparently, using one set of credentials.

14.1. The Meaning of "Trust"

Kerberos has the ability to create a relationship between two otherwise separate realms. This is called a *cross-realm trust*. This is described in some detail in [Managing Single Sign-On and Smart Cards](#). These realms create a shared ticket and key so a member of one realm is perceived as a member of both realms. One realm *trusts* another.

14.1.1. How Trust Works: Transparency Between Kerberos and DNS Realms

Both Active Directory and Identity Management manage a variety of core services: Kerberos, LDAP, DNS, certificate services. For these two disparate domains to be integrated transparently, all of these core services need to be able to interact cleanly with one another.

Those services can be broken into two major points of interaction: a Kerberos realm and a DNS domain. Certificate services, LDAP entries, and other services can be managed independently for Active Directory and IdM. The place where they interact is where identities need to be authenticated (Kerberos) and a mechanism to route queries between domains (DNS).

14.1.1.1. Components Involved in Trusts

IdM cross-realm trusts leverage four primary components:

- Active Directory
- Samba, to perform identity lookups on Active Directory and retrieve user and group security identifiers (SIDs) for authorization information
- SSSD, to cache user, group, and ticket information for users and to map Kerberos and DNS domains
- Identity Management

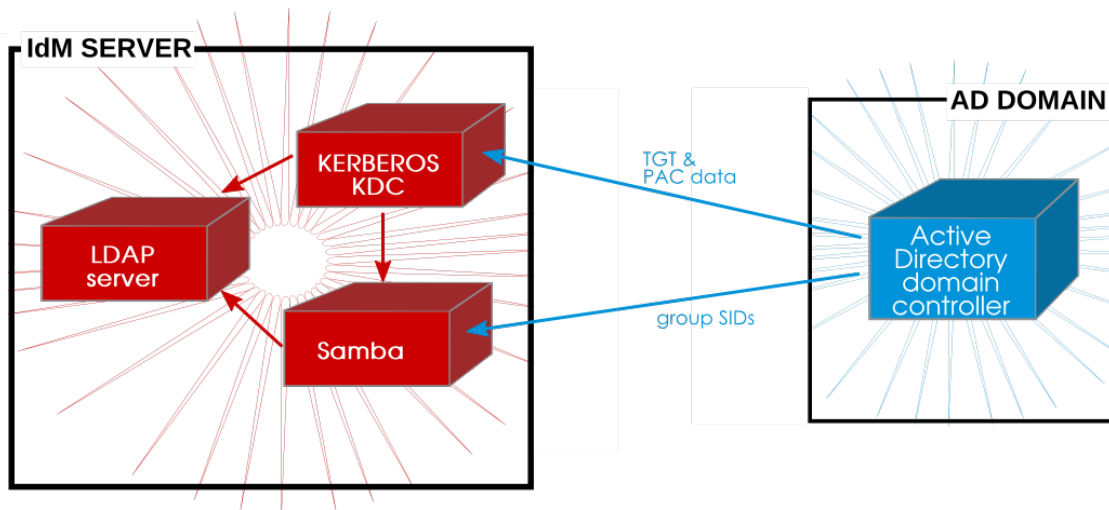


Figure 14.1. Applications and Services for Trust

14.1.1.2. Active Directory and Identity Management Directories

One of the most common backends for user identities is Active Directory, and many environments — even primarily Linux or heterogenous environments — rely on Active Directory for user management. In many environments, however, that means that an entirely different set of users must be defined to access Linux systems.

Trusts allows a natural division of labor in an IT environment between user administration (in Active Directory) and Linux or data center management (through IdM). All user accounts can be stored in Active Directory, without needing to recreate user accounts on Linux systems, while all Linux systems can still be centrally managed using native Linux tools.

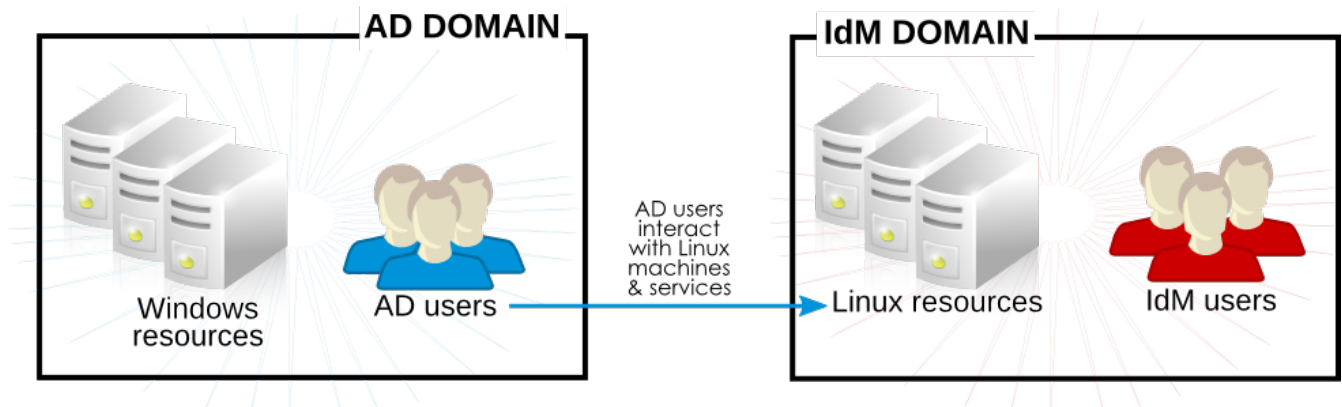


Figure 14.2. Trust Direction: Active Directory Users to Linux Resources

Trusts, then, are essentially unidirectional. Active Directory users can access IdM resources and services, but IdM users cannot access Active Directory resources. Trust allows Windows administrators and users to be able to access and manage Linux resources [5].

A trust relationship is established between a single Active Directory domain and a single IdM domain.

**NOTE**

No Windows machine can be a client of the IdM domain in a trust environment. All Windows machines must be in the Active Directory domain.

A relationship is established between the Active Directory environment and the IdM environment through a *trust agreement*, which identifies the involved domains and the settings for the trust environment.

14.1.1.3. DNS Domains

Both Active Directory and Identity Management can define DNS services, and those DNS domains must interact cleanly with each other. There are two potential DNS configurations:

- The DNS domains can be independent.
- Identity Management can be configured as a subdomain of Active Directory.

In both cases, the different domains forward requests to each other as necessary and maintain different DNS namespaces. It is just a matter of defining how they recognize each other for forwarding queries.

**IMPORTANT**

Both Active Directory and Identity Management must be configured with integrated DNS servers.

Separate DNS Domains

In this case, there are two entirely different namespaces, such as **ipaexample.com** and **adexample.com**. For these domains to communicate, they must be configured as conditional forwarders of each other's domain.

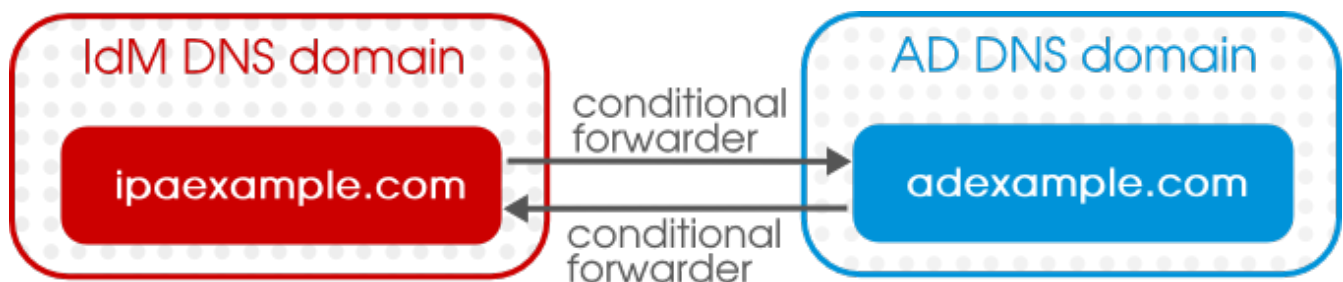


Figure 14.3. Trust with Separate DNS Domains

Identity Management as a Subdomain

In this case, Identity Management is a namespace within the larger Active Directory space, such as **linux.example.com** and **example.com**. IdM can be configured to send all requests to the Active Directory domain (a forward-only policy) or it can send queries first to Active Directory and then attempt to resolve them itself (a forward-first policy).

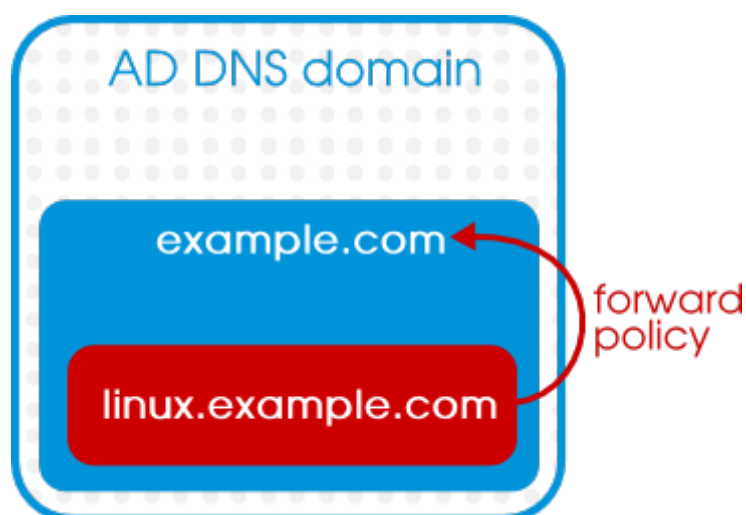


Figure 14.4. Trust with IdM as a DNS Subdaomin of Active Directory

14.1.1.4. Kerberos Realms, Authentication, and Authorization

Group information in Active Directory is stored in a list of identifiers in each Kerberos ticket for Active Directory users in a special dataset called *privileged access certificates* or MS-PAC. The group information in the PAC has to be mapped to the Active Directory groups and then to the corresponding IdM groups to help determine access. A PAC is essentially an account usability extension.

Understanding the group mapping for trusts can help clarify how groups should be structured in trust environments.

Microsoft uses a special authorization structure called *privileged access certificates* or MS-PAC. A PAC is embedded in a Kerberos ticket as a way of identifying the entity to other Windows clients and servers in the Windows domain.

On IdM resources, if an Active Directory user requests a ticket for a service, then IdM forwards the request to Active Directory to retrieve the user information. The PAC information sent back by Active Directory is embedded in the Kerberos ticket.



NOTE

All Kerberos communication for both Active Directory and IdM for trusts uses GSS-API.

IdM (through SSSD, as an IdM client), extracts the Active Directory group *security identifiers* (SIDs) from the PAC. IdM then compares the Active Directory SIDs in the PAC to the group SIDs configured as members in IdM groups. If the Active Directory group is a member of an IdM group, then the IdM group SID is added to the PAC, and the Kerberos ticket is updated with the new PAC.

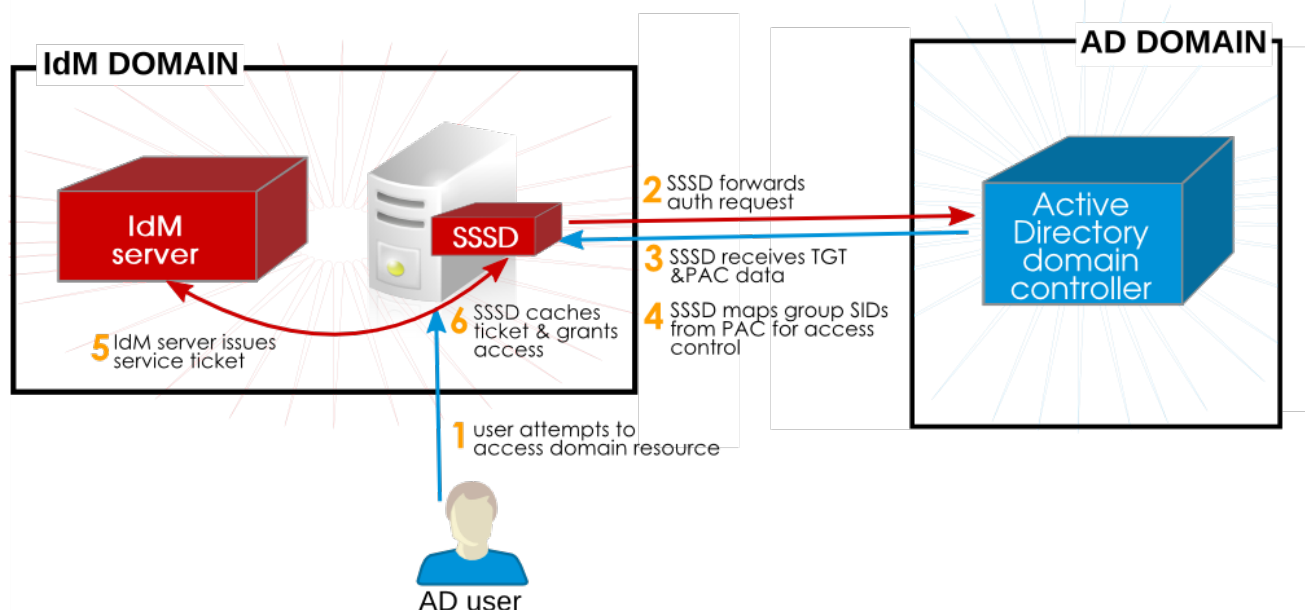


Figure 14.5. IdM, SSSD, and Active Directory

That new ticket is then used to generate a service ticket for the user, and the user is granted access to the IdM-hosted services, according to their access rules. Additionally, the IdM group information in the SSSD user cache is updated to include the mapped IdM groups for the Active Directory user.

SSSD stores multiple TGTs and tickets for each user, as new services are accessed.

A simpler way of saying this is that Active Directory supplies a list of groups for each user, based on an identifier for the group. IdM compares that list of Active Directory groups to memberships in IdM groups (where each group member is identified by that SID, rather than by a name or DN). If the Active Directory groups to which the user belongs are known to the IdM domain, then the user is recognized by the IdM domain.

The crucial factor to realize in this is that Active Directory users are recognized to the IdM domain not by their Active Directory user entry, but by their Active Directory group memberships. In a sense, Active Directory *users* are not trusted by the IdM domain — Active Directory *groups* are.

But this method of mapping Active Directory group SIDs to IdM group members means that group structure in IdM is important. Active Directory groups have different attributes than Linux groups and, therefore, different attributes than IdM groups. Most critically, Active Directory groups are not POSIX groups, while IdM groups are.

IdM has introduced an intermediary, non-POSIX group type, *external groups*, which allow entities outside IdM or a Linux system to be added as member. That external group can then be added to a standard IdM (POSIX) group as a member.

When Active Directory groups are added to an IdM group, they can be identified by their SID or by name, in the formats `DOMAIN\group_name` or `group_name@domain`. IdM then resolves the group name to the SID and stores the SID as the group member entry, to be compared to any offered user PAC.

Actually configuring groups for Active Directory users is described in [Section 14.5, “Creating IdM Groups for Active Directory Users”](#).



IMPORTANT

Both Active Directory and Identity Management must be configured with integrated certificate services.

All sessions in a trust environment are ultimately secured with Kerberos tickets, but users have different login options:

- ▶ Ticket-based authentication through **kinit**
- ▶ Simple username/password authentication that is negotiated into a ticket
- ▶ Passwordless authentication that is negotiated into a ticket (depending on the Kerberos client configuration)

14.1.2. Trust in Contrast to Synchronization

Trusts and synchronization are fundamentally different approaches to integrating an IdM domain and Active Directory domain. The structure of trust domains (outlined in [Section 14.1.1, “How Trust Works: Transparency Between Kerberos and DNS Realms”](#)), the complexity of group assignments, the location of user and group entries, and other factors all influence which solution is most effective for a given environment.

Synchronization has a certain simplicity in its overall topology and setup, and it has a relatively small administrative footprint. However, it is much more limited in how it handles directory data, its ability to map entries, its overall performance, and its security.

Table 14.1. Positives and Negatives of Using Sync

Positives of Sync	Negatives of Sync
▶ Simple setup procedures ▶ Few rules about the Active Directory configuration, including being agnostic about DNS and Kerberos domains ▶ Users and groups can originate in both Active Directory and IdM domains ▶ Active Directory users can function as IdM users, including as administrators ▶ Windows machines can be added as clients to the IdM domain	▶ Limited set of synchronized attributes and problematic data mapping ▶ Potential data inconsistency between Active Directory and IdM entries for the same user ▶ Different LDAP versions, synchronization protocols, and other technology differences ▶ Delays in relaying updates between directories ▶ Decreased performance ▶ Security implications of syncing passwords — or administrative complexity for maintaining different passwords for the same user account in difference locations

The initial environment configuration for trusts is much more complex than synchronization, but it has advantages in simplifying single sign-on to systems, web applications, or terminals; not requiring additional directory administration; and preserving data integrity.

Table 14.2. Positives and Negatives of Using Trusts

Positives of Trusts	Negatives of Trusts
► Pulls in authentication, group, and authorization data automatically when a user logs in ► Allows true single sign-on, with a single stored password and without having to synchronize passwords between directories ► Caches data in a local database ► Allows users to be entirely defined in a single domain, yet have access to multiple domains ► Can be configured without having to know or restructure the underlying directory trees ► Allows Kerberos authentication, username/password authentication (which generates a Kerberos ticket), or passwordless logins	► Has very specific DNS configuration requirements ► Can potentially have long wait times to retrieve data when a user initially logs in ► Prefers that users be located in a single directory and resources in another ► Windows machines <i>cannot</i> be clients of the IdM domain

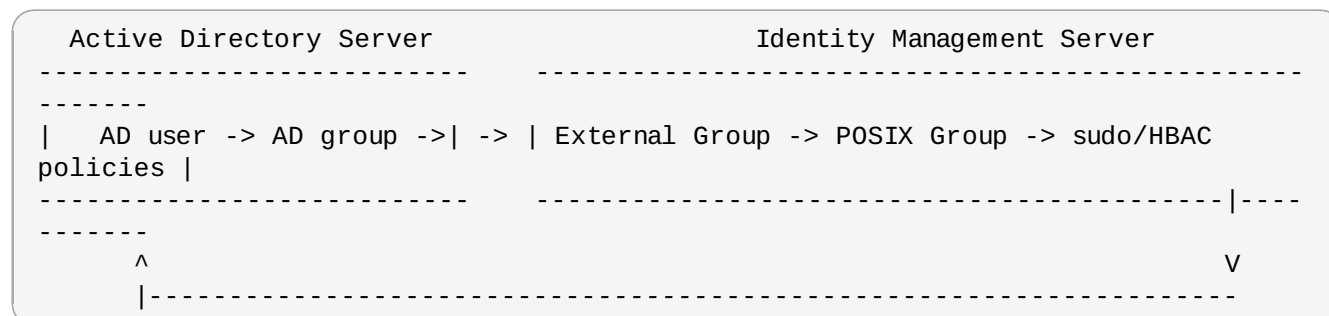
**NOTE**

There is no clear migration path from using synchronization to using trusts because the entries already exist in the backend IdM LDAP directory. This means that Active Directory user entries (or all user entries, if IdM users are also synced) are duplicated, which can lead to odd behavior with logins, group associations, and lookups.

14.1.3. Active Directory Users and IdM Features: sudo and Host-Based Access Control Policies

Active Directory users *cannot* be added directly to an IdM group as a member. This means that policies and configuration in IdM which rely on group associations — such as host-based access control rules and sudo policies — must be configured in a kind of daisy-chain.

The Active Directory user is added to an Active Directory group, then that Active Directory group is added to an IdM external group, which is added as a member to a POSIX group. The sudo, host-based access controls, and other policies are applied against that POSIX group and, ultimately, through nesting memberships applied to the Active Directory user when accessing IdM domain resources.



**NOTE**

Testing tools such as **hbactest** will not work with trusted users because the trusted user group associations are resolved dynamically, not stored in the IdM directory.

14.1.4. Potential Issues with Group Mapping and SIDs

Losing Kerberos Tickets

Running any command to obtain a SID from the Samba service — such as **net getlocalsid** or **net getdomainsid** — kills any existing admin ticket in the Kerberos cache.

Cannot Verify Group Membership for Users

There is no way to verify that a specific trusted user is associated with a specific IdM group, external or POSIX.

Cannot Display (Remote) Active Directory Group Memberships for an Active Directory User

For Linux system users, local group associations can be shown for a user using the **id** command. However, Active Directory group memberships are not displayed with **id** for Active Directory users, even though they are with Samba tools.

The **wbinfo** command can be used to obtain a SID for an Active Directory user and then to display groups associated with that SID.

```
[root@ipaserver ~]# wbinfo -n ADDDOMAIN\jsmith
S-1-5-21-1689615952-3716327440-3249090444-1104 SID_USER (1)

[root@ipaserver ~]# wbinfo --user-domgroups=S-1-5-21-1689615952-3716327440-
3249090444-1104
S-1-5-21-1689615952-3716327440-3249090444-513
S-1-5-21-1689615952-3716327440-3249090444-1106
```

The same query using **id** shows only the user information, not the Active Directory group membership information.

```
[root@ipaserver ~]# id ADDDOMAIN\jsmith
uid=1921801104(jsmith@adexample.com) gid=1921801104(jsmith@adexample.com)
groups=1921801104(jsmith@adexample.com)
```

**TIP**

To work around this, ssh into an IdM client machine as the given Active Directory user. After the first successful login, the Active Directory group memberships are detected and returned in the **id** search.

```
[root@ipaserver ~]# id ADDDOMAIN\jsmith
uid=1921801107(jsmith@adexample.com) gid=1921801107(jsmith@adexample.com)
groups=1921801107(jsmith@adexample.com),129600004(ad_users),1921800513(domain
 users@adexample.com)
```

14.1.5. Active Directory Users and IdM Administration

Trust relationships are unidirectional. Active Directory users exist only within the Active Directory domain and are limited to what resources within the IdM domain they can access. **Active Directory users are not administrators for IdM because they do not exist within IdM.**

Active Directory users, then, cannot use any IdM administrative tools, including the web UI and command-line tools.

14.2. Environment and Machine Requirements to Set Up Trusts

Make sure that both the Active Directory and IdM servers, machines, and environments meet the requirements and settings in this section *before* configuring a trust agreement.

14.2.1. Supported Windows Platforms

Trust relationships can be configured with these Windows server versions:

- Windows Server 2008 R2 (32-bit)
- Windows Server 2008 R2 (64-bit)

14.2.2. Domain and Realm Names

The IdM DNS domain name and Kerberos realm name *must* be different than the Active Directory DNS domain name and Kerberos realm name.

14.2.3. NetBIOS Names

The NetBIOS name is the far-left component of the domain name. For example, if the domain is *linux.example.com*, the NetBIOS name is *linux*, while if the domain name is simply *example.com*, it is *example*. The NetBIOS name is critical for identifying the Active Directory domain and, if the IdM domain is within a subdomain of Active Directory DNS, for identifying the IdM domain and services.

The IdM domain and Active Directory domain must have different NetBIOS names.

14.2.4. Integrated DNS

Both the Active Directory server and the IdM server must be configured to run their own respective DNS services.

14.2.5. Firewalls and Ports

Required Ports

For a trust relationship, the Active Directory server and IdM server must have almost all of the required system ports open that are required for an IdM server installation, **with the exception of the LDAP ports**.

Table 14.3. IdM Ports

Service	Ports	Type
HTTP/HTTPS	80 443	TCP
Kerberos	88 464	TCP and UDP
DNS	53	TCP and UDP
NTP	123	UDP



IMPORTANT

The IdM backend LDAP server *must not be reachable* by the Active Directory domain controller. The associated ports — 389 and 636 — on the IdM server host must be shut down for the Active Directory domain controller.

Starting iptables at Boot Time

Configure the **iptables** service to start when the system boots:

```
[root@ipaserver ~]# chkconfig iptables on
```

Setting iptables Configuration

The **iptables** configuration needs to allow access to the required IdM ports and reject access to the IdM LDAP ports. The order of the rules is important. Active Directory-based requests to LDAP ports must be blocked first (based on the Active Directory server IP address), then there must be connections allowed to all IdM TCP and UDP ports.

1. Open the **iptables** configuration file.

```
[root@ipaserver ~]# vim /etc/sysconfig/iptables
```

2. Add the rule to restrict access to LDAP ports for the Active Directory host.

```
-A INPUT -s ad_ip_address -p tcp -m multiport --dports 389,636 -m state --state NEW,ESTABLISHED -j REJECT
```

3. Make sure that there lines to allow access to the TCP and UDP ports required by IdM.

```
-A INPUT -p tcp -m multiport --dports 80,443,389,636,88,464,53,138,139,445
-m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m multiport --dports 88,464,53,123,138,139,389,445 -
m state --state NEW,ESTABLISHED -j ACCEPT
```

4. Save the file.

5. Restart the **iptables** service:

```
[root@ipaserver]# service iptables restart
```

Example 14.1. Example iptables Configuration File

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -s ad_ip_address -p tcp -m multiport --dports 389,636 -m state --state
NEW,ESTABLISHED -j REJECT
-A INPUT -p tcp -m multiport --dports 80,443,389,636,88,464,53,138,139,445 -m
state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m multiport --dports 88,464,53,123,138,139,389,445 -m state -
-state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -j REJECT
-A INPUT -p tcp -j REJECT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

14.2.6. Clock Settings

Both the Active Directory server and the IdM server must have their clocks in sync.

14.2.7. Supported Username Formats

Username mapping is performed in the local SSSD client. A Python regular expression is used by SSSD to identify the username and the domain to which it belongs.

By default in SSSD, the username format is defined in the form *name@domain*. This uses the regular expression:

```
re_expression = (?P<name>[^\@]+)@?(?P<domain>[^\@]*$)
```

Active Directory can support several different kinds of name formats, however, so the **re_expression** parameter in the SSSD configuration file for IdM backends or Active Directory backends uses a more complex expression:

```
re_expression = (((?P<domain>[^\@]+)\.?(?P<name>.+))|((?P<name>[^\@]+)@(?
P<domain>.+))|(^(?P<name>[^\@]+)$))
```

This supports usernames in multiple formats:

- `username`
- `username@domain.name`
- `DOMAIN\username`



TIP

An additional SSSD parameter, **`default_domain_suffix`**, can be used to supply a default domain value for usernames. For example, if all users are in a trusted Active Directory domain of **`adexample.com`** and the identity backend is the IdM domain of **`ipa.example.com`**, the **`default_domain_suffix`** parameter can be set with the value **`adexample.com`**. All users are automatically assumed to belong to that user domain unless the domain value is explicitly given with the username.

This is explained in more detail in the [SSSD chapter of the Deployment Guide](#).

14.2.8. Trust Can Only Be Configured Once



WARNING

The **`ipa-adtrust-install`** command can only be run once. If any information is entered incorrectly — particularly the NetBIOS name for the IdM server, but also the administrative credentials or other settings — then the trust services and all IdM packages must be uninstalled and then reinstalled and rerun. It is not possible to rerun the **`ipa-adtrust-install`** command to change the settings.

14.3. Setting up Trust with IdM as a DNS Subdomain of Active Directory

1. Stop the Windows firewall service.
2. Stop **`iptables`** and **`ip6tables`** on the IdM server.

```
[root@ipaserver]# service iptables stop
```

3. Install the required trust packages, updated Samba4 packages, and LDAP-DNS packages for IdM DNS management.

```
[root@ipaserver]# yum install ipa-server "*ipa-server-trust-ad" samba4-winbind-clients bind-dyndb-ldap samba4-client
```



IMPORTANT

The Samba4 packages conflict with the default Samba3 packages on the Red Hat Enterprise Linux system. There may be dependency issues with other applications as the Samba3 packages are removed.

The **`cifs-utils`** package is removed when Samba3 is removed. This must be re-installed.


```
[root@ipaserver]# yum install cifs-utils
```

It is recommended that you remove the **samba4-winbind-krb5-locator** package to improve Kerberos performance.

```
[root@ipaserver]# yum remove samba4-winbind-krb5-locator
```

4. *For a new IdM server.* Set up the IdM server to use its own, integrated DNS services (**--setup-dns**), its own DNS domain (**-n**), and the Active Directory DNS server as a forwarder (**--forwarder**). For example:

```
[root@ipaserver]# ipa-server-install --setup-dns --
forwarder=255.255.255.255 -p secret -a secret -r IPAEXAMPLE -n
linux.adexample.com --hostname ipaserver.linux.adexample.com -U
```

ipa-server-install options are described in [Section 3.2, “About ipa-server-install”](#).

If the IdM server was set up without using Active Directory as a forwarder. If an IdM server was configured without using Active Directory as a forwarder, then the Active Directory server can be added as a conditional forwarder. This requires the IP address of the Active Directory DNS server.

```
[root@ipaserver]# ipa dnsconfig-mod --forwarder=255.255.255.255 --forward-
policy=first
```

Using a first policy means that queries are sent to the forwarder first and then to the local **named** process. Alternatively, this can be set to **only**, so that only the DNS forwarder is queried, never **named**.

5. Add the IdM domain as a subdomain entry in the Active Directory configuration and add an NS record for the IdM DNS. For this example, the IdM configuration has a NetBIOS name of **linux** (the subdomain) and the domain name is **adexample.com**.
 - a. Open the command prompt, using *Run as Administrator*.
 - b. Use the **dnscmd** command to add the A record for the IdM server, using the IdM hostname, NetBIOS name, and IP address.

```
/RecordAdd ad_domain ipa_hostname.ipa_netbios A ipa_ip_address
```

For example:

```
C:\> dnscmd 127.0.0.1 /RecordAdd adexample.com ipaserver.linux A
255.255.255.0
```

- c. Then add the NS record for the IdM server. This has the format:

```
/RecordAdd ad_domain ipa_netbios NS ipa_hostname.ipa_subdomain
```

For example:

```
C:\> dnscmd 127.0.0.1 /RecordAdd adexample.com linux NS
ipaserver.linux.adexample.com
```

6. Check the SRV records for both domains from both servers.

On the IdM server, use the **dig SRV** command to list the records for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# dig SRV _ldap._tcp.adexample.com
;; ANSWER SECTION:
_ldap._tcp.adexample.com. 600      IN      SRV     0 100 389
adserver.adexample.com.
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161

[root@ipaserver ~]# dig SRV _ldap._tcp.linux.adexample.com
;; ANSWER SECTION:
_ldap._tcp.linux.adexample.com. 86400   IN      SRV     0 100 389
ipaserver.linux.adexample.com.
;; AUTHORITY SECTION:
linux.adexample.com. 86400    IN      NS
ipaserver.linux.adexample.com.
;; ADDITIONAL SECTION:
ipaserver.linux.adexample.com. 1200    IN      A       192.168.2.158
```

On the Active Directory server, open the **nslookup** tool and check the corresponding SRV records.

```
> nslookup
> set type=srv
> _ldap._tcp.adexample.com
> _ldap._tcp.linux.adexample.com
> quit
```

7. Enable DNS lookups in the Kerberos realm for the Kerberos client.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ~]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, add or set the **dns_lookup_kdc** value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

8. Configure the IdM server to enable trust services. This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator with the **-a**. Optionally, use the **-U** argument to run the script non-interactively.

```
[root@ipaserver ~]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret -U
```

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The **smbclient** command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----
  IPC$           IPC            IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server          Comment
  -----
  Workgroup       Master
  -----
```

The **wbinfo** command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
BUILTIN : online
IPAEXAMPLE : online
```

10. *If there are existing IdM users and groups.* For existing IdM users, it is required that all users and groups have an Active Directory-style security identifier (SID). A new **ipaNTSecurityIdentifier** containing a SID can be created automatically for each entry by running a special **ipa-sidgen-task** operation on the backend LDAP directory.

If there are no existing IdM users or groups, then this step can be skipped.

```
[root@ipaserver ]# ldapmodify -x -H ldap://ipaserver.ipaexample.com:389 -D
"cn=directory manager" -w Passwd -f

dn: cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
cn: sidgen
nsslapd-basedn: dc=ipadomain,dc=com
delay: 0

adding new entry "cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config"
```

When the task completes successfully, there will be a message in the error logs that the SID generation task (**Sidgen task**) finished with a status of zero (0).

```
[root@ipaserver ]# grep "sidgen_task_thread" /var/log/dirsrv/slapd-IPALAB-
QE/errors
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c,
line 191]: Sidgen task starts ...
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c,
line 196]: Sidgen task finished [0].
```

11. Create a trust agreement for the Active Directory domain and the IdM domain. This command requires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=type ad_domain_name --admin ad_admin_username --
password
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
Realm name: adexample.com
Domain NetBIOS name: ADEXAMPLE
Domain Security Identifier: S-1-5-21-1689615952-3716327440-3249090444
Trust direction: Two-way trust
Trust type: Active Directory domain
Trust status: Established and verified
```

12. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

```
[root@ipaserver ~]# kinit jsmith
```

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ~]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ~]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@IPA.DOMAIN

Valid starting    Expires          Service principal
06/15/12 12:13:04 06/16/12 12:12:55 krbtgt/IPA.DOMAIN@IPA.DOMAIN
06/15/12 12:13:13 06/16/12 12:12:55 host/ipaserver.ipaexample.com@IPA.DOMAIN
06/15/12 12:13:23 06/16/12 12:12:55 krbtgt/AD.DOMAIN@IPA.DOMAIN
06/15/12 12:14:58 06/15/12 22:14:58 cifs/adserver.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

13. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ~]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
....
dns_lookup_kdc = true
```

- c. In the **[realms]** section, identify the IdM realm by name, and then add two **auth_to_local** lines to define the Kerberos principal name mapping. One rule should have a value of **DEFAULT**, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format *first.last@ADDOMAIN*, *username@ADDOMAIN[.something]*, or *username@addomain[.something]*; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
auth_to_local = DEFAULT
}
```

- d. Restart the KDC service.

```
[root@ipaserver ~]# service krb5kdc restart
```

14. Configure domain mapping in SSSD.

- a. Open the **/etc/sss/sss.conf**.

```
[root@ipaserver ~]# vim /etc/sss/sss.conf
```

- b. In the **[sss]** section, add **pac** to the **services** list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

- c. In the IdM domain section, add the **subdomains_provider** parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed; adding this line makes it possible to use the existing configuration.

Subdomains and SSSD are described in more detail in the [IdM provider configuration examples in the SSSD chapter of the Deployment Guide](#).

- d. Save the changes to the **sssd.conf** file.
- e. Restart SSSD.

```
[root@ipaserver ]# service sssd restart
```

15. Restart the **iptables** and **ip6tables** services on the IdM server.

```
[root@ipaserver ]# service iptables start
```

16. Restart the Windows firewall.

14.4. Setting up Trust with IdM and Active Directory in Different DNS Domains

1. Stop the Windows firewall service.
2. Stop **iptables** and **ip6tables** on the IdM server.

```
[root@ipaserver ]# service iptables stop
```

3. Install the required trust packages, updated Samba4 packages, and LDAP-DNS packages for IdM DNS management.

```
[root@ipaserver ]# yum install ipa-server "*ipa-server-trust-ad" samba4-winbind-clients bind-dyndb-ldap samba4-client
```



IMPORTANT

The Samba4 packages conflict with the default Samba3 packages on the Red Hat Enterprise Linux system. There may be dependency issues with other applications as the Samba3 packages are removed.

The **cifs-utils** package is removed when Samba3 is removed. This must be re-installed.

```
[root@ipaserver ]# yum install cifs-utils
```

It is recommended that you remove the **samba4-winbind-krb5-locator** package to improve Kerberos performance.

```
[root@ipaserver ]# yum remove samba4-winbind-krb5-locator
```

4. *For a new IdM server.* Set up the IdM server to use its own, integrated DNS services (**--setup-dns**), its own DNS domain (**-n**), and the Active Directory DNS server as a forwarder (**--forwarder**). For example:

```
[root@ipaserver ]# ipa-server-install --setup-dns --forwarder=ad-dns.adserver.example.com -p secret -a secret -r IPAEXAMPLE -n ipaexample.com --hostname ipaserver.ipaexample.com -U
```

ipa-server-install options are described in [Section 3.2, “About ipa-server-install”](#).

If the IdM server was set up without using Active Directory as a forwarder. If an IdM server was configured without using Active Directory as a forwarder, then the Active Directory server can be added as a conditional forwarder. This requires the IP address of the Active Directory DNS server.

```
[root@ipaserver]# ipa dnsconfig-mod --forwarder=255.255.255.255 --forward-policy=first
```

Using a first policy means that queries are sent to the forwarder first and then to the local **named** process. Alternatively, this can be set to only, so that only the DNS forwarder is queried, never **named**.

5. Add the IdM server as a conditional forwarder in the Active Directory DNS configuration.
 - a. Open the **Administrative Tools** menu, and select the **DNS** item.
 - b. Right-click the **Conditional Forwarders** item in the left column of the window.
 - c. Select the **New Conditional Forwarder...** button.
 - d. Enter the DNS domain name of the IdM domain and the IP address of the IdM DNS server.
 - e. Save the new forwarder.

Alternatively, use the **dnscmd** command-line utility to add the forwarder entry:

```
> dnscmd 127.0.0.1 /ZoneAdd IPAEXAMPLE.COM /Forwarder 255.255.255.0
```

6. Check the SRV records for both domains from both servers.

On the IdM server, use the **dig SRV** command to list the records for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# dig SRV _ldap._tcp.adexample.com
;; ANSWER SECTION:
_ldap._tcp.adexample.com. 600      IN      SRV     0 100 389
adserver.adexample.com.
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161

[root@ipaserver ~]# dig SRV _ldap._tcp.ipaexample.com
;; ANSWER SECTION:
_ldap._tcp.ipaexample.com. 86400   IN      SRV     0 100 389
ipaserver.ipaexample.com.
;; AUTHORITY SECTION:
ipaexample.com. 86400   IN      NS      ipaserver.ipaexample.com.
;; ADDITIONAL SECTION:
ipaserver.ipaexample.com. 1200    IN      A       192.168.2.158
```

On the Active Directory server, open the **nslookup** tool and check the corresponding SRV records.

```
> nslookup
> set type=srv
> _ldap._tcp.adexample.com
> _ldap._tcp.ipaexample.com
> quit
```

7. Enable DNS lookups in the Kerberos realm for the Kerberos client.

- a. Open the `/etc/krb5.conf` configuration file.

```
[root@ipaserver ~]# vim /etc/krb5.conf
```

- b. In the `[libdefaults]` section, add or set the `dns_lookup_kdc` value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

8. Configure the IdM server to enable trust services. This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator with the `-a`. Optionally, use the `-U` argument to run the script non-interactively.

```
[root@ipaserver ~]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret -U
```

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The `smbclient` command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----
  IPC$           IPC            IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server          Comment
  -----
  Workgroup       Master
  -----
```

The `wbinfo` command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
BUILTIN : online
IPAEXAMPLE : online
```

10. *If there are existing IdM users and groups.* For existing IdM users, it is required that all users and groups have an Active Directory-style security identifier (SID). A new ***ipaNTSecurityIdentifier*** containing a SID can be created automatically for each entry by running a special ***ipa-sidgen-task*** operation on the backend LDAP directory.

If there are no existing IdM users or groups, then this step can be skipped.


```
[root@ipaserver ]# ldapmodify -x -H ldap://ipaserver.ipaexample.com:389 -D
"cn=directory manager" -w Passwd -f

dn: cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
cn: sidgen
nsslapd-basedn: dc=ipadomain,dc=com
delay: 0

adding new entry "cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config"
```

When the task completes successfully, there will be a message in the error logs that the SID generation task (**Sidgen task**) finished with a status of zero (0).

```
[root@ipaserver ]# grep "sidgen_task_thread" /var/log/dirsrv/slapd-IPALAB-
QE/errors
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c,
line 191]: Sidgen task starts ...
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c,
line 196]: Sidgen task finished [0].
```

11. Create a trust agreement for the Active Directory domain and the IdM domain. This command requires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=type ad_domain_name --admin ad_admin_username --
password
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
  Realm name: adexample.com
  Domain NetBIOS name: ADEXAMPLE
  Domain Security Identifier: S-1-5-21-1689615952-3716327440-3249090444
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  Trust status: Established and verified
```

12. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

```
[root@ipaserver ~]# kinit jsmith
```

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@IPA.DOMAIN

Valid starting    Expires          Service principal
06/15/12 12:13:04 06/16/12 12:12:55 krbtgt/IPA.DOMAIN@IPA.DOMAIN
06/15/12 12:13:13 06/16/12 12:12:55
host/ipaserver.ipaexample.com@IPA.DOMAIN
06/15/12 12:13:23 06/16/12 12:12:55 krbtgt/AD.DOMAIN@IPA.DOMAIN
06/15/12 12:14:58 06/15/12 22:14:58 cifs/adserver.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

13. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
....
dns_lookup_kdc = true
```

- c. In the **[realms]** section, identify the IdM realm by name, and then add two **auth_to_local** lines to define the Kerberos principal name mapping. One rule should have a value of **DEFAULT**, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format *first.last@ADDOMAIN*, *username@ADDOMAIN[.something]*, or *username@addomain[.something]*; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
auth_to_local = DEFAULT
}
```

- d. Restart the KDC service.

```
[root@ipaserver ~]# service krb5kdc restart
```

14. Configure domain mapping in SSSD.

- a. Open the **/etc/sss/sss.conf**.

```
[root@ipaserver ~]# vim /etc/sss/sss.conf
```

- b. In the **[sss]** section, add **pac** to the **services** list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

- c. In the IdM domain section, add the **subdomains_provider** parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed; adding this line makes it possible to use the existing configuration.

Subdomains and SSSD are described in more detail in the [IdM provider configuration examples in the SSSD chapter of the Deployment Guide](#).

- d. Save the changes to the **sss.conf** file.
- e. Restart SSSD.

```
[root@ipaserver ~]# service sss restart
```

15. Restart the **iptables** and **ip6tables** services on the IdM server.

```
[root@ipaserver ~]# service iptables start
```

16. Restart the Windows firewall.

14.5. Creating IdM Groups for Active Directory Users

User groups are required to set access permissions, host-based access control, sudo rules, and other controls on IdM users. These groups are what grant access to IdM domain resources, as well as restricting access.

However, Active Directory users cannot be added directly to IdM user groups. This means that Active

Directory users require special configuration in order to access IdM domain resources.

As described in [Section 14.1.1.4, “Kerberos Realms, Authentication, and Authorization”](#), Active Directory users are added to the IdM domain in a kind of daisy chain. They are added to a group on the Active Directory side, then that group is added to an IdM external group (meaning, a non-POSIX group), and then that external group is added to a local POSIX group as a member. The IdM POSIX group can then be used for user/role management of Active Directory users.

1. Create or select the group in the Active Directory domain to use to manage Active Directory users in the IdM realm. (Multiple groups can be used and added to different groups on the IdM side.)
2. Create an *external* group in the IdM domain for the Active Directory users. This correlates to the Active Directory group. Using the **--external** argument indicates that this group will contain members from outside the IdM domain. For example:

```
[root@ipaserver ~]# ipa group-add --desc='AD users external map'
ad_users_external --external
-----
Added group "ad_users_external"
-----
Group name: ad_users_external
Description: AD users external map
```

3. Create the *POSIX* group for actually administering the IdM policies.

```
[root@ipaserver ~]# ipa group-add --desc='AD users' ad_users
-----
Added group "ad_users"
-----
Group name: ad_users
Description: AD users
GID: 129600004
```

4. Add the Active Directory group to the IdM external group as an external member. The Active Directory group is identified by the name *DOMAIN\group_name*. The group name is then mapped to the Active Directory SID for the group. For example:

```
[root@ipaserver ~]# ipa group-add-member ad_users_external --external
"AD\Domain Users"
[member user]:
[member group]:
Group name: ad_users_external
Description: AD users external map
External member: S-1-5-21-3655990580-1375374850-1633065477-513
SID_DOM_GROUP (2)
-----
Number of members added 1
-----
```

5. Add the external IdM group to the POSIX IdM group as a member. For example:

```
[root@ipaserver ~]# ipa group-add-member ad_users --groups ad_users_external
Group name: ad_users
Description: AD users
GID: 129600004
Member groups: ad_users_external
-----
Number of members added 1
-----
```

14.6. Using SSH from Active Directory Machines for IdM Resources

When a trust is configured, Active Directory users can access machines, services, and files on IdM hosts using SSH and their Active Directory credentials.

One critical factor when using SSH is the username. The username must meet several criteria:

- The username must have the format *ad_user@ad_domain*.
- The domain name itself must be lower-case. This is required for Kerberos principal mapping.
- The case of the username must match, exactly, the case of the username in Active Directory. *jsmith* and *JSmith* are considered different users because of the different cases.



NOTE

When using PuTTY on the Windows machine, make sure that GSS-API credential delegation is enabled.

14.7. Using Trust with Kerberized Web Applications

Any existing web application can be configured to use Kerberos authentication, which references the trusted Active Directory and IdM Kerberos realms.

For example, for an Apache server, set the ***KrbAuthRealms*** directive for the application location to the name of the IdM domain and set the location for the keytab (***Krb5Keytab***). Also set other parameters to enable Kerberos authentication, the service name used for the keytab (HTTP), and the Kerberos methods (which enables password-based authentication for valid users).

```
<Location "/mywebapp">

AuthType Kerberos
AuthName "IPA Kerberos authentication"
KrbMethodNegotiate on
KrbMethodK5Passwd on
KrbServiceName HTTP
KrbAuthRealms IDM_DOMAIN
Krb5Keytab /etc/httpd/conf/ipa.keytab
KrbSaveCredentials off
Require valid-user
</Location>
```

The Kerberos configuration directives are covered in the [mod_auth_kerb](#) module man pages.

After changing the Apache application configuration, restart the Apache service:

```
[root@ipaserver ~]# service httpd restart
```

[5] Trusted users, however, cannot manage Identity Management itself.

Chapter 15. Identity: Integrating with Microsoft Active Directory Through Synchronization

Identity Management uses active *synchronization* to integrate user data stored in an Active Directory domain and the user data stored in the IdM domain. Critical user attributes, including passwords, are synchronized between the services.

The capability to sync Active Directory and IdM domains is inherent when an IdM server is first installed. The synchronization process is configured by creating *agreements* between the IdM server and the Active Directory domain controller.

This chapter describes how to configure synchronization, how to configure Active Directory for integration with IdM, and how to configure Windows systems within the Active Directory domain to be aware of the IdM domain.

15.1. Supported Windows Platforms

Entry synchronization is performed through a process similar to replication, which uses hooks to connect to and retrieve directory data from the Windows server.

Password synchronization is performed through a Windows service which is installed on the Windows server and then communicates to the Identity Management server.

Both entry and password synchronization are supported with these Windows servers:

- Windows Server 2008 R2 (32-bit)
- Windows Server 2008 R2 (64-bit)

The version of the password sync service which works with Windows is 1.1.5. This is available in the Red Hat Directory Server downloads part of Red Hat Network.

15.2. About Active Directory and Identity Management

Within the IdM domain, information is shared among servers and replicas by copying that information, reliably and predictably, between the data masters (servers) and other data masters. This process is *replication*.

A similar process can be used to share data between the IdM domain and a Microsoft Active Directory domain. This is *synchronization*.

Synchronization is the process of copying data back and forth between Active Directory and Identity Management.

Synchronization is defined in an *agreement* between an IdM server and an Active Directory domain controller. The sync agreement defines all of the information required to identify sync-able user entries (like the subtree to synchronize and requisite object classes in the user entries) as well as defining how account attributes are handled. The sync agreements are created with default values which can be tweaked to meet the needs of a specific domain. When two servers are involved in synchronization, they are called *peers*.

Synchronization is most commonly *bi-directional*. Information is sent back and forth between the IdM domain and the Windows domain in a process that is very similar to how IdM servers and replicas share information among themselves. It is possible to configure synchronization — or certain data areas — to

only sync one way. That is *uni-directional* synchronization.

To prevent the risk of data conflicts, synchronization is configured between one Identity Management server and one Active Directory domain controller. The Identity Management server propagates changes back to the IdM domain, while the domain controller propagates changes back to the Windows domain.

There are some key features to IdM synchronization:

- A synchronization operation runs every five minutes.
- Synchronization can only be configured with one Active Directory domain. Multiple domains are not supported.
- Synchronization can only be configured with *one* Active Directory domain controller. However, it is possible to have a list of failover Active Directory domain controllers. Likewise, there can be a list of failover IdM servers to keep synchronization uninterrupted.
- Only user information is synchronized.
- Both user attributes and passwords can be synchronized.
- While modifications are bi-directional (going both from Active Directory to IdM and from IdM to Active Directory), creating or adding accounts are only uni-directional, from Active Directory to Identity Management. New accounts created in Active Directory are synchronized over to IdM automatically. However, user accounts created in IdM must also be created in Active Directory before they will be synchronized.
- Account lock information is synchronized by default, so a user account which is disabled in one domain is disabled in the other.
- Password synchronization changes take effect immediately.

When Active Directory users are synchronized over to IdM, certain attributes (including Kerberos and POSIX attributes) will have IPA attributes are automatically added to the user entries. These attributes are used by IdM within its domain. They are not synchronized back over the corresponding Active Directory user entry.

Some of the data in synchronization can be modified as part of the synchronization process. For examples, certain attributes can be automatically added to Active Directory user accounts when they are synced over to the IdM domain. These attribute changes are defined as part of the synchronization agreement and are described in [Section 15.5.3, “Changing the Behavior for Syncing User Account Attributes”](#).

15.3. About Synchronized Attributes

Identity Management synchronizes a subset of user attributes between IdM and Active Directory user entries. Any other attributes present in the entry, either in Identity Management or in Active Directory, are ignored by synchronization.



NOTE

Most POSIX attributes are not synchronized.

Although there are significant schema differences between the Active Directory LDAP schema and the 389 Directory Server LDAP schema used by Identity Management, there are many attributes that are the same. These attributes are simply synchronized between the Active Directory and IdM user entries, with no changes to the attribute name or value format.

User Schema That Are the Same in Identity Management and Windows Servers

- cn [6]
- physicalDeliveryOfficeName
- description
- postOfficeBox
- destinationIndicator
- postalAddress
- facsimileTelephoneNumber
- postalCode
- givenname
- registeredAddress
- homePhone
- sn
- homePostalAddress
- st
- initials
- street
- l
- telephoneNumber
- mail
- teletexTerminalIdentifier
- mobile
- telexNumber
- o
- title
- ou
- usercertificate
- pager
- x121Address

Some attributes have different names but still have direct parity between IdM (which uses 389 Directory Server) and Active Directory. These attributes are *mapped* by the synchronization process.

Table 15.1. User Schema Mapped between Identity Management and Active Directory

Identity Management	Active Directory
cn [a]	name
nsAccountLock	userAccountControl
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations
[a] The cn is mapped directly (cn to cn) when syncing from Identity Management to Active Directory. When syncing from Active Directory cn is mapped from the name attribute in Active Directory to the cn attribute in Identity Management.	

15.3.1. User Schema Differences between Identity Management and Active Directory

Even though attributes may be successfully synced between Active Directory and IdM, there may still be differences in how Active Directory and Identity Management define the underlying X.500 object classes. This could lead to differences in how the data are handled in the different LDAP services.

This section describes the differences in how Active Directory and Identity Management handle some of the attributes which can be synchronized between the two domains.

15.3.1.1. Values for cn Attributes

In 389 Directory Server, the **cn** attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the Identity Management **cn** attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that, potentially, if a **cn** value is added to an Active Directory entry and that value is not one of the values for **cn** in Identity Management, then all of the Identity Management **cn** values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the **cn** attribute as its naming attribute, where Identity Management uses **uid**. This means that there is the potential to rename the entry entirely (and accidentally) if the **cn** attribute is edited in the Identity Management. If that **cn** change is written over to the Active Directory entry, then the entry is renamed, and the new named entry is written back over to Identity Management.

15.3.1.2. Values for street and streetAddress

Active Directory uses the attribute **streetAddress** for a user's postal address; this is the way that 389 Directory Server uses the **street** attribute. There are two important differences in the way that Active Directory and Identity Management use the **streetAddress** and **street** attributes, respectively:

- ▶ In 389 Directory Server, **streetAddress** is an alias for **street**. Active Directory also has the **street** attribute, but it is a separate attribute that can hold an independent value, not an alias for **streetAddress**.
- ▶ Active Directory defines both **streetAddress** and **street** as single-valued attributes, while 389 Directory Server defines **street** as a multi-valued attribute, as specified in RFC 4519.

Because of the different ways that 389 Directory Server and Active Directory handle **streetAddress** and **street** attributes, there are two rules to follow when setting address attributes in Active Directory and Identity Management:

- ▶ The synchronization process maps **streetAddress** in the Active Directory entry to **street** in Identity Management. To avoid conflicts, the **street** attribute should not be used in Active Directory.
- ▶ Only one Identity Management **street** attribute value is synced to Active Directory. If the **streetAddress** attribute is changed in Active Directory and the new value does not already exist in Identity Management, then all **street** attribute values in Identity Management are replaced with the new, single Active Directory value.

15.3.1.3. Constraints on the initials Attribute

For the **initials** attribute, Active Directory imposes a maximum length constraint of six characters, but 389 Directory Server does not have a length limit. If an **initials** attribute longer than six characters is added to Identity Management, the value is trimmed when it is synchronized with the Active Directory entry.

15.3.1.4. Requiring the surname (sn) Attribute

Active Directory allows **person** entries to be created without a surname attribute. However, RFC 4519 defines the **person** object class as requiring a surname attribute, and this is the definition used in Directory Server.

If an Active Directory **person** entry is created without a surname attribute, that entry will not be synced over to IdM since it fails with an object class violation.

15.3.2. Active Directory Entries and RFC 2307 Attributes

Windows uses unique, random *security IDs (SIDs)* to identify users. These SIDs are assigned in blocks or ranges, identifying different system user types within the Windows domain. When users are synchronized between Identity Management and Active Directory, Windows SIDs for users are mapped to the Unix UIDs used by the Identity Management entry. Another way of saying this is that the Windows SID is the only ID within the Windows entry which is used as an identifier in the corresponding Unix entry, and then it is used in a mapping.

When Active Directory domains interact with Unix-style applications or domains, then the Active Directory domain may use Services for Unix or IdM for Unix to enable Unix-style **uidNumber** and **gidNumber** attributes. This allows Windows user entries to follow the specifications for those attributes in [RFC 2307](#).

However, the **uidNumber** and **gidNumber** attributes are not actually used as the **uidNumber** and **gidNumber** attributes for the Identity Management entry. The Identity Management **uidNumber** and **gidNumber** attributes are generated when the Windows user is synced over.

**NOTE**

The **uidNumber** and **gidNumber** attributes defined and used in Identity Management are not the same **uidNumber** and **gidNumber** attributes defined and used in the Active Directory entry, and the numbers are not related.

15.4. Setting up Active Directory for Synchronization

Synchronizing user accounts alone is enabled within IdM, so all that is necessary is to set up a sync agreement ([Section 15.5.2, “Creating Synchronization Agreements”](#)). However, the Active Directory does need to be configured in a way that allows the Identity Management server to connect to it.

15.4.1. Creating an Active Directory User for Sync

On the Windows server, it is necessary to create the user that the IdM server will use to connect to the Active Directory domain.

The process for creating a user in Active Directory is covered in the Windows server documentation at <http://technet.microsoft.com/en-us/library/cc732336.aspx>. The new user account must have the proper permissions:

- Grant the sync user account **Replicating directory changes** rights to the synchronized Active Directory subtree. Replicator rights are required for the sync user to perform synchronization operations.
Replicator rights are described in <http://support.microsoft.com/kb/303972>.
- Add the sync user as a member of the **Account Operator** and **Enterprise Read-Only Domain controller** groups. It is not necessary for the user to belong to the full **Domain Admin** group.

15.4.2. Setting up an Active Directory Certificate Authority

The Identity Management server connects to the Active Directory server using a secure connection. This requires that the Active Directory server have an available CA certificate or CA certificate chain available, which can be imported into the Identity Management security databases, so that the Windows server is a trusted peer.

While this could technically be done with an external (to Active Directory) CA, most deployments should use the Certificate Services available with Active Directory.

The procedure for setting up and configuring certificate services on Active Directory is covered in the Microsoft documentation at [http://technet.microsoft.com/en-us/library/cc772393\(v=WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc772393(v=WS.10).aspx).

15.5. Managing Synchronization Agreements

15.5.1. Trusting the Active Directory and IdM CA Certificates

Both Active Directory and Identity Management use certificates for server authentication. For the Active Directory and IdM SSL server certificates to be trusted by each other, both servers need to trust the CA certificate for the CA which issued those certificates. This means that the Active Directory CA certificate needs to be imported into the IdM database, and the IdM CA certificate needs to be imported into the Active Directory database.

1. On the Active Directory server, download the IdM server's CA certificate from

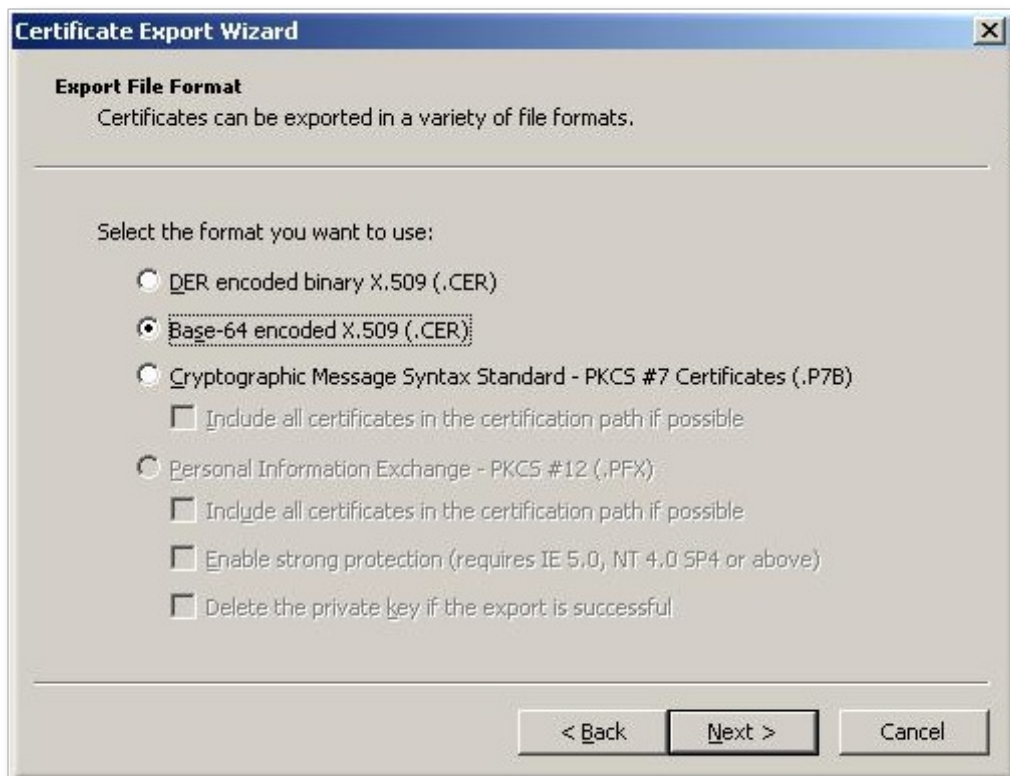
<http://ipa.example.com/ipa/config/ca.crt>.

2. Install the IdM CA certificate in the Active Directory certificate database. This can be done using the Microsoft Management Console or the [certutil utility](#). For example:

```
certutil -installcert -v -config "ipaserver.example.com\Example Domain CA"
c:\path\to\ca.crt
```

For more details, see the Active Directory documentation.

3. Export the Active Directory CA certificate.
 - a. In **My Network Places**, open the CA distribution point.
 - b. Double-click the security certificate file (**.crt** file) to display the **Certificate** dialog box.
 - c. On the **Details** tab, click **Copy to File** to start the **Certificate Export Wizard**.
 - d. Click **Next**, and then select **Base-64 encoded X.509 (.CER)**.



- e. Specify a suitable directory and file name for the exported file. Click **Next** to export the certificate, and then click **Finish**.
4. Copy the Active Directory certificate over to the IdM server machine.
5. Download the IdM server's CA certificate from **<http://ipa.example.com/ipa/config/ca.crt>**.
6. Copy both the Active Directory CA certificate and the IdM CA certificate into the **/etc/openldap/cacerts/** directory.
7. Update the hash symlinks for the certificates.

```
cacertdir_rehash /etc/openldap/cacerts/
```

8. Edit the **/etc/openldap/ldap.conf** file, and add the information to point to and use the certificates in the **/etc/openldap/cacerts/** directory.

```
TLS_CACERTDIR /etc/openldap/cacerts/
TLS_REQCERT allow
```

15.5.2. Creating Synchronization Agreements

Synchronization agreements are created on the IdM server using the **ipa-replica-manage connect** command because it creates a *connection* to the Active Directory domain. The options to create the synchronization agreement are listed in [Table 15.2, “Synchronization Agreement Options”](#).

1. Make sure that the Active Directory and IdM servers trust each other's CA certificates, as in [Section 15.5.1, “Trusting the Active Directory and IdM CA Certificates”](#).
2. Remove any existing Kerberos credentials on the IdM server.

```
$ kdestroy
```

3. Use the **ipa-replica-manage connect** command to create a Windows synchronization agreement. This requires the **--winsync** option. If passwords will be synchronized as well as user accounts, then also use the **--passsync** option and set a password to use for Password Sync.

The **--binddn** and **--bindpwd** options give the username and password of the system account on the Active Directory server that IdM will use to connect to the Active Directory server.

```
$ ipa-replica-manage connect --winsync
--binddn cn=administrator,cn=users,dc=example,dc=com
--bindpw Windows-secret
--passsync secretpwd
--cacert /etc/openldap/cacerts/windows.cer
adserver.example.com -v
```

4. When prompted, enter the Directory Manager password.
5. *Optional.* Configure Password Synchronization, as in [Section 15.6.2, “Setting up Password Synchronization”](#).

Table 15.2. Synchronization Agreement Options

Option	Description
<code>--winsync</code>	Identifies this as a synchronization agreement.
<code>--binddn</code>	Gives the full user DN of the synchronization identity. This is the user DN that the IdM LDAP server uses to bind to Active Directory. This user must exist in the Active Directory domain and must have replicator, read, search, and write permissions on the Active Directory subtree.
<code>--bindpw</code>	Gives the password for the sync user.
<code>--passsync</code>	Gives the password for the Windows user account which is involved in synchronization.
<code>--cacert</code>	Gives the full path and file name of the Active Directory CA certificate. This certificate is exported in Section 15.5.1, “Trusting the Active Directory and IdM CA Certificates” .
<code>--win-subtree</code>	Gives the DN of the Windows subtree containing the users to synchronize. The default value is cn=Users,\$SUFFIX .
<code>AD_server_name</code>	Gives the hostname of the Active Directory domain controller.

15.5.3. Changing the Behavior for Syncing User Account Attributes

When the sync agreement is created, it has certain default behaviors defined for how the synchronization process handled the user account attributes during synchronization. The types of behaviors are things like how to handle lockout attributes or how to handle different DN formats. This behavior can be changed by editing the synchronization agreement. The list of attribute-related parameters are in [Table 15.3, “Synced Attribute Settings”](#).

The sync agreement exists as a special plug-in entry in the LDAP server and each attribute behavior is set through an LDAP attribute. To change the sync behavior, use the **ldapmodify** command to modify the LDAP server entry directly.

For example, account lockout attributes are synchronized between IdM and Active Directory by default, but this can be disabled by editing the **ipaWinSyncAcctDisable** attribute. (Changing this means that if an account is disabled in Active Directory, it is still active in IdM and vice versa.)

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password

dn: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
replace: ipaWinSyncAcctDisable
ipaWinSyncAcctDisable: none

modifying entry "cn=ipa-winsync,cn=plugins,cn=config"
```

Table 15.3. Synced Attribute Settings

Parameter	Description	Possible Values
General User Account Parameters		
ipaWinSyncNewEntryFilter	Sets the search filter to use to find the entry which contains the list of object classes to add to new user entries.	The default is (cn=ipaConfig) .
ipaWinSyncNewUserOCCAttr	Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.	The default is ipauserobjectclasses .
ipaWinSyncHomeDirAttr	Identifies which attribute in the entry contains the default location of the POSIX home directory.	The default is ipaHomesRootDir .
ipaWinSyncUserAttr	Sets an additional attribute with a specific value to add to Active Directory users when they are synced over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and the sync process adds all of the values to the entry.	ipaWinSyncUserAttr: <i>attributeName attributeValue</i>
 NOTE This only sets the attribute value if the entry does not already have that attribute present. If the attribute is present, then the entry's value is used when the Active Directory entry is synced over.		
ipaWinSyncForceSync	Sets whether to check existing IdM users which match an existing Active Directory user should be automatically edited so they can be synchronized. If an IdM user account has a uid parameter which is identical to the samAccountName in an existing Active Directory user, then that account is <i>not</i> synced by default. This attribute tells the sync service to add the	true false

<i>ntUser</i> and <i>ntUserDomainId</i> to the IdM user entries automatically, which allows them to be synchronized.		
User Account Lock Parameters		
<code>ipaWinSyncAcctDisable</code>	Sets which way to synchronize account lockout attributes. It is possible to control which account lockout settings are in effect. For example, <i>to_ad</i> means that when account lockout attribute is set in IdM, its value is synced over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synced from both domains.	» both (default) » to_ad » to_ds » none
<code>ipaWinSyncInactivatedFilter</code>	Sets the search filter to use to find the DN of the group used to hold inactivated (disabled) users. This does not need to be changed in most deployments.	The default is <code>(&(cn=inactivated)(objectclass=groupOfNames))</code> .
<code>ipaWinSyncActivatedFilter</code>	Sets the search filter to use to find the DN of the group used to hold active users. This does not need to be changed in most deployments.	The default is <code>(&(cn=activated)(objectclass=groupOfNames))</code> .
Group Parameters		
<code>ipaWinSyncDefaultGroupAttr</code>	Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the <i>gidNumber</i> for the user account.	The default is <code>ipaDefaultPrimaryGroup</code> .
<code>ipaWinSyncDefaultGroupFilter</code>	Sets the search filter to map the group name to the POSIX <i>gidNumber</i> .	The default is <code>(&(gidNumber=*)(objectclass=posixGroup)(cn=groupAttr_value))</code> .
Realm Parameters		
<code>ipaWinSyncRealmAttr</code>	Sets the attribute which contains the realm name in the realm entry.	The default is <i>cn</i> .
<code>ipaWinSyncRealmFilter</code>	Sets the search filter to use to find the entry which contains the IdM realm name.	The default is <code>(objectclass=krbRealmContainer)</code> .

15.5.4. Changing the Synchronized Windows Subtree

Creating a synchronization agreement automatically sets the two subtrees to use as the synchronized

user database. In IdM, the default is **cn=users,cn=accounts,\$SUFFIX**, and for Active Directory, the default is **CN=Users,\$SUFFIX**.

The value for the Active Directory subtree can be set to a non-default value when the sync agreement is created by using the **--win-subtree** option. After the agreement is created, the Active Directory subtree can be changed by using the **ldapmodify** command to edit the **nsds7WindowsReplicaSubtree** value in the sync agreement entry.

1. Get the name of the sync agreement, using **ldapsearch**. This search returns only the values for the **dn** and **nsds7WindowsReplicaSubtree** attributes instead of the entire entry.

```
[jsmith@ipaserver ~]$ ldapsearch -xLLL -D "cn=directory manager" -w password
-p 389 -h ipaserver.example.com -b cn=config
objectclass=nsds7WindowsReplicationAgreement dn nsds7WindowsReplicaSubtree

dn:
cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapp
ing tree,cn=config
nsds7WindowsReplicaSubtree: cn=users,dc=example,dc=com

... 8< ...
```

2. Modify the sync agreement

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -W -p 389 -h
ipaserver.example.com <<EOF
dn:
cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapp
ing tree,cn=config
changetype: modify
replace: nsds7WindowsReplicaSubtree
nsds7WindowsReplicaSubtree: cn=alternateusers,dc=example,dc=com
EOF

modifying entry
"cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapp
ing tree,cn=config"
```

The new subtree setting takes effect immediately. If a sync operation is currently running, then it takes effect as soon as the current operation completes.

15.5.5. Configuring Uni-Directional Sync

By default, all modifications and deletions are bi-directional. A change in Active Directory is synced over to Identity Management, and a change to an entry in Identity Management is synced over to Active Directory. This is essentially an equitable, multi-master relationship, where both Active Directory and Identity Management are equal peers in synchronization and are both data masters.

However, there can be some data structure or IT designs where only one domain should be a data master and the other domain should accept updates. This changes the sync relationship from a multi-master relationship (where the peer servers are equal) to a master-consumer relationship.

This is done by setting the **oneWaySync** parameter on the sync agreement. The possible values are **fromWindows** (for Active Directory to Identity Management sync) and **toWindows** (for Identity Management to Active Directory sync).

For example, to sync changes from Active Directory to Identity Management:

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password -p 389
-h ipaserver.example.com

dn: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```



IMPORTANT

Enabling uni-directional sync does *not* automatically prevent changes on the un-synchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync is configured to go from Active Directory to Identity Management, so Active Directory is (in essence) the data master. If an entry is modified or even deleted on the Identity Management, then the Identity Management information is different than the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

15.5.6. Deleting Synchronization Agreements

Synchronization can be stopped by deleting the sync agreement which *disconnects* the IdM and Active Directory servers. In the inverse of creating a sync agreement, deleting a sync agreement uses the **ipa-replica-manage disconnect** command and then the hostname of the Active Directory server.

1. Delete the sync agreement.

```
# ipa-replica-manage disconnect adserver.example.com
```

2. Remove the Active Directory CA certificate from the IdM server database:

```
# certutil -D -d /etc/dirsrv/slapd-EXAMPLE.COM/ -n "Imported CA"
```

15.5.7. Winsync Agreement Failures

Creating the sync agreement fails because it cannot connect to the Active Directory server.

One of the most common sync agreement failures is that the IdM server cannot connect to the Active Directory server:

```
"Update failed! Status: [81 - LDAP error: Can't contact LDAP server]"
```

This can occur if the wrong Active Directory CA certificate was specified when the agreement was created. This creates duplicate certificates in the IdM LDAP database (in the `/etc/dirsrv/slapd-DOMAIN/` directory) with the name *Imported CA*. This can be checked using **certutil**:

```
$ certutil -L -d /etc/dirsrv/slapd-DOMAIN/
```

Certificate Nickname	Trust Attributes
SSL,S/MIME,JAR/XPI	
CA certificate	CTu, u, Cu
Imported CA	CT, , C
Server-Cert	u, u, u
Imported CA	CT, , C

To resolve this issue, clear the certificate database:

```
# certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"
```

This deletes the CA certificate from the LDAP database.

There are errors saying passwords are not being synced because it says the entry exists

For some entries in the user database, there may be an informational error message that the password is not being reset because the entry already exists:

```
"Windows PassSync entry exists, not resetting password"
```

This is not an error. This message occurs when an exempt user, the Password Sync user, is not being changed. The Password Sync user is the operational user which is used by the service to change the passwords in IdM.

15.6. Managing Password Synchronization

Password synchronization is configured separately from Windows Synchronization.

15.6.1. Setting up the Windows Server for Password Synchronization

Synchronizing passwords requires two things:

- Active Directory must be running in SSL.
- The Password Sync Service must be installed on *each* Active Directory domain controller.

The Password Sync Service records password changes and synchronizes them, over a secure connection, to the IdM entry.



TIP

Install the Microsoft Certificate System in Enterprise Root Mode. Active Directory will then automatically enroll to retrieve its SSL server certificate.

1. Make sure that the Active Directory password complexity policies are enabled so that the Password Sync service will run.
 - a. Run **secpol.msc** from the command line.
 - b. Select **Security Settings**.
 - c. Open **Account Policies**, and then open **Password Policy**.
 - d. Enable the **Password must meet complexity requirements** option and save.



2. If SSL is not already enabled, set up SSL on the Active Directory server. Setting up LDAPS is explained in more detail in the Microsoft knowledgebase at <http://support.microsoft.com/kb/321051>.
 - a. Install a certificate authority in the **Windows Components** section in **Add/Remove Programs**.
 - b. Select the **Enterprise Root CA** option.
 - c. Reboot the Active Directory server. If IIS web services are running, the CA certificate can be accessed by opening **http://servername/certsrv**.
 - d. Set up the Active Directory server to use the SSL server certificate.
 - a. Create a certificate request **.inf**, using the fully-qualified domain name of the Active Directory as the certificate subject. For example:

```

;----- request.inf -----

[Version]

Signature="$Windows NT$"

[NewRequest]

Subject = "CN=ad.server.example.com, O=Engineering, L=Raleigh,
S=North Carolina, C=US"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = False
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0

[EnhancedKeyUsageExtension]

OID=1.3.6.1.5.5.7.3.1

;-----

```

For more information on the **.inf** request file, see the Microsoft documentation, such as <http://technet.microsoft.com/en-us/library/cc783835.aspx>.

- b. Generate the certificate request.

```
certreq -new request.inf request.req
```

- c. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```

**NOTE**

If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is **http://servername/certsrv**.

- d. Accept the certificate request. For example:

```
certreq -accept certnew.cer
```

- e. Make sure that the server certificate is present on the Active Directory server. In the **File** menu, click **Add/Remove**, then click **Certificates** and **Personal>Certificates**.
- f. Import the CA certificate from Directory Server into Active Directory. Click **Trusted Root CA**, then **Import**, and browse for the Directory Server CA certificate.
- e. Reboot the domain controller.

15.6.2. Setting up Password Synchronization

Install the Password Sync Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the **PassSync.msi** file to the Active Directory machine.
 - a. Log into the Customer Portal.
 - b. Click the **Downloads** tab.
 - c. Click the **Red Hat Enterprise Linux** downloads button in the middle of the page.
 - d. Filter the downloads by using a search term such as *Directory Server*, and then expand one of the Red Hat Enterprise Linux versions.
 - e. Click the Directory Server link.
 - f. On the Directory Server page, download the appropriate version of the WinSync Installer. This is the Password Sync MSI file (**RedHat-PassSync-1.1.5-arch.msi**).

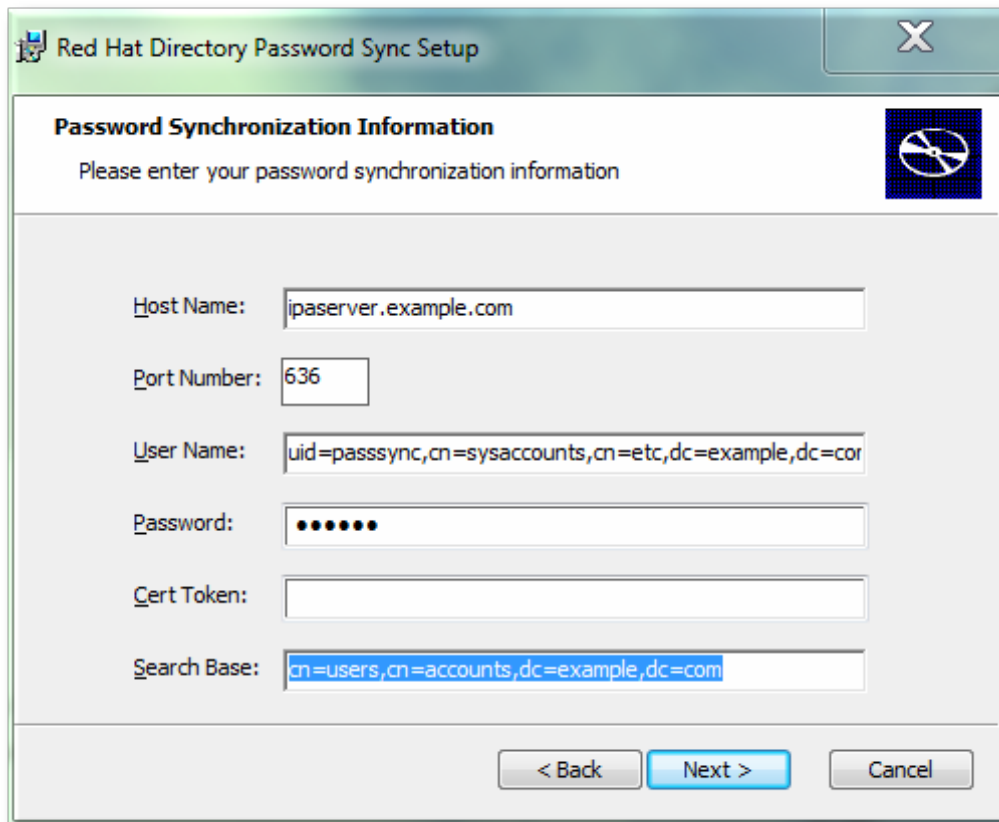
**NOTE**

Regardless of the Red Hat Enterprise Linux architecture, there are two PassSync packages available, one for 32-bit Windows servers and one for 64-bit. Make sure to select the appropriate packages for your Windows platform.

2. Double-click the Password Sync MSI file to install it.
3. The **Password Sync Setup** window appears. Hit **Next** to begin installing.
4. Fill in the information to establish the connection to the IdM server.
 - The IdM server connection information, including the hostname and secure port number.
 - The username of the system user which Active Directory uses to connect to the IdM machine. This account is configured automatically when sync is configured on the IdM server. The default account is **uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com**.
 - The password set in the **--passsync** option when the sync agreement was created.
 - The search base for the people subtree on the IdM server. The Active Directory server

connects to the IdM server similar to an **ldapsearch** or replication operation, so it has to know where in the IdM subtree to look for user accounts. The user subtree is **cn=users,cn=accounts,dc=example,dc=com**.

- » The certificate token is not used at this time, so that field should be left blank.



The image shows a Windows-style dialog box titled "Red Hat Directory Password Sync Setup". Inside, there is a section titled "Password Synchronization Information" with a sub-header "Please enter your password synchronization information". The dialog contains several input fields: "Host Name" with the value "ipaserver.example.com", "Port Number" with "636", "User Name" with "uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com", "Password" with masked characters, "Cert Token" which is empty, and "Search Base" with "cn=users,cn=accounts,dc=example,dc=com". At the bottom, there are three buttons: "< Back", "Next >", and "Cancel".

Hit **Next**, then **Finish** to install Password Sync.

5. Import the IdM server's CA certificate into the Active Directory certificate store.
 - a. Download the IdM server's CA certificate from **<http://ipa.example.com/ipa/config/ca.crt>**.
 - b. Copy the IdM CA certificate to the Active Directory server.
 - c. Install the IdM CA certificate in the Password Sync database. For example:

```
cd "C:\Program Files\Red Hat Directory Password Synchronization"
certutil.exe -d . -A -n "IPASERVER.EXAMPLE.COM IPA CA" -t CT,, -a -i
ipaca.crt
```

6. Reboot the Windows machine to start Password Sync.



NOTE

The Windows machine must be rebooted. Without the rebooting, **PasswordHook.dll** is not enabled, and password synchronization will not function.

The first attempt to synchronize passwords, which happened when the Password Sync application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory sync peers. The tools to create the certificate and key databases is installed with the **.msi**.

15.6.3. Allowing Users to Change Other Users' Passwords Cleanly

By default, every time an administrator changes a user password, that user is required to reset the password at the next login. However, this behavior can be changed to allow administrators to reset a password *without* requiring an immediate password reset.

The ***passSyncManagersDNs*** attribute lists administrator accounts which are allowed to perform password change operations *and* which will not then require a password reset.



IMPORTANT

This is required for password synchronization because, otherwise, whenever a password is synchronized, the IdM server would interpret that as a password change operation and then require a password change at the next login.

Edit the password synchronization entry, **cn=ipa_pwd_extop,cn=plugins,cn=config**, and add the ***passSyncManagersDNs*** attribute with the name of the user. This attribute is multi-valued. For example:

```
$ ldapmodify -x -D "cn=Directory Manager" -w secret -h ldap.example.com -p 389

dn: cn=ipa_pwd_extop,cn=plugins,cn=config
changetype: modify
add: passSyncManagersDNs
passSyncManagersDNs: uid=admin,cn=users,cn=accounts,dc=example,dc=com
```



WARNING

Be careful to limit the listed DNs only to administrator accounts which require the ability to set user passwords. Any user listed here is given access to all user passwords, which is extremely powerful.

[6] The ***cn*** is treated differently than other synced attributes. It is mapped directly (***cn*** to ***cn***) when syncing from Identity Management to Active Directory. When syncing from Active Directory to Identity Management, however, ***cn*** is mapped from the ***name*** attribute on Windows to the ***cn*** attribute in Identity Management.

Chapter 16. Identity: Managing DNS

If the IdM server was installed with DNS configured, then all of the DNS entries for the domain — host entries, locations, records — can be managed using the IdM tools.

16.1. About DNS in IdM

DNS is one of the services that can be configured and maintained by the IdM domain. DNS is critical to the performance of the IdM domain; DNS is used for the Kerberos services and SSL connections for all servers and clients and for connections to domain services like LDAP.

While IdM can use an external DNS service, there is a lot more flexibility and control over IdM — DNS interactions when the DNS service is configured within the domain. For example, DNS records and zones can be managed within the domain using IdM tools, and clients can update their own DNS records dynamically. When a host is added to IdM, a DNS record is automatically created in IdM's DNS service for that host machine.

IdM stores all DNS information as LDAP entries. Every resource record for each machine is stored for the domain. For example, the `client1` resource has three IPv4 (A) records and one IPv6 (AAAA) record:

```
dn: idnsname=client1,idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: client1
arecord: 10.0.0.1
arecord: 10.0.0.2
arecord: 10.0.0.3
aaaaarecord: fc00::1
objectclass: top
objectclass: idnsrecord
```

The schema used to define the DNS entries is in the `/usr/share/ipa/60basev2.ldif` schema file [7].

The BIND service communicates with the Directory Server using the system `bind-dyndb-ldap` plug-in. When Identity Management is configured to manage DNS ([Section 16.3, “Setting up DNS After IdM Server Installation”](#)), IdM creates a `dynamic-db` configuration section in the `/etc/named.conf` file for the BIND service. This configures the `bind-dyndb-ldap` plug-in for the BIND (`named`) service.

When this plug-in is properly configured, it delivers the DNS records from the Directory Server to the `named` service. The configuration can be changed to adapt the behavior of the plug-in and, therefore, the LDAP-BIND interactions.

16.2. Using IdM and DNS Service Discovery with an Existing DNS Configuration

To help create and configure a suitable DNS setup, the IdM installation script creates a sample zone file. During the installation, IdM displays a message similar to the following:

```
Sample zone file for bind has been created in /tmp/sample.zone.F_uMf4.db
```

If a DNS server is already configured in the network, then the configuration in the IdM-generated file can be added to the existing DNS zone file. This allows IdM clients to find `.`. For example, this DNS zone configuration is created for an IdM server with the KDC and DNS servers all on the same machine in the `EXAMPLE.COM` realm:

Example 16.1. Default IdM DNS File

```

; ldap servers
_ldap._tcp          IN SRV 0 100 389      ipaserver.example.com.

;kerberos realm
_kerberos           IN TXT EXAMPLE.COM

; kerberos servers
_kerberos._tcp      IN SRV 0 100 88      ipaserver.example.com.
_kerberos._udp      IN SRV 0 100 88      ipaserver.example.com.
_kerberos-master._tcp IN SRV 0 100 88      ipaserver.example.com.
_kerberos-master._udp IN SRV 0 100 88      ipaserver.example.com.
_kpasswd._tcp       IN SRV 0 100 464     ipaserver.example.com.
_kpasswd._udp       IN SRV 0 100 464     ipaserver.example.com.

```

**TIP**

If DNS services are hosted by a server outside the IdM domain, then an administrator can add the lines in [Example 16.1, “Default IdM DNS File”](#) to the existing DNS zone file. This allows IdM clients and servers to continue to use DNS service discovery to find the LDAP and Kerberos servers (meaning, the IdM servers) that are required for them to participate in the IdM domain.

16.3. Setting up DNS After IdM Server Installation

DNS can be configured as part of the IdM server installation, simply by using the **--setup-dns** option. If DNS is not configured then, it can be configured later using the **ipa-dns-install** command.

16.4. DNS Notes

- ▶ Wildcards cannot be used when configuring DNS names. Only explicit DNS domain names are supported.
- ▶ The **rndc** service is not configured by the **--setup-dns** option. This service must be configured manually after the IdM server is configured.

16.5. Configuring DNS Services in Identity Management

The **ipa-dns-install** command configures the DNS services on the IdM server.

```
[root@server ~]# ipa-dns-install -p secret --ip-address=1.2.34.56 --no-forwarders
[--zone-refresh=60 | --zone-notif]
```

- ▶ **-p** gives the password for the Directory Manager user in the 389 Directory Server. All of the DNS entries are stored in the LDAP directory, so this directory must be accessed to add the DNS configuration.
- ▶ **--ip-address** gives the IP address for the master DNS server.
- ▶ **--no-forwarders** means that there are no forwarders used with the DNS service, only root servers. Alternatively, use the **--forwarder** option to define a forward to use; to specify multiple

forwarders, use the **--forwarder** option multiple times.

- Reverse DNS is configured automatically. It is possible to disable reverse DNS by using the **--no-reverse** option.

If an existing reverse DNS zone is already configured, using the **--no-reverse** option uses the existing reverse zone rather than creating a new reverse zone.

- The IdM server can actively check to see when new DNS zones are added and to update its DNS server accordingly. If no value is explicitly given, the zone refresh period is 30 seconds. The refresh interval can be set to another value using the **--zone-refresh** option, which sets the polling interval in seconds.
- Similar to refreshing the zones, the IdM server can leave a persistent search open with its Directory Server and capture any new zone changes immediately. This is enabled with the **--zone-notif** option.

If the **--zone-notif** option is used to configure DNS, then the automatic zone refresh is disabled.

16.6. Setting up the rndc Service

The **ipa-dns-install** command does not automatically configure the system's **rndc** service. This must be configured manually, after DNS is configured for IdM.

1. Create the **rndc** configuration file and key.

```
[root@server ~]# /usr/sbin/rndc-confgen -a
[root@server ~]# /sbin/restorecon /etc/rndc.conf
```

This may require some user input to create entropy while the key is being created.

2. Change the owner and permissions of the **rndc** key file.

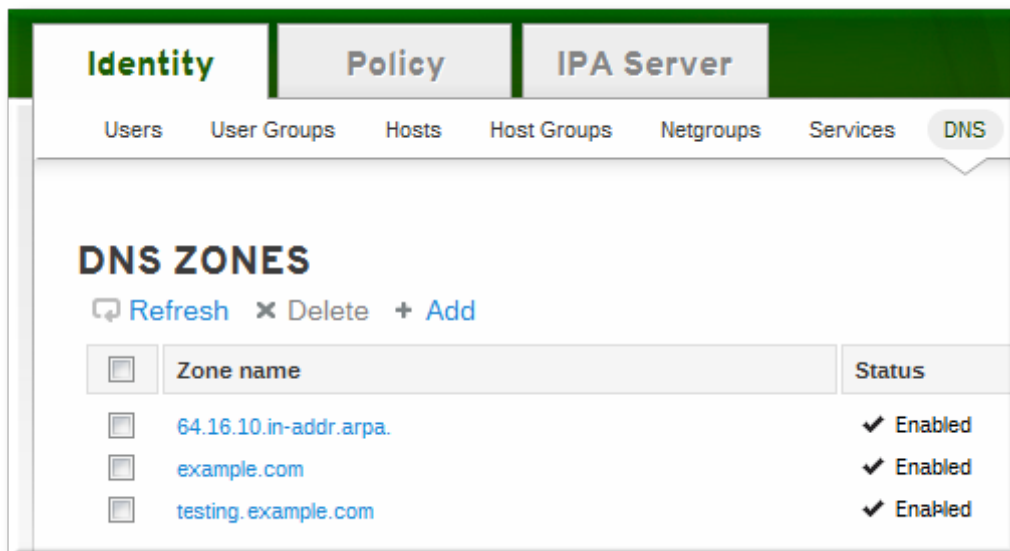
```
[root@server ~]# chown root:named /etc/rndc.key
[root@server ~]# chmod 0640 /etc/rndc.key
```

16.7. Managing DNS Zone Entries

16.7.1. Adding Forward DNS Zones

16.7.1.1. From the Web UI

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the **Add** link at the top of the list of DNS zones.



- Fill in the information about the new DNS zone. The **Zone Name** is required; this is the actual domain name. The other information about the administrator email and the authoritative name server are optional.

Add DNS Zone

☒ Zone name:

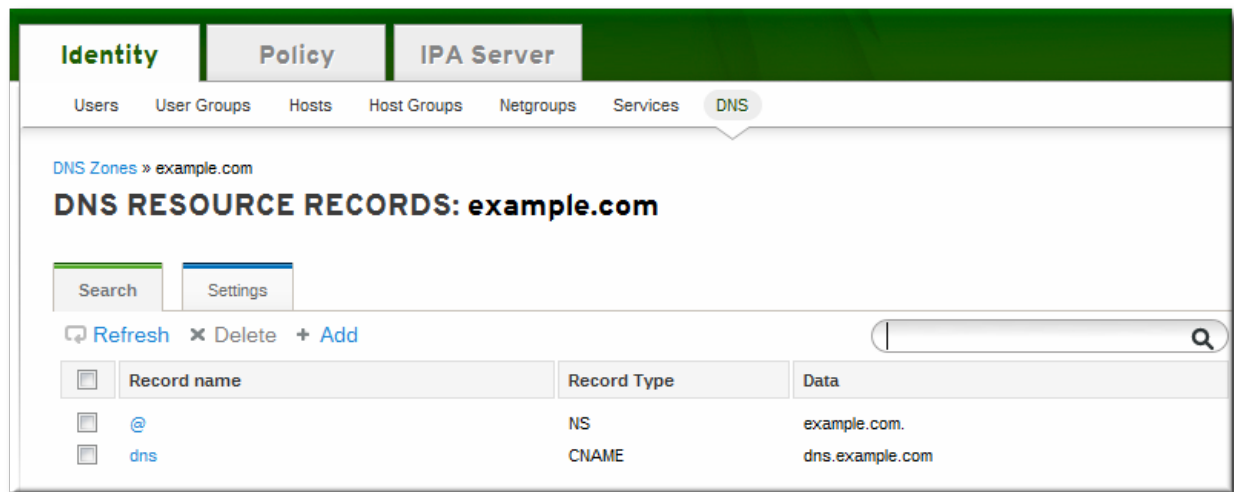
☐ Reverse zone IP network:

Authoritative nameserver:

Administrator e-mail address:

Force: ☐

- Click the **Add and Add Another** button to go directly to the DNS zone page. In the **Settings** tab, it is possible to reset the default zone configuration to enable dynamic binds ([Section 16.11.1, “Enabling Dynamic DNS Updates in the Web UI”](#)) or change other default records information ([Section 16.7.2.2, “Editing the Zone Configuration in the Web UI”](#)). It is also possible to begin adding new DNS resource records ([Section 16.8.1.1, “Adding DNS Resource Records from the Web UI”](#)) in the **DNS Resource Records** tab.



16.7.1.2. From the Command Line

The **ipa dnszone-add** command adds a new zone to the DNS domain. At a minimum, this requires the name of the new subdomain:

```
$ ipa dnszone-add domainName
```

If the name is not given, the script prompts for it. Other command-line options can also be passed with the **ipa dnszone-add** command.

To add a zone entry:

1. Add the new zone. For example:

```
$ ipa dnszone-add newserver.example.com --admin-email=admin@example.com --
minimum=3000 --dynamic-update=TRUE
```

2. Reload the name service.

```
# rndc reload
```



TIP

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the **named** service or configure the BIND service to poll the Directory Server automatically for zone changes. See [Section 16.9.2, “Enabling Zone Refreshes and Persistent Searches”](#).

16.7.2. Adding Additional Configuration for DNS Zones

A zone is created with a certain amount of configuration, set to default values.

Example 16.2. Default DNS Zone Entry Settings

```
dn: idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: example.com
idnssoamname: server.example.com.
idnssoaname: root.server.example.com.
idnssoaserial: 2011130701
idnssoarefresh: 3600
idnssoaretry: 900
idnssoaexpire: 1209600
idnssoaminimum: 3600
idnsupdatepolicy: grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self
* AAAA;
idnszoneactive: TRUE
idnsallowdynupdate: TRUE
nsrecord: server.example.com.
objectclass: top
objectclass: idnsrecord
objectclass: idnszone
```

16.7.2.1. DNS Zone Configuration Attributes

All of the possible zone settings are listed in [Table 16.1, “Zone Attributes”](#). Along with setting the actual information for the zone, the settings define how the DNS server handles the *start of authority* (SOA) record entries and how it updates its records from the DNS name server.

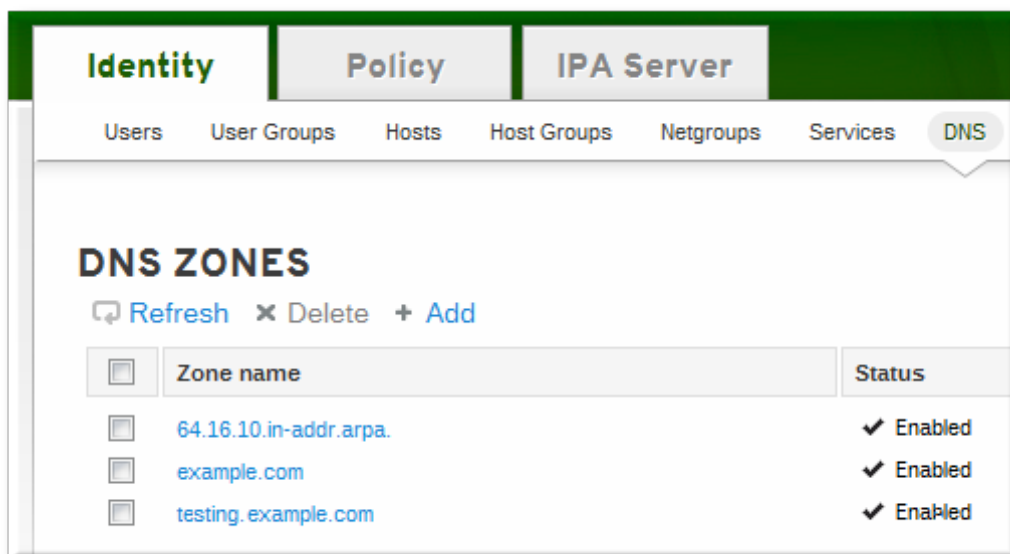
Table 16.1. Zone Attributes

Attribute	Command-Line Option	Description
Zone name	--name	Sets the name of the zone.
Authoritative nameserver	--name-server	Sets the fully-qualified domain name of the DNS name server.
Administrator e-mail address	--admin-email	Sets the email address to use for the zone administrator. This defaults to the root account on the host.
SOA serial	--serial	Sets a version number for the SOA record file.
SOA refresh	--refresh	Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.
SOA retry	--retry	Sets the time, in seconds, to wait before retrying a failed refresh operation.
SOA expire	--expire	Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.
SOA minimum	--minimum	Sets the minimum amount of time, in seconds, that data are kept in cache.
SOA time to live	--ttl	Sets the maximum time, in seconds, that information is kept in the data cache.
SOA class	--class	Sets the type of record. This is almost always IN, which stands for Internet.
BIND update policy	--update-policy	Sets the permissions allowed to clients in the DNS zone.
 IMPORTANT If this is set to false, IdM client machines will not be able to add or update their IP address. See Section 16.11, “Enabling Dynamic DNS Updates” for more information.		
Dynamic update	--dynamic-update=TRUE FALSE	Enables dynamic updates to DNS records for clients.
Name server	--ip-address	Adds the DNS name server by its IP address.

Allow transfer	<code>--allow-transfer=<i>string</i></code>	Gives a semi-colon-separated listed of IP addresses or network names which are allowed to transfer the given zone.
Allow query	<code>--allow-query</code>	Gives a semi-colon-separated listed of IP addresses or network names which are allowed to issue DNS queries.
Allow PTR sync	<code>--allow-sync-ptr=1 0</code>	Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.
Zone forwarders	<code>--forwarder=<i>string</i></code>	Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the IdM domain. To specify multiple forwarders, use the option multiple times.
Forward policy	<code>--forward-policy=only first</code>	Sets whether the zone will only forward requests to configured the DNS name servers (a <i>forward-only</i> zone) or whether it will check the forwarders first for DNS records and then check its own local records.

16.7.2.2. Editing the Zone Configuration in the Web UI

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone to edit.



3. Open the **Settings** tab.

4. Change any of the DNS zone settings. The full list of attributes is described in [Table 16.1, “Zone Attributes”](#). There are some common attributes to change:

- » *Authoritative name server*, the fully-qualified domain name of the DNS name server.
- » *Dynamic update*, to enable dynamic updates to DNS records for clients.
- » *SOA refresh*, the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.

The screenshot shows the 'DNS ZONES' section for 'example.com'. The page has a green header with tabs for 'Identity', 'Policy', and 'IPA Server'. Below the header is a navigation bar with links for 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', 'Services', and 'DNS'. The main content area is titled 'DNS ZONE: example.com' and includes a 'Search' button and a 'Settings' tab. Below the tabs are 'Refresh', 'Reset', and 'Update' links. The 'DNS ZONE SETTINGS' section contains the following fields:

- Zone name: example.com
- Status: ☒ Enabled ☐ Disabled
- Authoritative nameserver: * example.com.
- Administrator e-mail address: * hostmaster.example.com.
- SOA serial: * 2012130201
- SOA refresh: * 3600
- SOA retry: * 900
- SOA expire: * 1209600
- SOA minimum: * 3600
- SOA time to live:
- SOA class:
- Dynamic update: ☐ True ☒ False
- BIND update policy:

5. Click the **Update** link at the top of the settings page.

16.7.2.3. Editing the Zone Configuration in the Command Line

The zone can be created with additional attributes and values different from the default by passing additional options with the **dnszone-add** command. Likewise, attributes can be added or modified in the zone entry by passing the same attribute options with the **dnszone-mod** command. These are listed in [Table 16.1, “Zone Attributes”](#).

If an attribute does not exist in the DNS zone entry, then the **dnszone-mod** command adds the attribute. If the attribute exists, then it overwrites the current value with the specified value.

For example, to set a time to live for SOA records:

```
$ ipa dnszone-mod server.example.com --ttl=1800
```

This adds a new attribute to the DNS zone entry:

```
dn: idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: example.com
idnssoamname: server.example.com.
idnssoaname: root.server.example.com.
idnssoaserial: 2011130701
idnssoarefresh: 3600
idnssoaretry: 900
idnssoaexpire: 1209600
idnssoaminimum: 3600
dnsttl: 1800
idnsupdatepolicy: grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self *
AAAA;
idnszoneactive: TRUE
idnsallowdynupdate: TRUE
nsrecord: server.example.com.
objectclass: top
objectclass: idnsrecord
objectclass: idnszone
```

16.7.3. Adding Reverse DNS Zones

The process to add a reverse zone is the same as for a forward zone, as described in [Section 16.7.1, “Adding Forward DNS Zones”](#). However, the required information is different.

There are two ways to identify a reverse DNS zone:

- By zone name, in the format *reverse_ip_address.in-addr.arpa*.
- By network address, in the format *network_ip_address/subnet_mask_bit_count*

When creating a reverse zone by the zone name, set it up exactly as creating a forward zone, only reverse the order of the components of the IP address. For example, if the IP address is *1.2.3.4*, then the reverse zone name is *3.2.1.in-addr.arpa*. (with the trailing period).

In the web UI, this is set in the **Zone name** field.

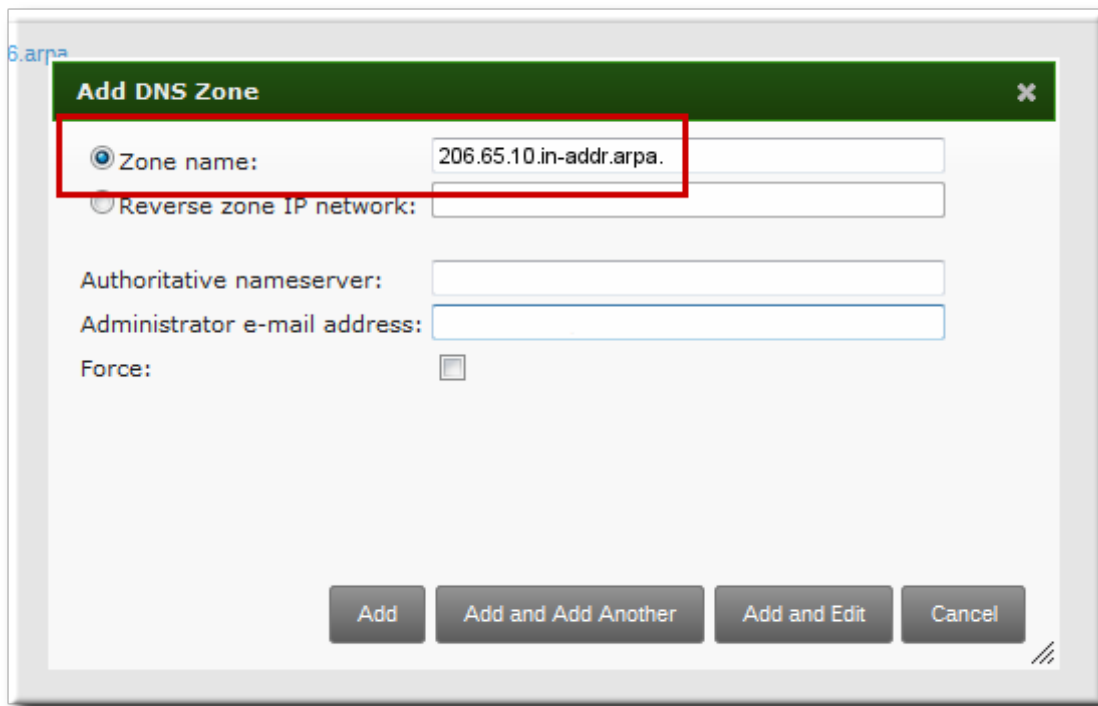


Figure 16.1. Creating a Reverse Zone by Name

With the command-line tools, the zone is created by name like this:

```
[bjensen@server ~]$ kinit  
[bjensen@server]$ ipa dnszone-add 206.65.10.in-addr.arpa.
```

To create the reverse zone by its IP network, set the network information to the (forward-style) IP address, with the subnet mask bit count. The bit count must be a multiple of eight for IPv4 addresses or a multiple of four for IPv6 addresses.

In the web UI, this is set in the **Reverse zone IP network** field.

Figure 16.2. Creating a Reverse Zone by IP Network

With the command-line tools, the zone is created by IP network like this:

```
[bjensen@server ~]$ kinit
[bjensen@server]$ ipa dnszone-add 10.65.206.0/24
```

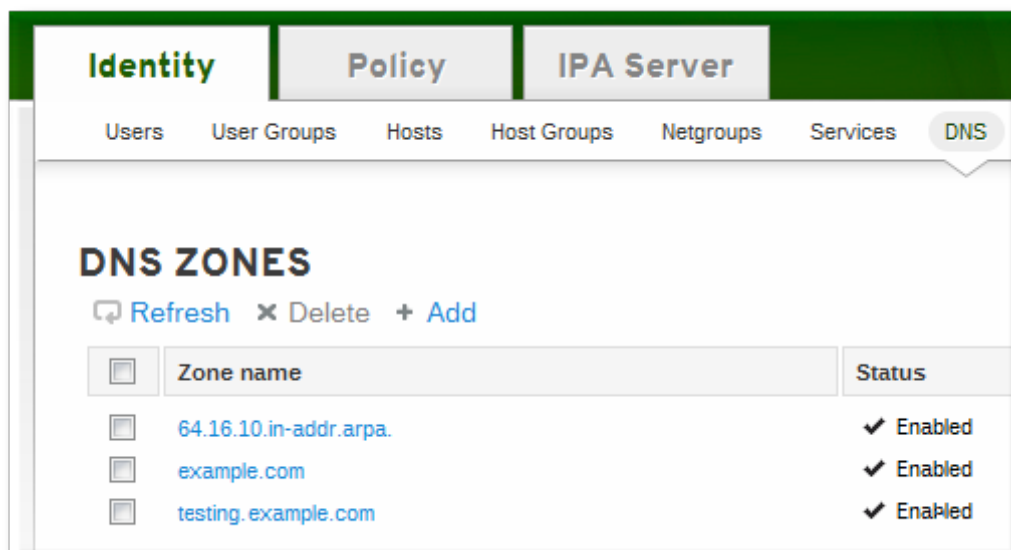
16.7.4. Enabling and Disabling Zones

Active zones can have clients added to them, are available for lookups, and are used by IdM services like Kerberos. Deleting a DNS zone removes the zone entry and all the associated configuration.

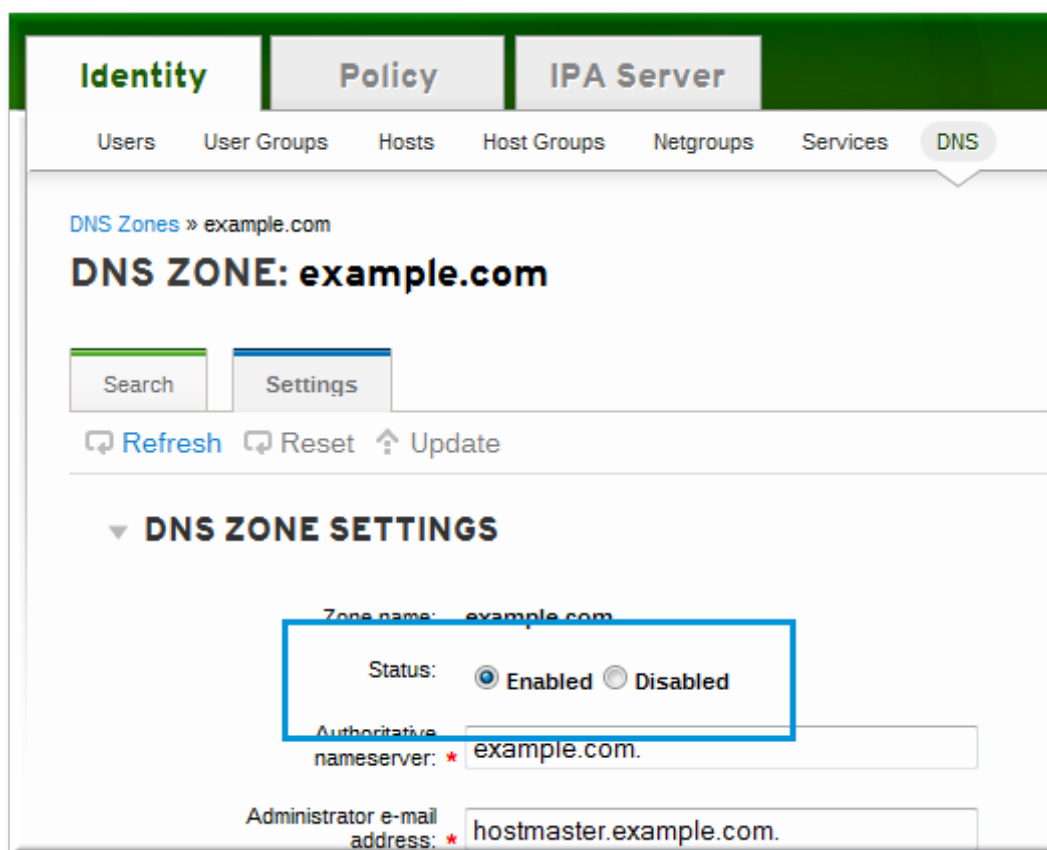
There can be situations when it is necessary to remove a zone from activity without permanently removing the zone. This is done by *disabling* the zone.

16.7.4.1. Disabling Zones in the Web UI

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone to edit.



3. Open the **Settings** tab.
4. Scroll down to the **Active zone** field. To disable the zone, set the value to **Disabled**.



5. Click the **Update** link at the top of the settings page.

16.7.4.2. Disabling Zones in the Command Line

Disabling a zone is done by using the **dnszone-disable** command.

For example:

```
$ ipa dnszone-disable server.example.com
```

When the zone needs to be brought back online, it can be re-enabled using the **dnszone-enable**

command.

16.8. Managing DNS Record Entries

16.8.1. Adding Records to DNS Zones

IdM supports several different types of DNS records, listed in [Table 16.2, “DNS Record Types”](#).

Table 16.2. DNS Record Types

A	CERT	KX	NS	SIG
AAAA	CNAME	LOC	NSEC	SRV
A6	DNAME	MX	PTR	SSHFP
AFSDB	DS	NAPTR	RRSIG	TXT

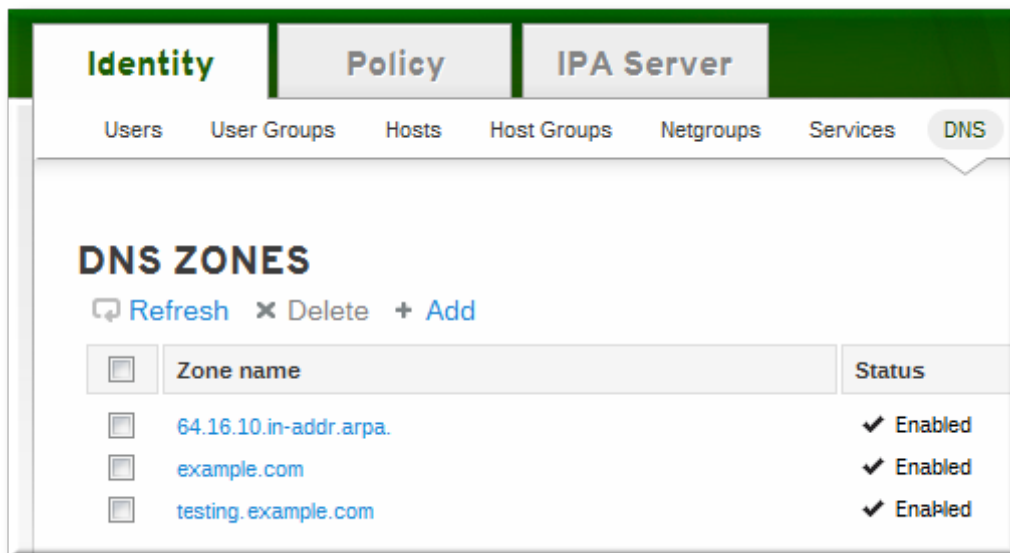
16.8.1.1. Adding DNS Resource Records from the Web UI



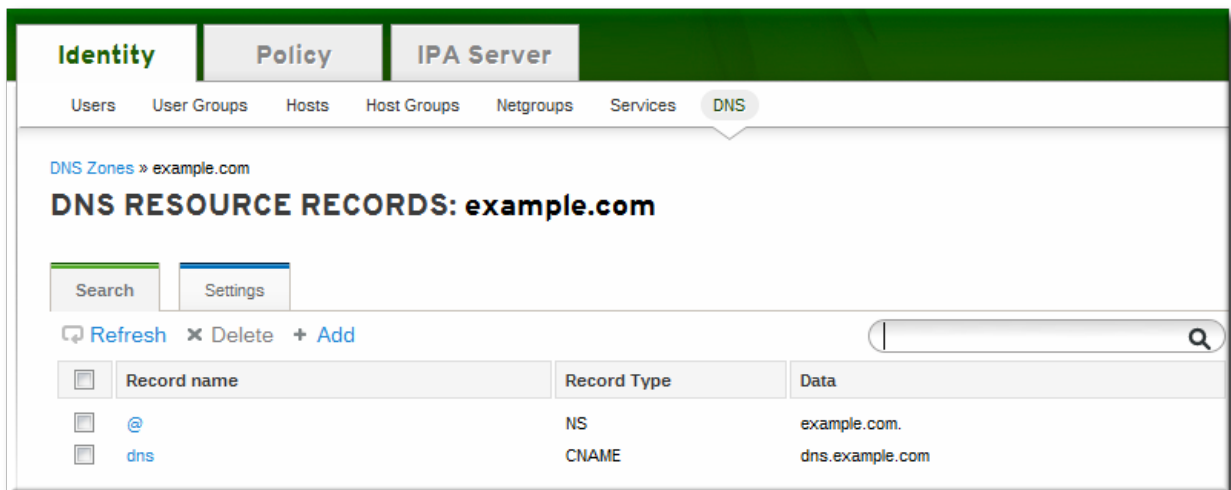
TIP

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the **named** service or configure the BIND service to poll the Directory Server automatically for zone changes. See [Section 16.9.2, “Enabling Zone Refreshes and Persistent Searches”](#).

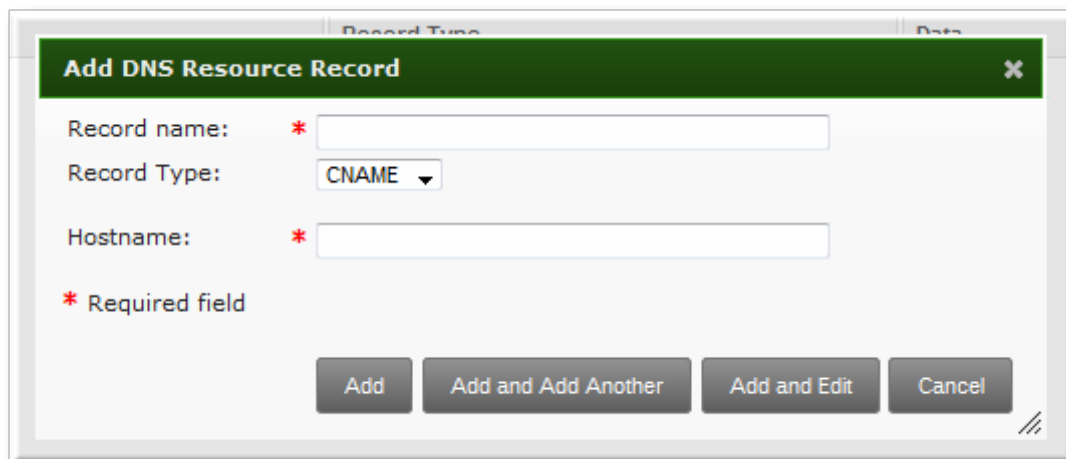
1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone to which to add records.



3. In the **DNS Resource Records** tab, click the **Add** link.



4. Select the type of record to create in the **Record Type** drop-down menu. The required data is different, depending on the record type. For example, a CNAME record requires a hostname. The data field name updates automatically to indicate what kind of information to supply.



Although IdM supports many different record types, there are four frequent record types that are used:

- » **A.** This is a basic map for a hostname and an ordinary IPv4 address. The **Record Name** is a hostname, such as **www**. The **IP Address** value is a standard IPv4 address, such as **192.168.1.2**.

More information about A records is in [RFC 1035](#).

- » **AAAA.** This is a basic map for a hostname and an IPv6 address. The **Record Name** is a hostname, such as **www**. The **IP Address** value is a standard hexadecimal IPv6 address, such as **fe80::20c:29ff:fe02:a1b3**.

More information about AAAA records is in [RFC 3596](#).

- » **SRV.** *Service (SRV) resource records* map service names to the DNS name of the server that is providing that particular service. The **Record Name** has the format **_service._protocol**, such as **_ldap._tcp**. There are individual fields to set the priority, weight, port number, and hostname for the target service.

More information about SRV records is in [RFC 2782](#).

- » **PTR.** A pointer record type (PTR) record adds a *reverse* DNS record, which maps an IP address to a domain name. In this case, the **Record Name** is the record ID number for the DNS entry of the resource and the **Hostname** value is the hostname with a terminal period, such as **server.example.com.**

More information about PTR records is in [RFC 1035](#).

5. Click the **Add** button to save the new resource record.

16.8.1.2. Adding DNS Resource Records from the Command Line

The same script, **ipa dnsrecord-add**, is used to add resource records of any type, but the options for the script and the required data are different, based on the resource record type.

16.8.1.2.1. About the Commands to Add DNS Records

The **ipa dnsrecord-add** command adds records to DNS zones, based on the type. Adding a record has the same basic command format:

```
$ ipa dnsrecord-add zoneName recordName --recordType-option=data
```

The *zoneName* is the name of the DNS zone to which the record is being added. The *recordName* is an identifier for the new DNS resource record.

[Table 16.3, “Common dnsrecord-add Options”](#) lists options for the most common resource record types: A (IPv4), AAAA (IPv6), SRV, and PTR. Options for other supported record types are listed in the **ipa dnsrecord-add** help and manpages.



NOTE

The **ipa dnsrecord-add** command only creates forward entries, not reverse entries.

Table 16.3. Common dnsrecord-add Options

General Record Options	
Option	Description
<code>--ttl=<i>number</i></code>	Sets the time to live for the record.
<code>--class=IN CS CH HS</code>	Sets the class of the record. This is usually IN, for Internet protocol.
<code>--structured</code>	Parses the raw DNS records and returns them in a structured format.
"A" Record Options	
Option	Description
<code>--a-rec=ARECORD</code>	Passes a comma-separated list of A records.
<code>--a-ip-address=<i>string</i></code>	Gives the IP address for the record.
"AAAA" Record Options	
Option	Description
<code>--aaaa-rec=AAAAARECORD</code>	Passes a comma-separated list of AAAA (IPv6) records.
<code>--aaaa-ip-address=<i>string</i></code>	Gives the IPv6 address for the record.
"PTR" Record Options	
Option	Description
<code>--ptr-rec=PTRRECORD</code>	Passes a comma-separated list of PTR records.
<code>--ptr-hostname=<i>string</i></code>	Gives the hostname for the record.
"SRV" Record Options	
Option	Description
<code>--srv-rec=SRVRECORD</code>	Passes a comma-separated list of SRV records.
<code>--srv-priority=<i>number</i></code>	Sets the priority of the record. There can be multiple SRV records for a service type. The priority (0 - 65535) sets the rank of the record; the lower the number, the higher the priority. A service has to use the record with the highest priority first.
<code>--srv-weight=<i>number</i></code>	Sets the weight of the record. This helps determine the order of SRV records with the same priority.
<code>--srv-port=<i>number</i></code>	Gives the port for the service on the target host.
<code>--srv-target=<i>string</i></code>	Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.

16.8.1.2.2. Examples of Adding DNS Resource Records

**TIP**

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the **named** service or configure the BIND service to poll the Directory Server automatically for zone changes. See [Section 16.9.2, “Enabling Zone Refreshes and Persistent Searches”](#).

Example 16.3. IPv4 Record

Type A resource records map hostnames to IPv4 addresses. The *record* value for these commands, then, is a standard IPv4 address. The URL label is usually `www`.

```
$ ipa dnsrecord-add example.com www --a-rec 10.64.14.165
```

This creates the record **www.example.com** with the IP address 10.64.14.165.

More information about A records is in [RFC 1035](#).

Example 16.4. IPv6 Record

Type AAAA resource records (*quad-A records*) map hostnames to IPv6 addresses. The *record* value for these commands is an IPv6 address. As with Type A records, the URL label is usually `www`.

```
$ ipa dnsrecord-add example.com www --aaaa-rec fe80::20c:29ff:fe02:a1b3
```

This creates the record **www.example.com** with the IP address fe80::20c:29ff:fe02:a1b3. More information about AAAA records is in [RFC 3596](#).

Example 16.5. SRV Record

Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the DNS server which manages it.

As with Type A and Type AAAA records, SRV records specify a way to connect to and identify the service, but the record format is different.

The *recordName* identifies the service type and the connection protocol, in the format `_service._protocol`.

The *record* information has the format *"priority weight port target"*.

```
$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="0 100 389
server1.example.com."
```

```
$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="1 100 389
server2.example.com."
```

More information about SRV records is in [RFC 2782](#).

Example 16.6. PTR Record

A pointer record type (PTR) record adds a *reverse* DNS record, which maps an IP address to a domain name, rather than the other way around.

All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the **in-addr.arpa.** domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the **in-addr.arpa.** domain appended to it. For example, for the IP address **192.0.1.2**, the reverse address is **2.1.0.192.in-addr.arpa.**

When adding the reverse DNS record, the format of the **dnsrecord-add** command is also reverse, compared to the usage for adding regular DNS entries:

```
$ ipa dnsrecord-add reverseIpAddress recordId --ptr-rec FQDN
```

The *recordId* is the numeric identifier to use for the entry in the zone.

For example, this adds a record with an ID of 4 for **server2.example.com**:

```
$ ipa dnsrecord-add 2.1.0.192.in-addr.arpa 4 --ptr-rec server2.example.com.
```

More information about PTR records is in [RFC 1035](#).



NOTE

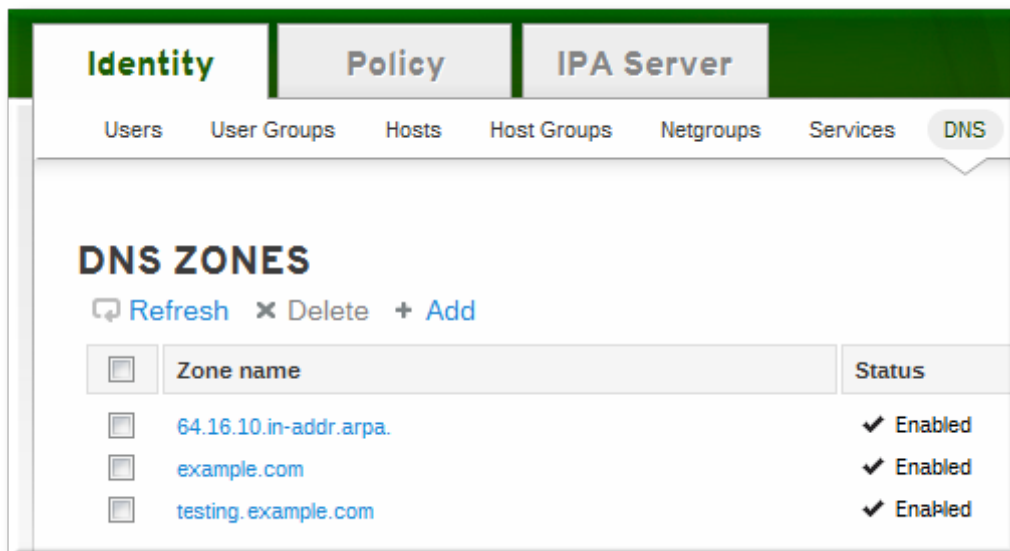
Reverse zones can also be configured for IPv6 addresses, with zones in the **.ip6.arpa.** domain. For more information about IPv6 reverse zones, see [RFC 3596](#).

16.8.2. Deleting Records from DNS Zones

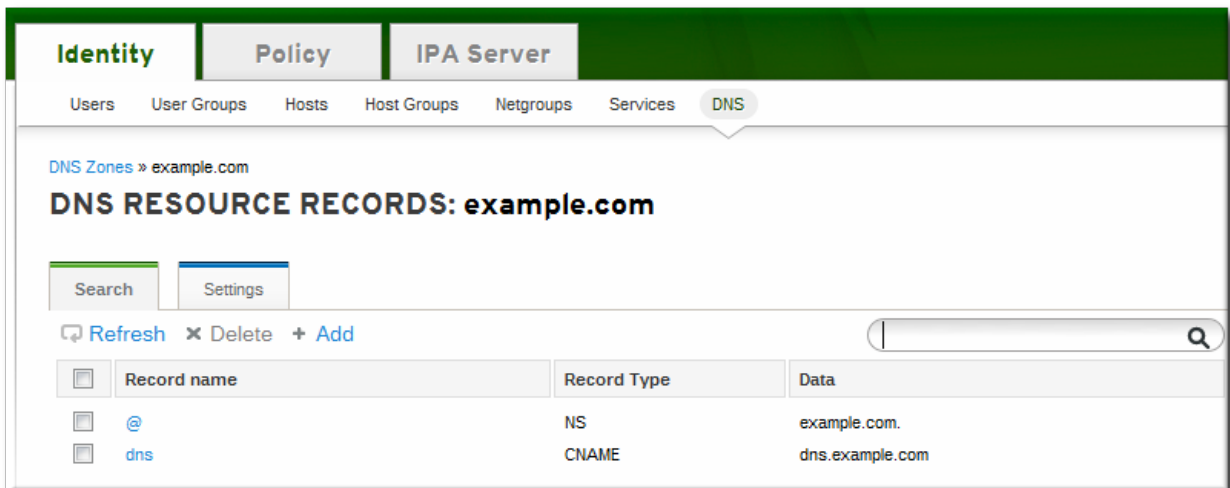
16.8.2.1. Deleting Records with the Web UI

To delete only a specific record from the resource record:

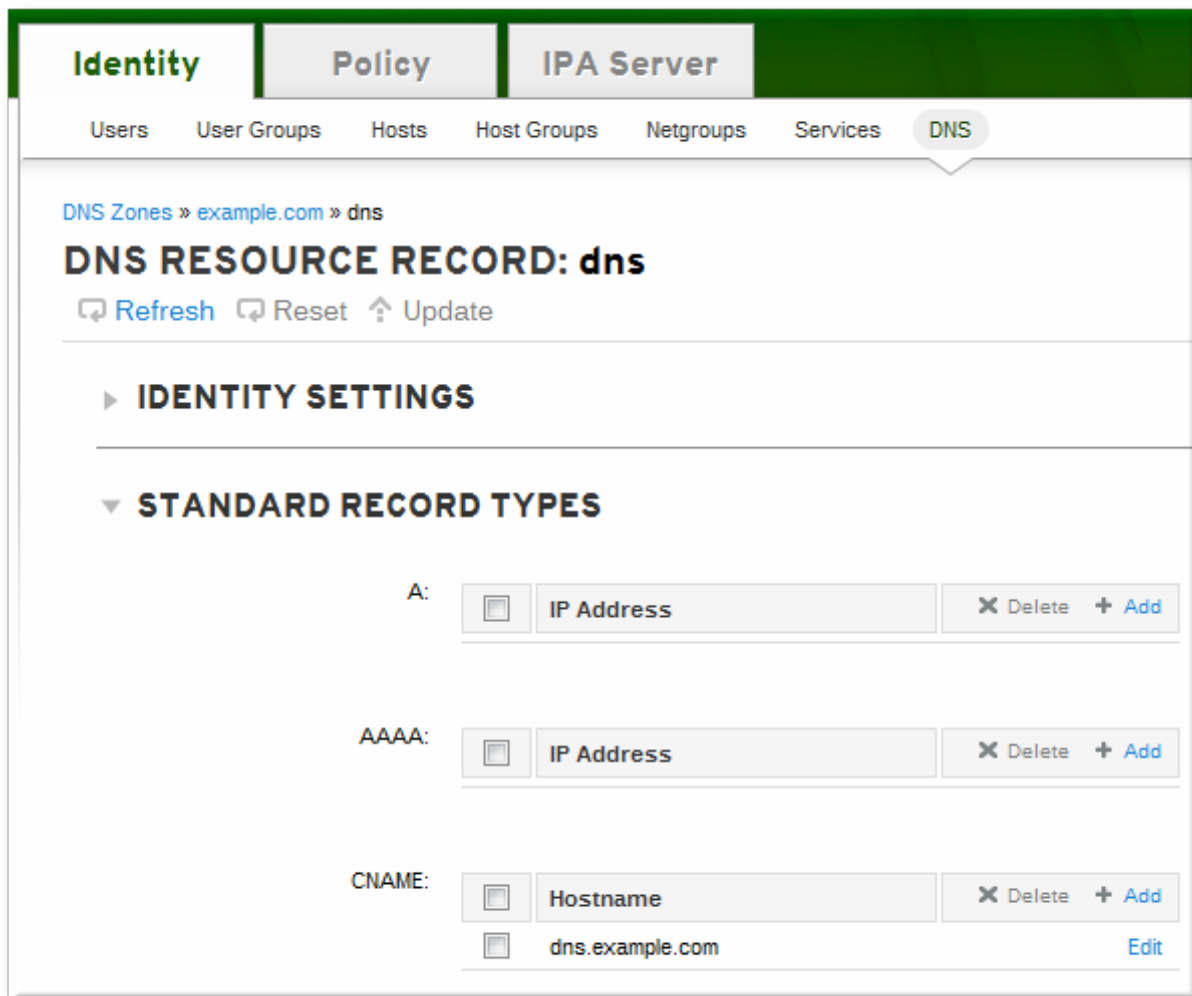
1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone.



3. In the **DNS Resource Records** tab, click the name of the resource record.



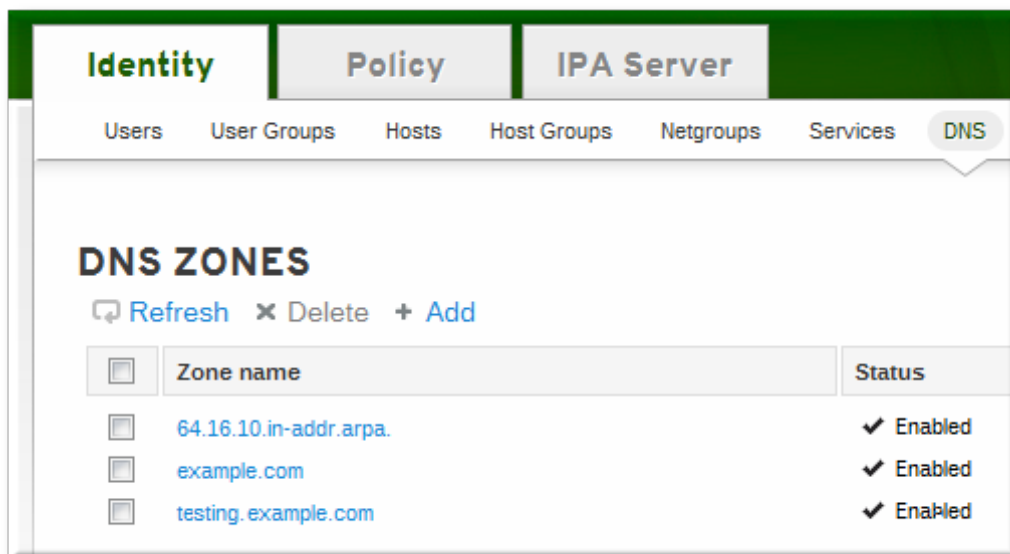
4. Click the checkbox by the name of the record type to delete, and then click the active **Delete** link at the top of the list.



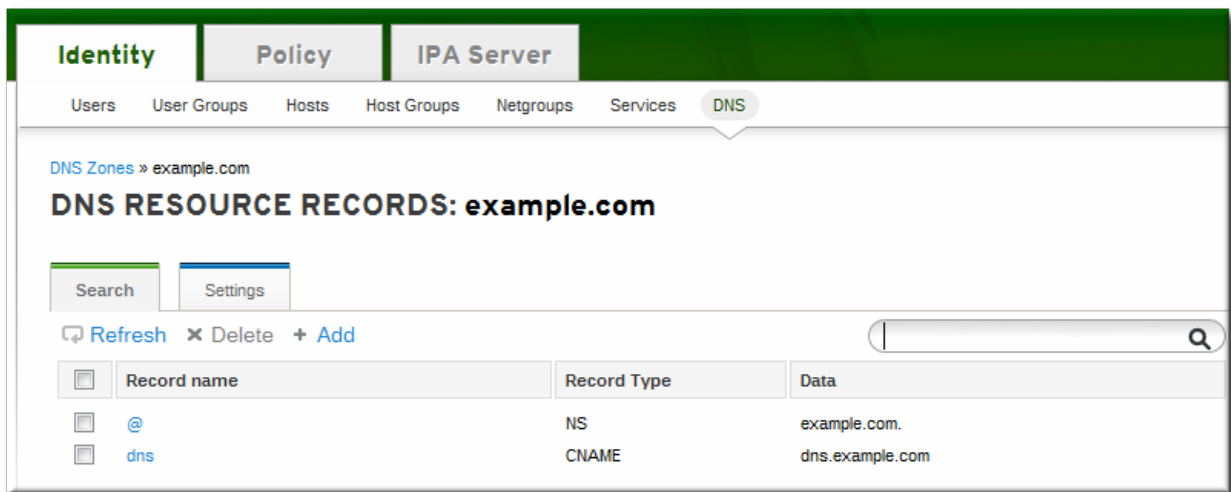
This deletes only that record type while leaving the other configuration intact.

Alternatively, delete all of the records for the resource in the zone:

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone.



3. In the **DNS Resource Records** tab, select the checkbox by the name of the resource record to delete. This deletes the entire record.



4. Click the **Delete** link at the top of the zone records page.

16.8.2.2. Deleting Records with the Command Line

Records are removed from the zone using the **ipa dnsrecord-del** command. As with adding records, records are deleted using an option that specifies the type of record (**--recordType-rec**) and the record value.

For example, to remove the A type record:

```
$ ipa dnsrecord-del example.com www --a-rec 10.64.14.213
```

If you run the **ipa dnsrecord-del** command without any options, the command prompts for information about the record to delete.

Alternatively, using the **--del-all** option removes all associated records for the zone.

16.9. Configuring the bind-dyndb-ldap Plug-in

The **bind-dyndb-ldap** system plug-in contains a DNS record cache for zones and a history of successful DNS resolutions. Maintaining the cache improves lookup performance in the Directory Server because it is not necessary to query the directory services every time there is a new DNS request.

When this plug-in is installed and IdM is configured to manage DNS, then a new configuration section is added to the plug-in configuration.

Example 16.7. Default dynamic-db Configuration

```
dynamic-db "ipa" {
    library "ldap.so";
    arg "uri ldapi://%2fvar%2frun%2fslapd-EXAMPLE-COM.socket";
    arg "base cn=dns, dc=example, dc=com";
    arg "fake_mname server.example.com.";
    arg "auth_method sasl";
    arg "sasl_mech GSSAPI";
    arg "sasl_user DNS/server.example.com";
};
```

This configuration uses implied default values for *other* plug-in behaviors, like how long it maintains the

cache. The assumed, default configuration can be changed by adding arguments to the **dynamic-db "ipa"** entry.

```
arg "argument value";
```

The additional parameters are listed in [Table 16.4, “Additional bind-dyndb-ldap Configuration Parameters”](#).



NOTE

Both cache updates and new zone detection can be forced by reloading the name server:

```
# rndc reload
```

Table 16.4. Additional bind-dyndb-ldap Configuration Parameters

Parameter	Description	Default Value
cache_ttl	Checks the DNS configuration in the Directory Server for new zones.	120 (seconds); this is defined in the <i>bind-dyndb-ldap</i> plug-in.
zone_refresh	Checks frequency, in seconds, that the server checks the DNS configuration in the Directory Server for new zones.	60 (seconds); this is configured by ipa-dns-install . If this is not set in the <i>/etc/named.conf</i> file, the <i>bind-dyndb-ldap</i> plug-in sets this value to zero (0), which disables zone refresh.
psearch	Enables persistent searches for the Directory Server so the BIND service immediately receives an update notification when a new DNS zone is added.	no

16.9.1. Changing the DNS Cache Setting

To improve DNS performance, it may be necessary to change the cache setting. By default, DNS records are kept in cache and considered valid for 120 seconds. This means that if a DNS record changes, it will not (necessarily) be propagated to the name server for up to 120 seconds. If the Directory Server has a high traffic volume or if records do not change frequently, then the cache time can be increased to improve performance by adding the **cache_ttl** parameter.

```
dynamic-db "ipa" {
...
    arg "cache_ttl 1800";
};
```

16.9.2. Enabling Zone Refreshes and Persistent Searches

The DNS service receives its information through the **bind-dyndb-ldap** plug-in. The plug-in resolves

only zones which were configured and enabled in the Directory Server when the name server started. When the name service restarts, the plug-in reloads its configuration and identifies any new zones or any new resource records.

However, the **bind-dyndb-ldap** plug-in pulls zone and resource record information from the IdM LDAP directory, and it is possible to pull information from that directory apart from simply restarting the plug-in. The **bind-dyndb-ldap** plug-in search for zone changes actively either by refreshing the zone data or by keeping a persistent connection open to the Directory Server and immediately catching any changes.

Periodically checking for new zones is the same as *refreshing* the zone configuration. This is set in the **zone_refresh** argument.

```
dynamic-db "ipa" {
...
    arg "zone_refresh 30";
};
```

Alternatively, the plug-in can maintain an open connection to the server through a *persistent search*. Persistent searches provide immediate notification of changes, unlike polling, and maintain a local cache of the configuration data.



NOTE

A persistent search catches updates both to zones and to zone resource records.

Persistent searches are disabled by default but can be enabled in the **psearch** argument:

```
dynamic-db "ipa" {
...
    arg "psearch yes";
};
```

Because persistent searches leave an ongoing, open connection with the Directory Server, there can be some performance issues. Performance implications are covered in the [Red Hat Directory Server Administrator's Guide](#).

16.10. Changing Recursive Queries Against Forwarders

The **ipa-client-install** script sets a configuration statement in the **/etc/named.conf** file that allows name resolution against hosts that are outside the IdM DNS domain. (This requires that the IdM server be set up with DNS configured and with forwarders configured.) What this means is that any host is permitted to issue recursive queries against configured forwarders.

By default, any host is permitted to issue recursive queries against configured forwarders. The IdM installation script automatically adds a line to the **/etc/named.conf** file to allow these recursive queries.

```
forward first;
forwarders { 10.16.36.29; };
allow-recursion { any; };
```


This behavior can be changed in the **allow-recursion** statement.

1. Open the `/etc/named.conf` file.
2. Reset the **allow-recursion** statement. This is set to **any** by default, which allows all hosts to resolve names against all forwarders.

```
forward first;
forwarders { 10.16.36.29; };
allow-recursion { any; };
```

3. Restart the **named** service.

```
service named restart
```

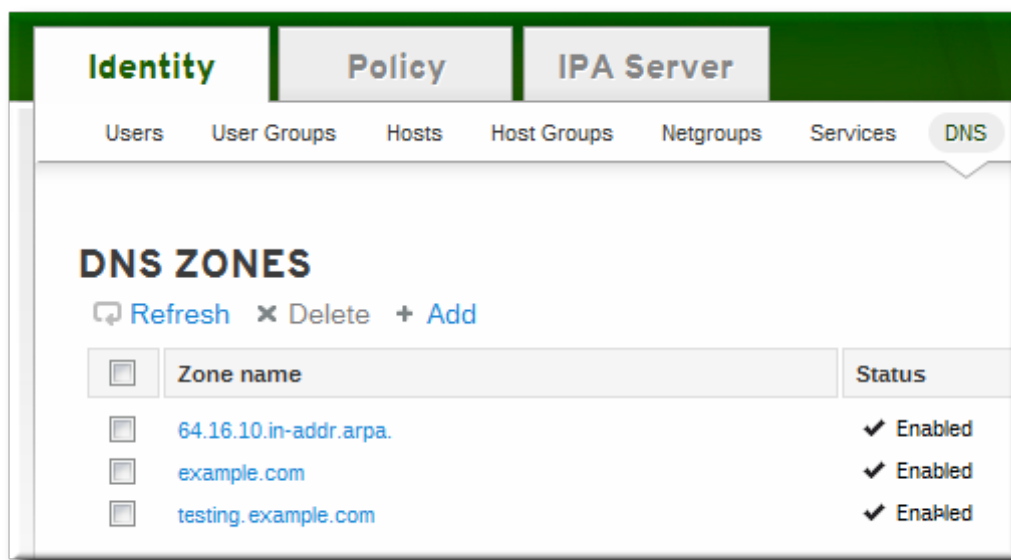
The name server documentation has more details on editing configuration statements.

16.11. Enabling Dynamic DNS Updates

Dynamic DNS updates are not enabled by default for new DNS zones in IdM. If dynamic updates are not allowed, then it may not be possible for the **ipa-client-install** script to join a client to the domain because it cannot add a DNS record pointing to the new client.

16.11.1. Enabling Dynamic DNS Updates in the Web UI

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone to edit.



3. Open the **Settings** tab.
4. Scroll down to the **Dynamic update** field, and set the value to True.

The screenshot shows the Identity Management web interface. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Below these are sub-tabs for 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', 'Services', and 'DNS'. The 'DNS' tab is selected, and the breadcrumb 'DNS Zones » example.com' is visible. The main heading is 'DNS ZONE: example.com'. Below this are 'Search' and 'Settings' tabs, with 'Settings' being the active one. Action links 'Refresh', 'Reset', and 'Update' are present. A section titled 'DNS ZONE SETTINGS' contains various configuration fields: 'Zone name' (example.com), 'Status' (radio buttons for 'Enabled' and 'Disabled', with 'Enabled' selected), 'Authoritative nameserver' (example.com.), 'Administrator e-mail address' (hostmaster.example.com.), 'SOA serial' (2012130201), 'SOA refresh' (3600), 'SOA retry' (900), 'SOA expire' (1209600), 'SOA minimum' (3600), 'SOA time to live' (empty), and 'SOA class' (dropdown menu). A blue box highlights the 'Dynamic update' section, which has radio buttons for 'True' and 'False', with 'False' selected. Below this is a 'BIND update policy' field.

5. Click the **Update** link at the top of the settings page.

16.11.2. Enabling Dynamic DNS Updates in the Command Line

To allow dynamic updates to the DNS zones, set the `--dynamic-update` option.

```
$ ipa dnszone-mod server.example.com --dynamic-update=TRUE
```

16.12. Configuring Forwarders and Forward Policy

A *DNS forwarder* is a server which passes DNS queries on to another, external DNS name server for resolution. Within the IdM DNS domain, there are three configuration properties that define how forwarders are used:

- A list of global forwarders which are used by all zones in IdM
- A list of forwarders which are used by a single, specific zone (as part of the zone configuration)
- A policy which defines how the zone sends requests to the forwarders

16.12.1. Configuring Global Forwarders

Global forwarders are configured as part of the IdM server configuration itself. Forwarders are (optionally) set up when the server is installed with the **setup-dns** option or when the **ipa-dns-install** script is used.

After server configuration, the list of global forwarders can be edited using the **dnsconfig-mod** command. For example:

```
[jsmith@server ~]$ ipa dnsconfig-mod --forwarder=0.9.8.7
Global forwarders: 0.9.8.7
```

16.12.2. Configuring Zone Forwarders

Forwarders can be configured to be used with a specific DNS zone as part of the zone configuration. The **--forwarder** option can be used multiple times to create a list of forwarders to use with the zone.

For example:

```
[jsmith@server ~]$ ipa dnszone-mod --forwarder=192.0.2.0 --forwarder=198.51.100.0
example.com

Zone name: example.com
...
Zone forwarders: 192.0.2.0, 198.51.100.0
```



NOTE

DNS forwarders must be specified as IP addresses, not as hostnames.

16.12.3. Configuring Forwarder Policy for a Zone

Once forwarders are configured, there are different ways that the zone can use them to service requests.

The zone can use the forwarders only for servicing name resolution requests; this is called a *forward-only zone*. A forward-only zone does not check its own name records. Only the forwarder server records are checked. If the record does not exist on the configured forwarders, then the zone returns a negative response to the client.

Alternatively, the zone can check the forwarder records first, and then fallback on its own resource records. This has a *first* policy.

This configuration is set in the **--forward-policy** option, using a policy of either **only** or **first**. For example:

```
[jsmith@server ~]$ ipa dnszone-mod --forward-policy=only example.com

Zone name: example.com
...
Zone forwarders: 1.2.3.4;255.255.255.255
Forward policy: only
```

16.13. Enabling Zone Transfers

Name servers maintain authoritative data for the zones; as changes are made to the zones, those changes must be sent to and distributed among the name servers for the DNS domain. A *zone transfer* moves resource records from one name server to another. An *authoritative transfer* (AXFR) is a zone transfer which includes that authoritative data for the zone (as opposed to an incremental transfer, which only delivers the most immediate zone change).

Zone transfers are defined in [RFC 1034](#) and [RFC 5936](#).

Zone transfers can be enabled when the zone is created or when it is modified by using the **--allow-transfer** option to set a list of name servers to which the zone records can be transferred.

For example:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-transfer=255.255.255.255;0.0.0.0;1.2.3.4
example-zone
```

The default is **any**, which the zone to be transferred anywhere in the DNS domain.

Once it is enabled in the **bind** service, IdM DNS zones can be transferred, by name, by clients like **dig**:

```
[root@server ~]# dig @ipa-server zone_name AXFR
```

16.14. Defining DNS Queries

To resolve hostnames within the DNS domain, a DNS client issues a query to the DNS name server. For some security contexts or for performance, it may be advisable to restrict what clients can query DNS records in the zone.

DNS queries can be configured when the zone is created or when it is modified by using the **--allow-query** option to set a list of clients which are allowed to issue queries.

For example:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-query=255.255.255.255;0.0.0.0;1.2.3.4
example-zone
```

The default is **any**, which allows the zone to be queried by any client.

16.15. Synchronizing Forward and Reverse Zone Entries

Forward entries (A and AAAA) are configured separately from reverse entries (PTR). Because these entries are configured independently, it is possible for forward entries to exist without corresponding reverse entries, and vice versa.

A DNS zone can be configured to allow its forward and reverse entries to be synchronized automatically, by setting the **--allow-sync-ptr** option to **1**. This can be done when the zone is created or when it is edited.



NOTE

This is set on the *forward* zone server, not the reverse DNS server.

For example, for editing an existing entry:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-sync-ptr=1 example-zone
```

The default is **0**, which disables synchronization and has better server performance.

There are some DNS setting requirements for PTR synchronization to work:

- Both forward and reverse zones must be managed by the IdM server.
- Both zones must have dynamic updates enabled.
Enabling dynamic updates is covered in [Section 16.11, “Enabling Dynamic DNS Updates”](#).
- The PTR record will be updated only if the name of the requesting client matches the name in the PTR record.



IMPORTANT

Changes made through the IdM web UI, through the IdM command-line tools, or by editing the LDAP entry directly **do not** update the PTR record. Only changes made by the DNS service itself trigger PTR record synchronization.



WARNING

A client system can update its own IP address. This means that a compromised client can be used to overwrite PTR records by changing its IP address.

16.16. Setting DNS Access Policies

The IdM DNS domain can define access controls, based on grant/deny rules, for zones. This creates an **update-policy** statement in the **/etc/named.conf** file, which defines the DNS access rule.

```
--update-policy "grant|deny zoneName policyName recordName recordType"
```

- *zoneName* is the IdM DNS zone to which to apply the rule.
- *policyName* is the name to use for the BIND rule.
- *recordName* sets the resource records to which to apply the rule. Using an asterisk (*) is used for self rules.
- *recordType* is the record type the rule applies to. Update access rules are applied individually for each record type, even within the same DNS zone entry.

For example, to grant the **EXAMPLE.COM** zone the ability to edit its own A and AAAA resource record entries:

```
$ ipa dnszone-mod example.com --update-policy="grant EXAMPLE.COM krb5-self * A;  
grant EXAMPLE.COM krb5-self * AAAA;"
```



IMPORTANT

If the update policy is set to false, IdM client machines will not be able to add or update their IP address. See [Section 16.11, “Enabling Dynamic DNS Updates”](#) for more information.

16.17. Resolving Hostnames in the IdM Domain

It is possible to check the DNS entries for IdM domain members using the **dns-resolve** command. If the record exists and is properly formatted in the DNS configuration, then the command returns the DNS record. If not, the command returns an error, that the hostname is not recognized within the DNS service.

```
$ipa dns-resolve server1.example.com
```

This can be helpful with troubleshooting connection problems between servers, clients, and services.

[7] Any updated schema files, included updated DNS schema elements, are located in the **/usr/share/ipa/updates** directory.

Chapter 17. Policy: Using Automount

Automount is a way of making directories on different servers available, automatically, when requested by a user. This works exceptionally well within an IdM domain since it allows directories on clients within the domain to be shared easily. This is especially important with user home directories ([Section 9.1, “Setting up User Home Directories”](#)).

In IdM, automount works with the internal LDAP directory and, if it is configured, DNS services.

17.1. About Automount and IdM

Automount is a way to manage, organize, and access directories across multiple systems. Automount automatically mounts a directory whenever that resource is requested. Automount also provides a coherent structure to the way that these directories are organized. Every single directory, or *mount point* is called a *key*. Multiple keys that are grouped together are a *map*, and maps are associated according to their physical or conceptual *location*.

The base configuration file for autofs is the **auto.master** file in the **/etc/** directory. There can be multiple **auto.master** configuration files in separate server locations, if necessary.

When **autofs** is configured on a server and that server is a client in an IdM domain, then all of the configuration information for automount is stored in the IdM directory. Rather than being stored in separate text files, the autofs configuration — maps, locations, and keys — are stored as LDAP entries. For example, the default map file, **auto.master**, is stored as:

```
dn: automountmapname=auto.master,cn=default,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master
```



IMPORTANT

Identity Management does not set up or configure autofs. That must be done separately. Identity Management works with an existing autofs deployment.

Each new location is added as a container entry under **cn=automount,dc=example,dc=com**, and each map and each key are stored beneath that location.

As with other IdM domain services, automount works with IdM natively. The automount configuration can be managed by IdM tools:

- *Locations*, using **ipa automountlocation*** commands
- Both direct and indirect *maps*, using **ipa automountmap*** commands
- *Keys*, using **ipa automountkey*** commands

For automount to work within the IdM domain, the NFS server must be configured as an IdM client. Configuring NFS itself is covered in the [Red Hat Enterprise Linux Storage Administration Guide](#).

17.2. Configuring Automount

**IMPORTANT**

Identity Management does not set up or configure autofs. That must be done separately, as described in these procedures. Identity Management works with an existing autofs deployment.

**TIP**

Test that the `/home` directory can be mounted from the command line successfully before changing the automount configuration. Making sure that NFS is already working properly makes it easier to troubleshoot any potential IdM automount configuration errors later.

17.2.1. Configuring autofs on Red Hat Enterprise Linux

1. Edit the `/etc/sysconfig/autofs` file to specify the schema attributes that autofs searches for:

```
#
# Other common LDAP naming
#
MAP_OBJECT_CLASS="automountMap"
ENTRY_OBJECT_CLASS="automount"
MAP_ATTRIBUTE="automountMapName"
ENTRY_ATTRIBUTE="automountKey"
VALUE_ATTRIBUTE="automountInformation"
```

2. Specify the LDAP configuration. There are two ways to do this. The simplest is to let the automount service discover the LDAP server and locations on its own:

```
LDAP_URI="ldap:///dc=example,dc=com"
```

Alternatively, explicitly set which LDAP server to use and the base DN for LDAP searches:

```
LDAP_URI="ldap://ipa.example.com"
SEARCH_BASE="cn=location,cn=automount,dc=example,dc=com"
```

**Note**

The default value for *location* is **default**. If additional locations are added ([Section 17.4, “Configuring Locations”](#)), then the client can be pointed to use those locations, instead.

3. Edit the `/etc/autofs_ldap_auth.conf` file so that autofs allows client authentication with the IdM LDAP server.
 - Change ***authrequired*** to **yes**.
 - Set the principal to the Kerberos host principal for the NFS client server, `host/fqdn@REALM`. The principal name is used to connect to the IdM directory as part of GSS client authentication.


```
<autofs_ldap_sasl_conf
    usetls="no"
    tlsrequired="no"
    authrequired="yes"
    authtype="GSSAPI"
    clientprinc="host/server.example.com@EXAMPLE.COM"
/>
```

If necessary, run **klist -k** to get the exact host principal information.

4. Check the **/etc/nsswitch.conf** file, so that LDAP is listed as a source for automount configuration:

```
automount: files ldap
```

5. Restart autofs:

```
# service autofs restart
```

6. Test the configuration by listing a user's **/home** directory:

```
# ls /home/userName
```

If this does not mount the remote file system, check the **/var/log/messages** file for errors. If necessary, increase the debug level in the **/etc/sysconfig/autofs** file by setting the **LOGGING** parameter to **debug**.



TIP

If there are problems with automount, then cross-reference the automount attempts with the 389 Directory Server access logs for the IdM instance, which will show the attempted access, user, and search base.

It is also simple to run automount in the foreground with debug logging on.

```
automount -f -d
```

This prints the debug log information directly, without having to cross-check the LDAP access log with automount's log.

17.3. Setting up a Kerberized NFS Server

Identity Management can be used to set up a Kerberized NFS server.



NOTE

The NFS server does not need to be running on Red Hat Enterprise Linux.

17.3.1. Setting up a Kerberized NFS Server

1. Obtain a Kerberos ticket before running IdM tools.

```
[jsmith@server ~]$ kinit admin
```

2. If the NFS host machine has not been added as a client to the IdM domain, then create the host entry. See [Section 10.2, “Adding Host Entries”](#).
3. Create the NFS service entry in the IdM domain. For example:

```
[jsmith@server ~]$ ipa service-add nfs/nfs-server.example.com
```

For more information, see [Section 11.1, “Adding and Editing Service Entries and Keytabs”](#).

4. Generate an NFS service keytab for the NFS server using the **ipa-getkeytab** command. The NFS server may be on a Red Hat Enterprise Linux machine in the IdM domain or a different Unix machine. For a Red Hat Enterprise Linux machine, the **ipa-getkeytab** command can be run on the NFS server machine. Otherwise, the **ipa-getkeytab** command should be run on a Red Hat Enterprise Linux machine in the IdM domain and then copied over to the NFS server. If **ipa-getkeytab** command is run on the NFS server, then save the keys directly to the host keytab. For example:

```
[jsmith@server ~]$ ipa-getkeytab -s server.example.com -p nfs/nfs-server.example.com -k /etc/krb5.keytab
```

For a Red Hat Enterprise Linux machine, that's all you need to do.



NOTE

Only DES keys are supported on Red Hat Enterprise Linux 5.

When generating keys to copy over to another system, then generate the key but do not save it in the host keytab. The key must be added separately to the keytab after it is copied to the NFS server:

- a. Save the keytab to a temporary file. For example:

```
[jsmith@server ~]$ ipa-getkeytab -s server.example.com -p nfs/nfs-server.example.com -k /tmp/nfs.keytab
```

- b. Copy the keytabs over to the NFS server.
- c. Set the file permissions to 0700.
- d. Add the service key to the keytab file.

```
[root@nfs-server ~]# ( echo rkt /tmp/nfs.keytab; echo wkt /etc/krb5.keytab ) | ktutil
```

**TIP**

Verify that the NFS service has been properly configured in IdM, with its keytab, by checking the service entry:

```
[jsmith@server ~]$ ipa service-show nfs/ipaclient2.example.com
Principal: NFS/ipaclient2.example.com@EXAMPLE.COM
Keytab: True
```

5. Install the NFS packages. For example:

```
[root@nfs-server ~]# yum install nfs-utils
```

6. Configure weak crypto support. This is required for every NFS client if *any* client (such as a Red Hat Enterprise Linux 5 client) in the domain will use older encryption options like DES.

- a. Edit the **krb5.conf** file to allow weak crypto.

```
[root@nfs-server ~]# vim /etc/krb5.conf

allow_weak_crypto = true
```

- b. Update the IdM server Kerberos configuration to support the DES encryption type.

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w
password -h ipaserver.example.com -p 389

dn: cn=EXAMPLEREALM,cn=kerberos,dc=example,dc=com
changetype: modify
add: krbSupportedEncSaltTypes
krbSupportedEncSaltTypes: des-cbc-crc:normal
-
add: krbSupportedEncSaltTypes
krbSupportedEncSaltTypes: des-cbc-crc:special
-
add: krbDefaultEncSaltTypes
krbDefaultEncSaltTypes: des-cbc-crc:special
```

7. Edit the NFS server configuration to use NFSv4 security by uncommenting the **SECURE_NFS** line.

```
[root@nfs-server ~]# vim /etc/sysconfig/nfs

SECURE_NFS="yes"
```

8. If the NFS server and client are in different DNS domains, then configure the NFS domain.

```
[root@nfs-server ~]# vim /etc/idmapd.conf

Domain = example.com
```

9. Edit the **/etc/exports** file and add the Kerberos information:

```
/export *(rw,sec=sys:krb5:krb5i:krb5p)
```

10. Restart the NFS server and related services.

```
[root@nfs-server ~]# service nfs restart
[root@nfs-server ~]# service nfs-server restart
[root@nfs-server ~]# service nfs-secure restart
[root@nfs-server ~]# service nfs-secure-server restart
[root@nfs-server ~]# service rpcsvcgssd restart
```

11. Configure the NFS server as an NFS client, following the directions in [Section 17.3.2, “Setting up a Kerberized NFS Client”](#).

17.3.2. Setting up a Kerberized NFS Client

1. Obtain a Kerberos ticket before running IdM tools.

```
[jsmith@server ~]$ kinit admin
```

2. If the NFS client is not enrolled as a client in the IdM domain, then set up the required host entries, as described in [Section 10.2, “Adding Host Entries”](#).
3. Generate an NFS service keytab for the NFS client using the **ipa-getkeytab** command.

The NFS client may be on a Red Hat Enterprise Linux machine in the IdM domain or a different Unix machine. For a Red Hat Enterprise Linux machine, the **ipa-getkeytab** command can be run on the NFS client machine. Otherwise, the **ipa-getkeytab** command should be run on a Red Hat Enterprise Linux machine in the IdM domain and then copied over to the NFS client.

If **ipa-getkeytab** command is run on the NFS client, then save the keys directly to the host keytab. For example:

```
[jsmith@server ~]$ ipa-getkeytab -k /etc/krb5.keytab -p host/nfs-client-
server.example.com@EXAMPLE.COM
```

For a Red Hat Enterprise Linux machine, that's all you need to do.

When generating keys to copy over to another system, then generate the key but do not save it in the host keytab. The key must be added separately to the keytab after it is copied to the NFS server:

- a. Save the keytab to a temporary file. For example:

```
[jsmith@server ~]$ ipa-getkeytab -p host/nfs-client-
server.example.com@EXAMPLE.COM -k /tmp/nfs.keytab
```

- b. Copy the keytabs over to the NFS client.
- c. Set the file permissions to 0700.
- d. Add the service key to the keytab file.

```
[root@nfs-client-server ~]# ( echo rkt /root/nfs-client.keytab; echo wkt
/etc/krb5.keytab ) | ktutil
```

4. Edit the NFS common configuration to enable client-side secure NFS, by uncommenting the **SECURE_NFS** line.

```
[root@nfs-client-server ~]# vim /etc/sysconfig/nfs

SECURE_NFS="yes"
```

5. If the NFS server and client are in different DNS domains, then configure the NFS domain. The **idmapd.conf** must be the same on the NFS client as it is on the NFS server.

```
[root@nfs-client-server ~]# vim /etc/idmapd.conf  
  
Domain = example.com
```

6. Start the GSS daemon.

```
[root@nfs-client-server ~]# service rpcgssd start  
[root@nfs-client-server ~]# service rpcbind start  
[root@nfs-client-server ~]# service rpcidmapd start
```

7. Mount the directory.

```
[root@nfs-client-server ~]# echo "$NFSSERVER:/this /mnt/this nfs4  
sec=krb5i,rw,proto=tcp,port=2049" >>/etc/fstab  
[root@nfs-client-server ~]# mount -av
```

17.4. Configuring Locations

A location is a set of maps, which are all stored in **auto.master**, and a location can store multiple maps. The location entry only works as a container for map entries; it is not an automount configuration in and of itself.

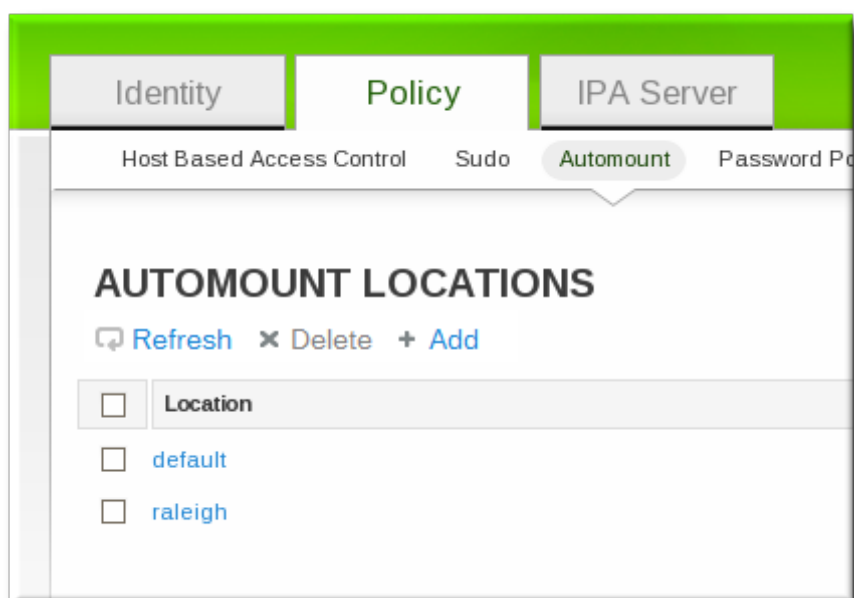


IMPORTANT

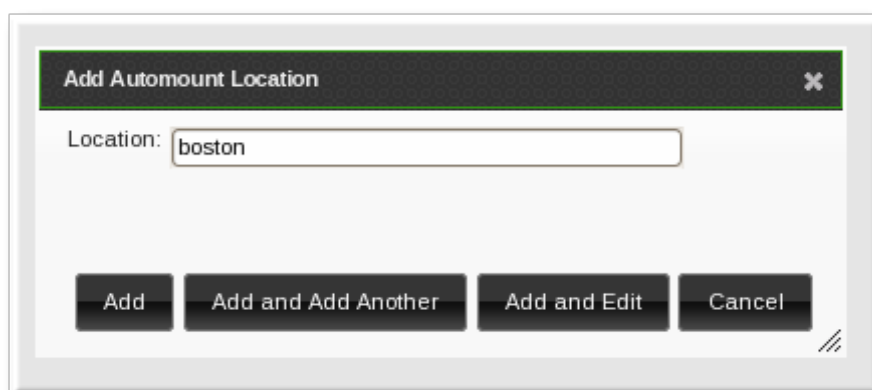
Identity Management does not set up or configure autofs. That must be done separately. Identity Management works with an existing autofs deployment.

17.4.1. Configuring Locations through the Web UI

1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click the **Add** link at the top of the list of automount locations.



4. Enter the name for the new location.



5. Click the **Add and Edit** button to go to the map configuration for the new location. Create maps, as described in [Section 17.5.1.1, “Configuring Direct Maps from the Web UI”](#) and [Section 17.5.2.1, “Configuring Indirect Maps from the Web UI”](#).

17.4.2. Configuring Locations through the Command Line

To create a map, using the **automountlocation-add** and give the location name.

```
$ ipa automountlocation-add location
```

For example:

```
$ ipa automountlocation-add raleigh
-----
Added automount location "raleigh"
-----
Location: raleigh
```

When a new location is created, two maps are automatically created for it, **auto.master** and **auto.direct**. **auto.master** is the root map for all automount maps for the location. **auto.direct** is the default map for direct mounts and is mounted on **/-**.

To view all of the maps configured for a location as if they were deployed on a filesystem, use the **automountlocation-tofiles** command:

```
$ ipa automountlocation-tofiles raleigh
/etc/auto.master:
/-          /etc/auto.direct
-----
/etc/auto.direct:
```

17.5. Configuring Maps

Configuring maps not only creates the maps, it associates mount points through the keys and it assigns mount options that should be used when the directory is accessed. IdM supports both direct and indirect maps.



NOTE

Different clients can use different map sets. Map sets use a tree structure, so maps *cannot* be shared between locations.



IMPORTANT

Identity Management does not set up or configure autofs. That must be done separately. Identity Management works with an existing autofs deployment.

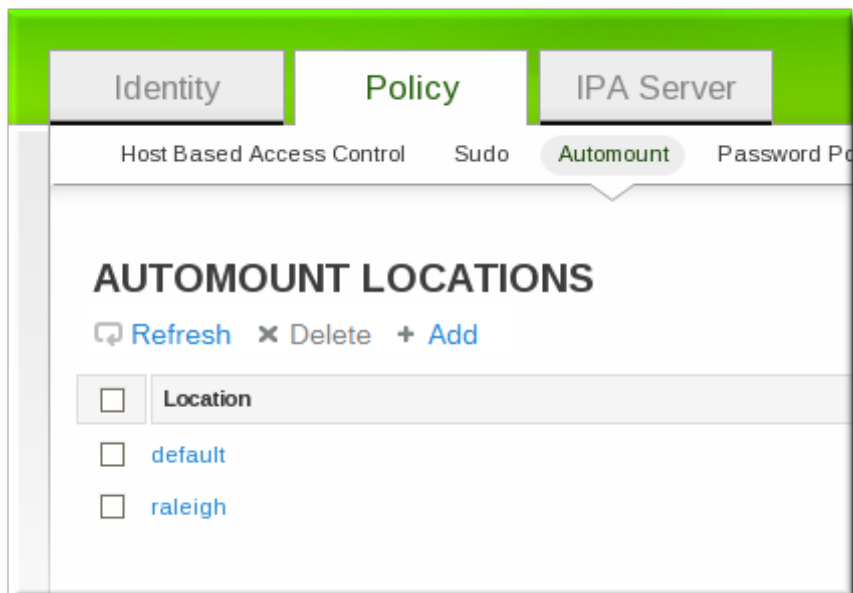
17.5.1. Configuring Direct Maps

Direct maps define exact locations, meaning absolute paths, to the file mount. In the location entry, a direct map is identified by the preceding forward slash:

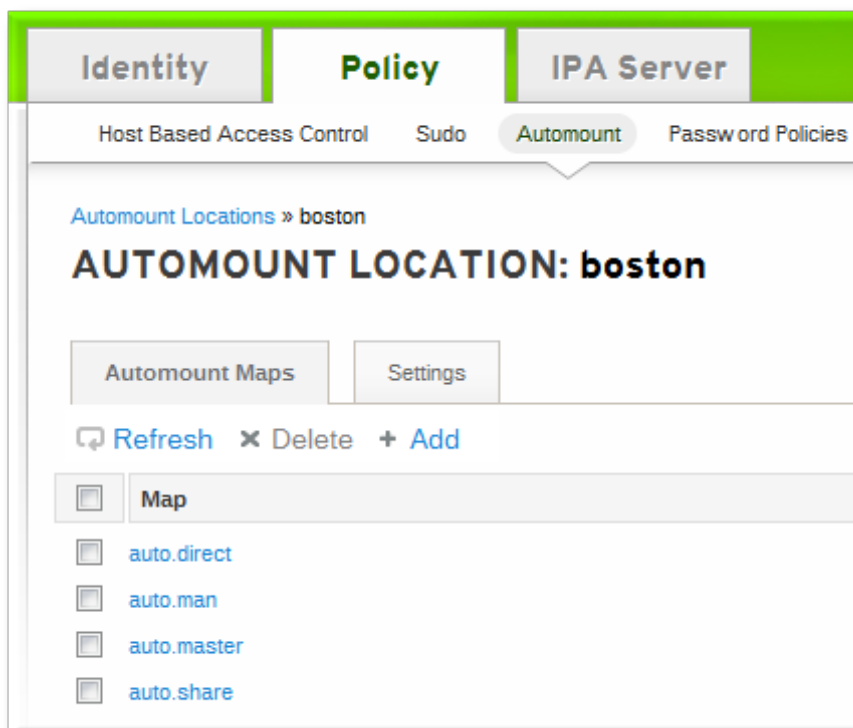
```
-----
/etc/auto.direct:
/shared/man server.example.com:/shared/man
```

17.5.1.1. Configuring Direct Maps from the Web UI

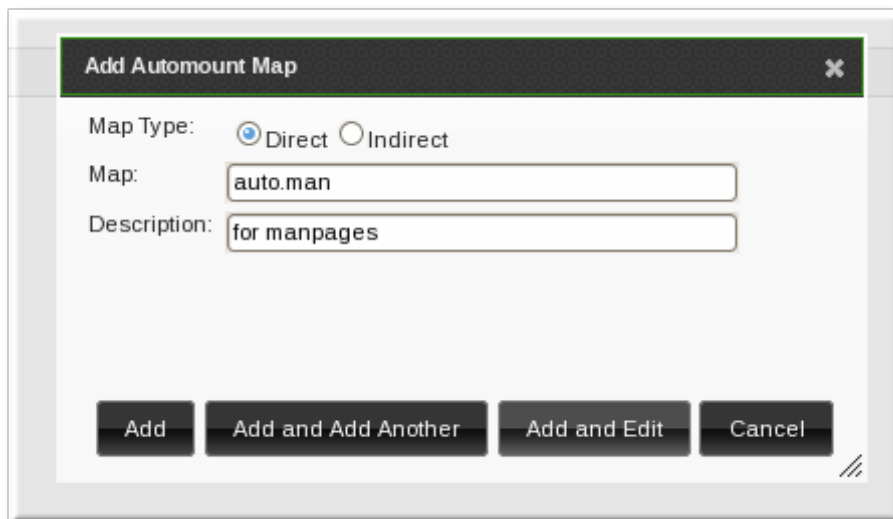
1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click name of the automount location to which to add the map.



4. In the **Automount Maps** tab, click the + **Add** link to create a new map.



5. In pop-up window, select the **Direct** radio button and enter the name of the new map.



Add Automount Map

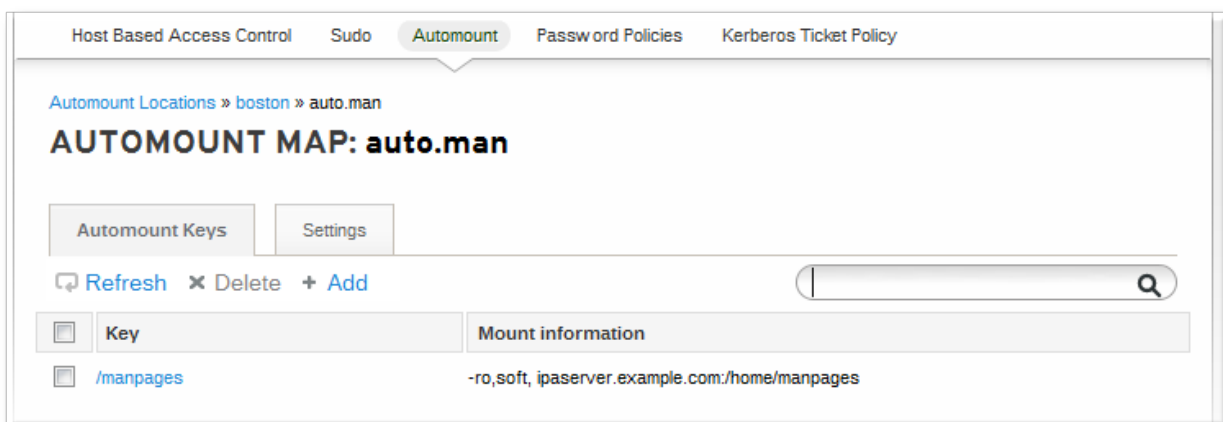
Map Type: ☒ Direct ☐ Indirect

Map:

Description:

Buttons: Add, Add and Add Another, Add and Edit, Cancel

6. In the **Automount Keys** tab, click the + **Add** link to create a new key for the map.



Host Based Access Control Sudo **Automount** Password Policies Kerberos Ticket Policy

Automount Locations » boston » auto.man

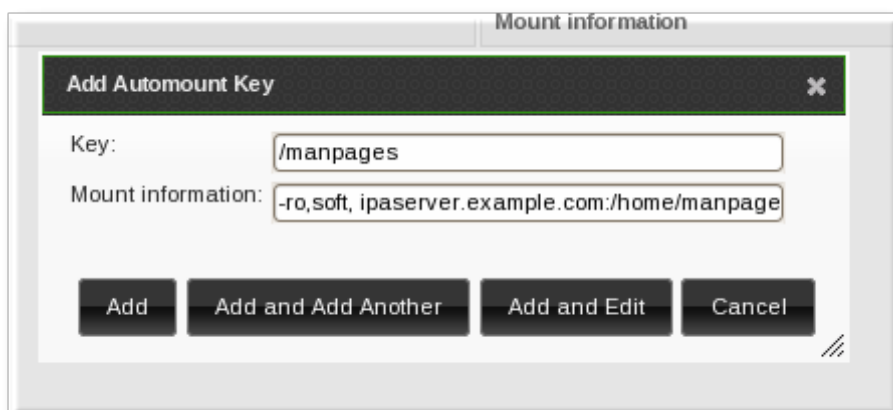
AUTOMOUNT MAP: auto.man

Automount Keys Settings

Refresh Delete Add

Key	Mount information
/manpages	-ro,soft, ipaserver.example.com:/home/manpages

7. Enter the mount point. The key defines the actual mount point in the key name. The **Info** field sets the network location of the directory, as well as any **mount** options to use.



Add Automount Key

Key:

Mount information:

Buttons: Add, Add and Add Another, Add and Edit, Cancel

8. Click the **Add** button to save the new key.

17.5.1.2. Configuring Direct Maps from the Command Line

The key defines the actual mount point (in the key name) and any options. A map is a direct or indirect map based on the format of its key.

Each location is created with an **auto.direct** item. The simplest configuration is to define a direct mapping by adding an automount key the existing direct map entry. It is also possible to create different direct map entries.

Add the key for the direct map to the location's **auto.direct** file. The **--key** option identifies the mount point, and **--info** gives the network location of the directory, as well as any **mount** options to use. For example:

```
$ ipa automountkey-add raleigh auto.direct --key=/share --info="-ro,soft,
ipaserver.example.com:/home/share"
Key: /share
Mount information: -ro,soft, ipaserver.example.com:/home/share
```

Mount options are described in the mount manpage, <http://linux.die.net/man/8/mount>.

On Solaris, add the direct map and key using the **ldapclient** command to add the LDAP entry directly:

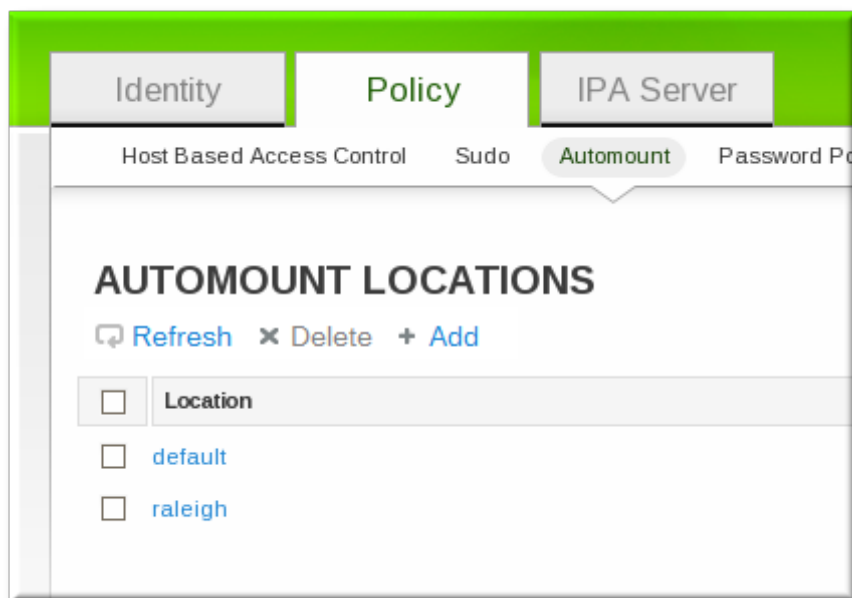
```
ldapclient -a
serviceSearchDescriptor=auto_direct:automountMapName=auto.direct,cn=location,cn=aut
omount,dc=example,dc=com?one
```

17.5.2. Configuring Indirect Maps

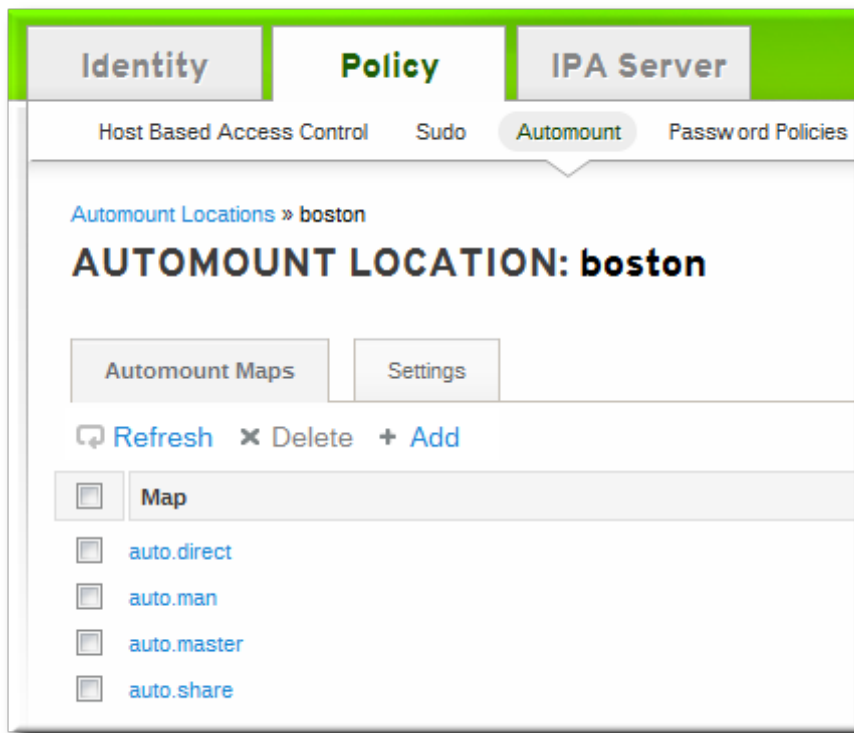
An indirect map essentially specifies a relative path for maps. A parent entry sets the base directory for all of the indirect maps. The indirect map key sets a sub directory; whenever the indirect map location is loaded, the key is appended to that base directory. For example, if the base directory is **/docs** and the key is **man**, then the map is **/docs/man**.

17.5.2.1. Configuring Indirect Maps from the Web UI

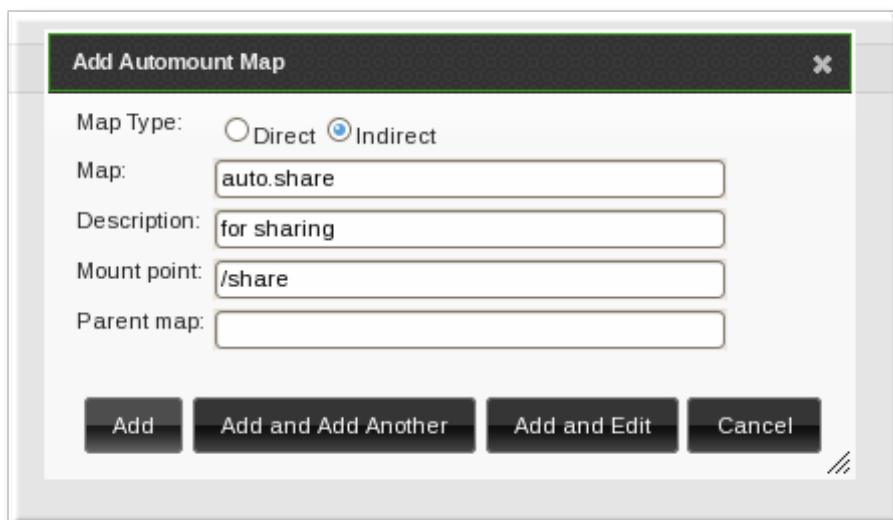
1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click name of the automount location to which to add the map.



4. In the **Automount Maps** tab, click the **+ Add** link to create a new map.



5. In pop-up window, select the **Indirect** radio button and enter the required information for the indirect map:



- » The name of the new map
- » The mount point. The **Mount** field sets the base directory to use for all the indirect map keys.
- » Optionally, a parent map. The default parent is **auto.master**, but if another map exists which should be used, that can be specified in the **Parent Map** field.

6. Click the **Add** button to save the new key.

17.5.2.2. Configuring Indirect Maps from the Command Line

The primary difference between a direct map and an indirect map is that there is no forward slash in front of an indirect key.

```
-----
/etc/auto.share:
man      ipa.example.com:/docs/man
-----
```

1. Create an indirect map to set the base entry using the **automountmap-add-indirect** command. The **--mount** option sets the base directory to use for all the indirect map keys. The default parent entry is **auto.master**, but if another map exists which should be used, that can be specified using the **--parentmap** option.

```
$ ipa automountmap-add-indirect location mapName --mount=directory [--parentmap=mapName]
```

For example:

```
$ ipa automountmap-add-indirect raleigh auto.share --mount=/share
-----
Added automount map "auto.share"
-----
```

2. Add the indirect key for the mount location:

```
$ ipa automountkey-add raleigh auto.share --key=docs --info="ipa.example.com:/export/docs"
-----
Added automount key "docs"
-----
Key: docs
Mount information: ipa.example.com:/export/docs
```

3. To verify the configuration, check the location file list using **automountlocation-tofiles**:

```
$ ipa automountlocation-tofiles raleigh
/etc/auto.master:
/-      /etc/auto.direct
/share  /etc/auto.share
-----
/etc/auto.direct:
-----
/etc/auto.share:
man      ipa.example.com:/export/docs
```

On Solaris, add the indirect map using the **ldapclient** command to add the LDAP entry directly:

```
ldapclient -a
serviceSearchDescriptor=auto_share:automountMapName=auto.share,cn=location,cn=automount,dc=example,dc=com?one
```

17.5.3. Importing Automount Maps

If there are existing automount maps, these can be imported into the IdM automount configuration.

```
ipa automountlocation-import location map_file [--continuous]
```

The only required information is the IdM automount location and the full path and name of the map file.

The **--continuous** option tells the **automountlocation-import** command to continue through the map file, even if the command encounters errors.

For example:

```
$ ipa automountlocation-import raleigh /etc/custom.map
```

Chapter 18. Policy: Defining Password Policies

All users must have a password which they use to authenticate to the Kerberos domain. Identity Management defines and enforces rules about password complexity, password histories, and account lockouts in order to maintain security.



NOTE

IdM, by default, does not expose passwords to clients, even hashed passwords, for system security.

18.1. About Password Policies and Policy Attributes

A *password policy* sets certain standards for passwords, such as the password complexity and the rules for changing passwords. A password policy minimizes the inherent risk of using passwords by ensuring that they meet adequate complexity standards to thwart brute force attacks and they are changed frequently enough to mitigate the risk of someone revealing or discovering a password.

There are three main configuration areas that are defined within the password policy:

- Strength or complexity requirements
- History
- Account lockout

The IdM password policy is enforced jointly by the KDC and the LDAP server. While the password policy is set in the LDAP directory and is based on 389 Directory Server password policy attributes, the policy is ultimately constrained by the KDC password policy framework. The KDC policy is less flexible than the 389 Directory Server policy framework, so the IdM password policy can only implement password policy elements supported in the KDC. Any other policy settings made within the 389 Directory Server are not visible or enforced in Identity Management.

Password policies are assigned either globally or to groups in IdM, not to individual users. The password policy is assigned a weight, so that if a user belongs to multiple groups with different password policies, the policy with the highest priority will take precedence.

The different policy attributes that can be set are listed in [Table 18.1, “Password Policy Settings”](#).

Table 18.1. Password Policy Settings

Configuration Property	Command-Line Option	Description
Options for both the UI and CLI		
Minimum Password Lifetime	--minlife	Sets the minimum period of time, in hours, that a user's password must be in effect before the user can change it. This can prevent a user from changing a password and then immediately changing it to the original value. The default value is one hour.
Maximum Password Lifetime	--maxlife	Sets the maximum period of time, in days, that a user's password can be in effect before it must be changed. The default value is 90 days.
Minimum Number of Character Classes	--minclasses	Sets the minimum number of different classes, or types, of character that must exist in a password before it is considered valid. For example, setting this value to 3 requires that any password must have characters from at least three categories in order to be approved. The default value is zero (0), meaning there are no required classes. There are six character classes: » Upper-case characters » Lower-case characters » Digits » Special characters (for example, punctuation) » 8-bit characters (characters whose decimal code starts at 128 or below) » Number of repeated characters This weights in the opposite direction, so that if you have too many repeated characters you will not meet the quorum to satisfy the "level" expressed by krbPwdMinDiffChars.
Minimum Length of Password	--minlength	Sets the minimum number of characters for a password. The default value is eight characters.

Password History	--history	Sets the number of previous passwords that are stored and which a user is prevented from using. For example, if this is set to ten, IdM prevents a user from reusing any of their previous ten passwords. The default value is zero (0), which disables password history.
 NOTE Even with the password history set to zero, users cannot reuse a <i>current</i> password.		
Options for the CLI only		
Priority	--priority	Sets the priority which determines which policy is in effect. The lower the number, the higher priority. Although this priority is required when the policy is first created in the UI, it cannot be reset in the UI. It can only be reset using the CLI.
Maximum Consecutive Failures	--maxfail	Specifies the maximum number of consecutive failures to input the correct password before the user's account is locked.
Fail Interval	--failinterval	Specifies the period (in seconds) after which the failure count will be reset.
Lockout Time	--lockouttime	Specifies the period (in seconds) for which a lockout is enforced.

18.2. Viewing Password Policies

There can be multiple password policies configured in IdM. There is always a global policy, which is created with the server. Additional policies can be created for groups in IdM.

The UI lists all of the group password policies and the global policy on the **Password Policies** page.

Using the CLI, both global and group-level password policies can be viewed using the **pwpolicy-show** command. The CLI can also display the password policy in effect for a user.

18.2.1. Viewing the Global Password Policy

The global password policy is created with the server. This applies to every user until a group-level password policy supersedes it.

The default settings for the global password policy are listed in [Table 18.2, “Default Global Password Policy”](#).

Table 18.2. Default Global Password Policy

Attribute	Value
Max lifetime	90 (days)
Min lifetime	1 (hour)
History size	0 (unset)
Character classes	0 (unset)
Min length	8
Max failures	6
Failure reset interval	60
Lockout duration	600

18.2.1.1. With the Web UI

1. Click the **Policy** tab, and then click the **Password Policies** subtab.
2. All of the policies in the UI are listed by group. The global password policy is defined by the **global_policy** group. Click the group link.



3. The global policy is displayed.

The screenshot shows the IPA web interface with the 'Policy' tab selected. Under 'Password Policies', the 'global_policy' is selected. The page title is 'PASSWORD POLICY: global_policy'. There is a 'Settings' tab and buttons for 'Refresh', 'Reset', and 'Update'. A section titled 'PASSWORD POLICY' contains the following configuration fields:

Field	Value
Group	global_policy
Max lifetime (days)	90
Min lifetime (hours)	1
History size	0
Character classes	0
Min length	8
Max failures	3
Failure reset interval	60
Lockout duration	10
Priority	

18.2.1.2. With the Command Line

To view the global policy, simply run the **pwpolicy-show** command with no arguments:

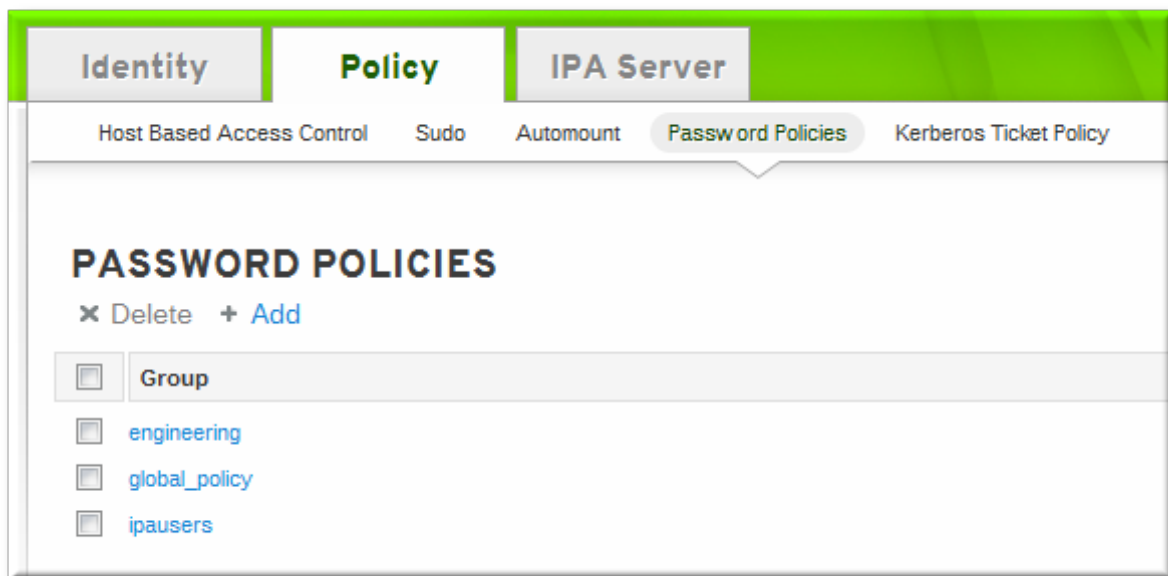
```
$ ipa pwpolicy-show

Group: global_policy
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 0
Character classes: 0
Min length: 8
Max failures: 6
Failure reset interval: 60
Lockout duration: 600
```

18.2.2. Viewing Group-Level Password Policies

18.2.2.1. With the Web UI

1. Click the **Policy** tab, and then click the **Password Policies** subtab.
2. All of the policies in the UI are listed by group. Click the name of the group which is assigned the policy.



3. The group policy is displayed.

Identity Policy IPA Server

Host Based Access Control Sudo Automount Password Policies Kerberos Ticket Policy

Password Policies » engineering

PASSWORD POLICY: ipausers

Settings

Refresh Reset Update

▼ PASSWORD POLICY

Group: engineering

Max lifetime (days): 90 [undo](#)

Min lifetime (hours): 48 [undo](#)

History size: 6 [undo](#)

Character classes: 3 [undo](#)

Min length: 8 [undo](#)

18.2.2.2. With the Command Line

For a group-level password policy, specify the group name with the command:

```
$ ipa pwpolicy-show examplegroup
```

```
Group: global_policy
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 3
Character classes: 4
Min length: 8
Max failures: 3
Failure reset interval: 15
Lockout duration: 150
```

18.2.3. Viewing the Password Policy in Effect for a User

A user may belong to multiple groups, each with their own separate password policies. These policies are not additive. Only one policy is in effect at a time and it applies to all password policy attributes. To see which policy is in effect for a specific user, the **pwpolicy-show** command can be run for a specific user. The results also show *which* group policy is in effect for that user.

```
$ ipa pwpolicy-show --user=jsmith
```

```
Group: admins
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 0
Character classes: 0
Min length: 8
Max failures: 6
Failure reset interval: 60
Lockout duration: 600
```

18.3. Creating and Editing Password Policies

A password policy can be selective; it may only define certain elements. A *global* password policy sets defaults that are used for every user entry, unless a group policy takes priority.



NOTE

A global policy always exists, so there is no reason to add a global password policy.

Group-level policies override the global policies and offer specific policies that only apply to group members. Password policies are not cumulative. Either a group policy or the global policy is in effect for a user or group, but not both simultaneously.

Group-level policies do not exist by default, so they must be created manually.

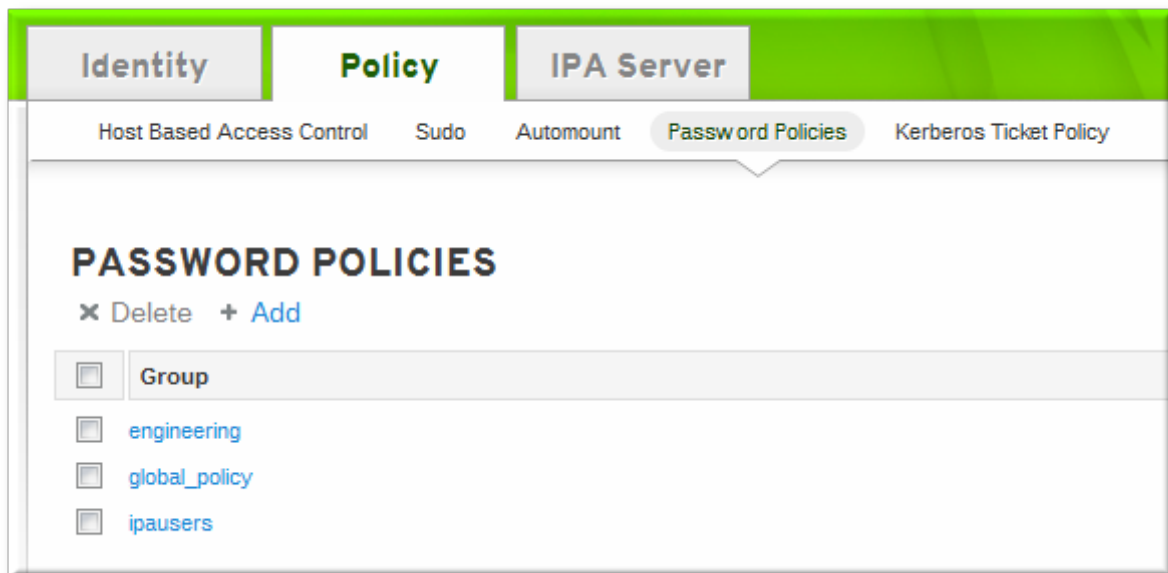


NOTE

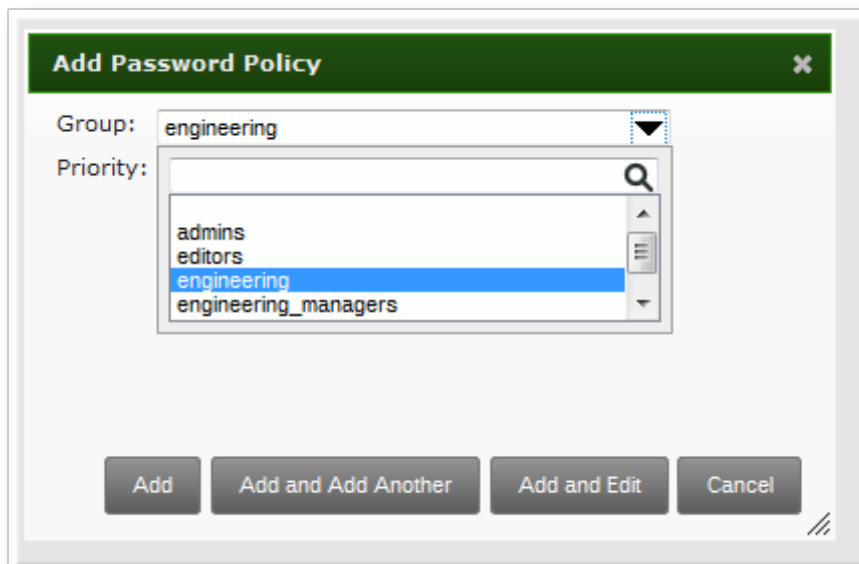
It is not possible to set a password policy for a non-existent group.

18.3.1. Creating Password Policies in the Web UI

1. Click the **Policy** tab, and then click the **Password Policies** subtab.
2. All of the policies in the UI are listed by group. The global password policy is defined by the **global_policy** group. Click the group link.



3. Click the **Add** link at the top.
4. In the pop-up box, select the group for which to create the password policy.



5. Set the priority of the policy. The higher the number, the lower the priority.
Only one password policy is in effect for a user, and that is the highest priority policy.

Add Password Policy [X]

Group:

Priority:

[Add] [Add and Add Another] [Add and Edit] [Cancel]

**NOTE**

The priority cannot be changed in the UI once the policy is created.

6. Click the **Add and Edit** button so that the policy form immediately opens.
7. Set the policy fields. Leaving a field blank means that attribute is not added the password policy configuration.
 - *Max lifetime* sets the maximum amount of time, in days, that a password is valid before a user must reset it.
 - *Min lifetime* sets the minimum amount of time, in hours, that a password must remain in effect before a user is permitted to change it. This prevents a user from attempting to change a password back immediately to an older password or from cycling through the password history.
 - *History size* sets how many previous passwords are stored. A user cannot re-use a password that is still in the password history.
 - *Character classes* sets the different categories of character that must be used in the password. For example, a character class can be a number, special character, or capital; the complete list of categories is in [Table 18.1, “Password Policy Settings”](#). This is part of setting the complexity requirements.
 - *Min length* sets how many characters must be in a password. This is part of setting the complexity requirements.

18.3.2. Creating Password Policies with the Command Line

Password policies are added with the **pwdpolicy-add** command.

```
$ ipa pwdpolicy-add groupName --attribute-value
```

For example:

```
$ ipa pwpolicy-add exampleGroup --minlife=7 --maxlife=49 --history= --priority=1
Group: exampleGroup
Max lifetime (days): 49
Min lifetime (hours): 7
Priority: 1
```

**TIP**

Setting an attribute to a blank value effectively removes that attribute from the password policy.

18.3.3. Editing Password Policies with the Command Line

As with most IdM entries, a password policy is edited by using a ***-mod** command, **pwpolicy-mod**, and then the policy name. However, there is one difference with editing password policies: there is a global policy which always exists. Editing a group-level password policy is slightly different than editing the global password policy.

Editing a group-level password policy follows the standard syntax of ***-mod** commands. It uses the **pwpolicy-mod** command, the name of the policy entry, and the attributes to change. For example:

```
[jsmith@ipaserver ~]$ ipa pwpolicy-mod exampleGroup --lockouttime=300 --history=5
--minlength=8
```

To edit the global password policy, use the **pwpolicy-mod** command with the attributes to change, *but without specifying a password policy name*. For example:

```
[jsmith@ipaserver ~]$ ipa pwpolicy-mod --lockouttime=300 --history=5 --
minlength=8
```

18.4. Managing Password Expiration Limits

Password policies are applied **at the time a password is changed**. So, when a password is set, it conforms to the password policy in effect at that time. If the password policy is changed later, that change is not applied, retroactively, to the password.

Setting password expiration periods is configured as part of the group password policy. Creating and editing password policies (including the expiration attribute in the policy) is covered in [Section 18.3, “Creating and Editing Password Policies”](#).

With password expiration periods, there are two attributes that are related:

- ▶ The maximum lifetime setting given in the password policy (**--maxlife**)
- ▶ The actual date that the password for a given user expires (**krbPasswordExpiration**)

Changing the password expiration time in the password policy does not affect the expiration date for a user, until the user password is changed. If the password expiration date needs to be changed immediately, it can be changed by editing the user entry.

To force the expiration date to change, reset the **krbPasswordExpiration** attribute value for the user. **This can only be done using ldapmodify**. For example, for a single user:


```
[bjensen@ipaserver ~]$ ldapmodify -Y GSSAPI -h ipaserver.example.com -p 389 -vv
dn: uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
changetype: modify
replace: krbpasswordexpiration
krbpasswordexpiration: 20140202203734Z
-
```

Multiple entries can be edited simultaneously by referencing an LDIF file in the **-f** option with the **ldapmodify** command.



TIP

If an administrator resets a password, it expires the previous password and forces the user to update the password. When the user updates the password, it automatically uses the new password policies, including a new expiration date.

18.5. Changing the Priority of Group Password Policies

A user may belong to multiple groups, each with different password policies. Since only one policy can be in effect for a user, there has to be a method to assign precedence to policies. That is done through *priority*.

The highest priority is zero (0). The lower the number, the higher the priority.

This is set initially when the password policy is created. It can be modified after the policy is created by resetting the **--priority** option.

```
$ ipa pwpolicy-mod examplegroup --priority=10
```

When a user belongs to multiple groups, the group password policy with the lowest number has the priority.



NOTE

The password policy priority cannot be changed in the UI after it is created.

18.6. Setting Account Lockout Policies

A brute force attack occurs when a malefactor attempts to guess a password by simply slamming the server with multiple login attempts. An account lockout policy prevents brute force attacks by blocking an account from logging into the system after a certain number of login failures — even if the correct password is subsequently entered.



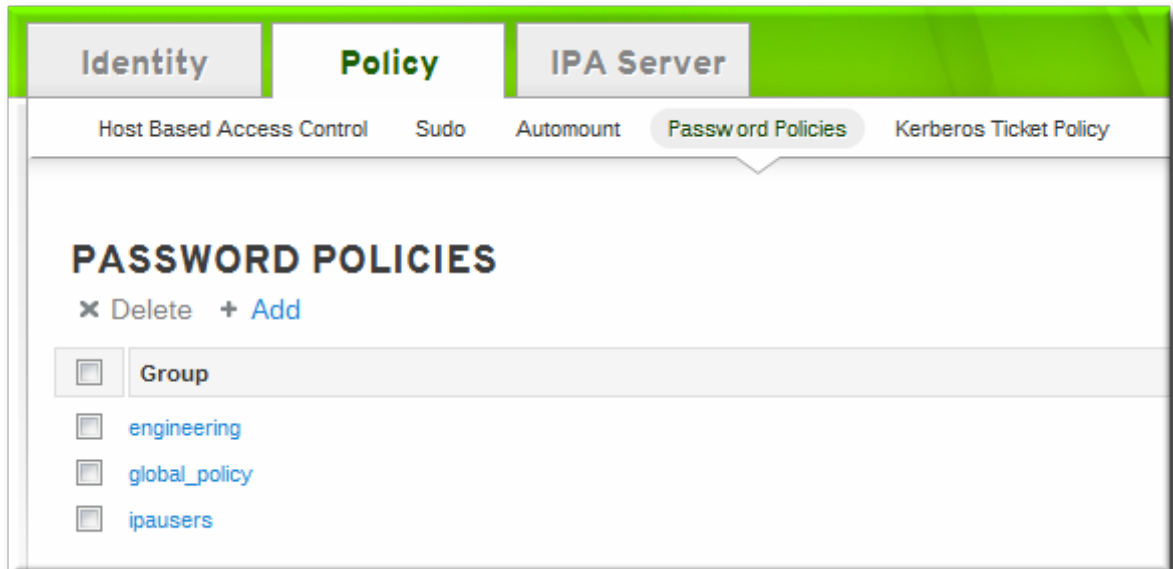
NOTE

A user account can be manually unlocked by an administrator using the **ipa user-unlock**. Refer to [Section 9.6, “Unlocking User Accounts After Password Failures”](#).

18.6.1. In the UI

These attributes are available in the password policy form when a group-level password policy is created or when any password policy (including the global password policy) is edited.

1. Click the **Policy** tab, and then click the **Password Policies** subtab.
2. Click the name of the policy to edit.



3. Set the account lockout attribute values.

There are three parts to the account lockout policy:

- » The number of failed login attempts before the account is locked (**Max Failures**).
- » The time after a failed login attempt before the counter resets (**Failure reset interval**). Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally

lapses after a certain amount of time. This is in seconds.

- How long an account is locked after the max number of failures is reached (**Lockout duration**). This is in seconds.

18.6.2. In the CLI

There are three parts to the account lockout policy:

- The number of failed login attempts before the account is locked (**--maxfail**).
- How long an account is locked after the max number of failures is reached (**--lockouttime**). This is in seconds.
- The time after a failed login attempt before the counter resets (**--failinterval**). Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally lapses after a certain amount of time. This is in seconds.

These account lockout attributes can all be set when a password policy is created with **pwpolicy-add** or added later using **pwpolicy-mod**. For example:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa pwpolicy-mod examplegroup --maxfail=4 --
lockouttime=600 --failinterval=30
```

18.7. Enabling a Password Change Dialog

There may be situations when a user exists in Identity Management but does not have a valid Kerberos ticket, meaning he cannot authenticate to the IdM domain. This is possible for new users or for users whose domain passwords have expired. Much like enabling password authentication in the web UI, it is possible to enable password-based authentication to the client. This opens up a password change dialog box to allow the user to reset the expired password.

The password change dialog is enabled by using OpenSSH's *challenge-response* authentication.

The challenge-response dialog is optional. In many environments, it is not necessary because SSSD can handle changing expired passwords by invoking the required PAM modules. However, using the challenge-response option in OpenSSH makes it possible to do password changes directly in PAM and to support full PAM conversations.

This is not enabled by default, but it can be enabled by editing the OpenSSH configuration.

1. Open the **/etc/ssh/sshd_config** file.
2. Set **ChallengeResponseAuthentication** to **yes**.

Chapter 19. Policy: Managing the Kerberos Domain

Kerberos authentication is the core of authentication within the IdM domain. The IdM server actually runs a Kerberos server within it, and this Kerberos server can be configured for custom policies for managing tickets and keytabs.

For more information on Kerberos concepts, see the MIT Kerberos documentation, <http://web.mit.edu/kerberos/www/>.



IMPORTANT

Identity Management has its own command-line tools to use to manage Kerberos policies. **Do not** use **kadmin** or **kadmin.local** to manage IdM Kerberos settings.

19.1. About Kerberos

Kerberos provides an authentication layer between services and users. Kerberos centralizes authentication into a single location; a user authenticates to the Kerberos server, and then when that user attempts to access any resource on the network, that resource can check the *key distribution center* (KDC) for the stored user credentials. This allows users to access multiple resources without having to supply credentials separately to each and every one.

All of the users and services, combined, and all of the KDCs and Kerberos servers that are aware of each other constitute a *realm*. Each user, machine, and service within the realm is identified by a unique name called the *principal*. The user or service uses the principal and a verifying credential (usually a password) to authenticate to the KDC. The credential that is shared with the KDC is a *key* and it is stored in a file called a *key table* or *keytab*.

When the KDC verifies the user's identity, it issues a *ticket*. The ticket is a long-term pass to any service and machine on the realm. The KDC issues the user a special kind of ticket called a *ticket-granting ticket* (TGT). Whenever the user tries to access a resource within the Kerberos realm, the resource sends a request for a ticket specifically for it. The TGT is used to issue a resource-specific ticket that the resource then uses to authenticate the user and grant access.



NOTE

When an IdM client is first configured, the host principal is automatically retrieved by the setup script and stored in the **/etc/krb5.keytab** file. This host principal is stored within the host record so that local service commands cannot be used with this principal. This prepares the client to function in the IdM realm.

19.1.1. About Principal Names

The principal identifies not only the user or service, but also the realm that that entity belongs to. A principal name has two parts, the identifier and the realm:

***identifier*@REALM**

For a user, the *identifier* is only the Kerberos username. For a service, the *identifier* is a combination of the service name and the hostname of the machine it runs on:

```
service/FQDN@REALM
```

The *service* name is a case-sensitive string that is specific to the service type, like **host**, **ldap**, **http**, and **dns**. Not all services have obvious principal identifiers; the **sshd** daemon, for example, uses the host service principal.

The host principal is usually stored in **/etc/krb5.keytab**.

When Kerberos requests a ticket, it always resolves the domain name aliases (DNS CNAME records) to the corresponding DNS address (A or AAAA records). The hostname from the address record is then used when service or host principals are created.

For example:

```
www.example.com CNAME web-01.example.com
web-01.example.com A 192.0.2.145
```

A service attempts to connect to the host using its CNAME alias:

```
$ ssh www.example.com
```

The Kerberos server requests a ticket for the resolved hostname, **web-01.example.com@EXAMPLE.COM**, so the host principal must be **host/web-01.example.com@EXAMPLE.COM**.

19.1.2. About Protecting Keytabs

To protect keytab files, reset the permissions and ownership to restrict access to the files to only the keytab owner. For example, set the owner of the Apache keytab (**/etc/httpd/conf/ipa.keytab**) to **apache** and the mode to **0600**.

19.2. Setting Kerberos Ticket Policies

The Kerberos *ticket policy* sets basic restrictions on managing tickets within the Kerberos realm, such as the maximum ticket lifetime and the maximum renewal age (the period during which the ticket is renewable).

The Kerberos ticket policy is set globally so that it applies to every ticket issued within the realm. IdM also has the ability to set user-level ticket policies which override the global policies. This can be used, for example, to set extended expiration times for administrators or to set shorter expiration times for some employees.

19.2.1. Setting Global Ticket Policies

19.2.1.1. From the Web UI

1. Click the **Policy** tab, and then click the **Kerberos Ticket Policy** subtab.
2. Change the ticket lifetime policies.

- *Max renew* sets the period after a ticket expires that it can be renewed.
- *Max life* sets the active period (lifetime) of a Kerberos ticket.

3. Click the **Update** link at the top of the policy page.

4. Restart the KDC.

```
# service krb5kdc restart
```



IMPORTANT

Any change to the global Kerberos ticket policy requires a restart of the KDC for the changes to take effect.

19.2.1.2. From the Command Line

The **ipa krbtpolicy-mod** command modifies the policy, while the **ipa krbtpolicy-reset** command resets the policy to the default values.

For example:

```
# ipa krbtpolicy-mod --maxlife=3600 --maxrenew=18000
Max life: 3600
Max renew: 18000
```



IMPORTANT

Any change to the global Kerberos ticket policy requires a restart of the KDC for the changes to take effect. Restart the KDC:

```
# service krb5kdc restart
```

19.2.2. Setting User-Level Ticket Policies

User-level Kerberos ticket policies are set using the same commands as global policies, but the user is specified in the command.

For example:

```
# ipa krbtpolicy-mod jsmith --maxlife=3600
Max life: 3600
```



IMPORTANT

User-level policies take effect immediately on the next requested ticket (such as running **kinit**), without having to restart the KDC service.

19.3. Refreshing Kerberos Tickets

Kerberos keys are analogous to passwords. Like passwords, security policies may require that Kerberos tickets are manually refreshed after a specified interval.

The version of the key is shown in its *key version number* (KVNO). Refreshing (also called rotating) the principal's key increments the KVNO in the keytab entry. When a key is refreshed, a new entry is added to the keytab with a higher KVNO. The original key remains in the keytab but is no longer used to issue tickets.

Each keytab for the IdM realm has an entry in the IdM LDAP server, which includes its last change time. The principals which need to be refreshed can be regenerated using the **ipa-getkeytab** command.



NOTE

The **ipa-getkeytab** command does not delete the old keytab in case it already exists in the file.

1. Find all keytabs issued before the requisite date. For example, this looks for any principals created between midnight on January 1, 2010, and 11:59 PM on December 31, 2010:

```
# ldapsearch -x -b "cn=computers,cn=accounts,dc=example,dc=com"
"(&(krblastpwdchange>=20100101000000)(krblastpwdchange<=20101231235959))" dn
krbprincipalname

# ldapsearch -x -b "cn=services,cn=accounts,dc=example,dc=com"
"(&(krblastpwdchange>=20100101000000)(krblastpwdchange<=20101231235959))" dn
krbprincipalname
```

- » Host (machine) principals are stored under the **cn=computers,cn=accounts,dc=example,dc=com** subtree.
- » Service principals are stored under the **cn=services,cn=accounts,dc=example,dc=com** subtree.
- » Filter by the last change date (**krblastpwdchange**).
- » Limit the search result information to only the entry name and principal by specifying the **dn**

krbprincipalname attributes.

Dates are expressed in YYYYMMDD format, and times in HHMMSS format (GMT).

2. Retrieve a new keytab for the principal using the **ipa-getkeytab** command. This requires the location of the original keytab for the service or host (**-k**), the principal (**-p**), and the IdM server hostname (**-s**).

For example, this refreshes the host principal with a keytab in the default location of **/etc/krb5.keytab**:

```
# ipa-getkeytab -p host/client.example.com@EXAMPLE.COM -s ipa.example.com -k /etc/krb5.keytab
```

This refreshes the keytab for the Apache service, with a keytab in the default location of **/etc/httpd/conf/ipa.keytab**:

```
# ipa-getkeytab -p HTTP/client.example.com@EXAMPLE.COM -s ipa.example.com -k /etc/httpd/conf/ipa.keytab
```

3. Regenerate the keytab using **ipa-getkeytab** for every service.

The **klist** command displays the new key version number for the refreshed keytab. The original keytab still exists in the database, and it is listed with the previous KVNO.

```
# klist -kt /etc/krb5.keytab
Keytab: WRFILE:/etc/krb5.keytab
KVNO Timestamp Principal
-----
1 06/09/10 11:23:01 host/client.example.com@EXAMPLE.COM(aes256-cts-hmac-sha1-96)
2 06/09/11 05:58:47 host/client.example.com@EXAMPLE.COM(aes256-cts-hmac-sha1-96)
1 03/09/11 13:57:16 krbtgt/EXAMPLE.COM@EXAMPLE.COM(aes256-cts-hmac-sha1-96)
1 03/09/11 13:57:16 HTTP/ipa.example.com@EXAMPLE.COM(aes256-cts-hmac-sha1-96)
1 03/09/11 13:57:16 ldap/ipa.example.com@EXAMPLE.COM(aes256-cts-hmac-sha1-96)
```

Tickets issued against the old keytab continue to work, while new tickets are issued using the key with the highest KVNO. This avoids any disruption to system operations.



IMPORTANT

Some services, such as NFSv4, only support a limited set of encryption types. Pass the appropriate arguments to the **ipa-getkeytab** command to configure the keytab properly.

19.4. Caching Kerberos Passwords

A machine may not always be on the same network as the IdM domain; for example, a machine may need to be logged into a VPN before it can access the IdM domain. If a user logs into a system when it is offline and then later attempts to connect to IdM services, then the user is blocked because there is no IdM Kerberos ticket for that user. IdM works around that limitation by using SSSD to store the Kerberos passwords in the SSSD cache.

This is configured by default by the **ipa-client-install** script. A configuration parameter is added to the **/etc/sss/sss.conf** file which specifically instructs SSSD to store those Kerberos

passwords for the IdM domain:

```
[domain/example.com]
cache_credentials = True
ipa_domain = example.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
chpass_provider = ipa
ipa_server = _srv_, server.example.com
krb5_store_password_if_offline = true
```

This default behavior can be disabled during the client installation by using the **--no-krb5-offline-passwords** option.

This behavior can also be disabled by editing the `/etc/sss/sss.conf` file and removing the **krb5_store_password_if_offline** line or changing its value to false.

```
[domain/example.com]
...
krb5_store_password_if_offline = false
```

The SSSD configuration options for Kerberos authentication is covered in [the "Configuring Domains" section of the SSSD chapter in the Red Hat Enterprise Linux Deployment Guide](#).

19.5. Removing Keytabs

Refreshing Kerberos tickets adds a new key to the keytab, but it does not clear the keytab. If a host is being unenrolled and re-added to the IdM domain or if there are Kerberos connection errors, then it may be necessary to remove the keytab and create a new keytab.

This is done using the **ipa-rmkeytab** command. To remove all principals on the host, specify the realm with the **-r** option:

```
# ipa-rmkeytab -r EXAMPLE.COM -k /etc/krb5.keytab
```

To remove the keytab for a specific service, use the **-p** option to specify the service principal:

```
# ipa-rmkeytab -p ldap/client.example.com -k /etc/krb5.keytab
```

Chapter 20. Policy: Using sudo

Identity Management provides a mechanism for predictably and consistently apply **sudo** policies across the IdM domain. The **sudo** policies apply to domain users and domain hosts.

20.1. About sudo and IPA

The **sudo** command allows a system administrator to delegate authority to specific users to run specific commands as root or another specified user. **sudo** provides an audit trail of the commands and their arguments, so access can be tracked.

20.1.1. General sudo Configuration in Identity Management

sudo uses a local configuration file, `/etc/sudoers`, which defines the commands and users with sudo access. While this file can be shared among machines, there's no native way to distribute **sudo** configuration files among machines.

Identity Management uses its centralized LDAP database to contain the **sudo** configuration, which makes it globally available to all domain hosts. Identity Management also has a specialized LDAP schema for **sudo** entries that allows a lot more flexible and simpler configuration. This schema adds two key features:

- ▶ The Identity Management schema supports host groups in addition to netgroups for **sudo**, while **sudo** only supports netgroups.
For every host group, Identity Management also creates a corresponding shadow netgroup. This allows IdM administrators to create **sudo** rules that reference host groups, while the local **sudo** command uses the corresponding netgroup.
- ▶ Identity Management introduces the concept of a *sudo command group*. The group contains multiple commands, and then the command group can be referenced in the **sudo** configuration.

Because **sudo** does not support host groups and command groups, Identity Management translates the IdM **sudo** configuration into native **sudo** configuration when the **sudo** rules are created.

Because the **sudo** information is not available anonymously over LDAP by default, Identity Management defines a default **sudo** user, `uid=sudo,cn=sysaccounts,cn=etc,$SUFFIX`, which can be set in the LDAP/**sudo** configuration file, `/etc/sud-ldap.conf`.

Both **sudo** and Identity Management support user groups as part of the **sudo** configuration. User groups can be either Unix or non-POSIX groups. Creating non-POSIX groups can create some access issues because any users in the group inherit non-POSIX rights from the group. Having the choice between Unix and non-POSIX groups allows administrators the choice in group formatting and to avoid problems with inherited permissions or GID information.

20.1.2. sudo and Netgroups

As [Section 20.1.1, “General sudo Configuration in Identity Management”](#) mentions, the LDAP schema used for sudo entries in Identity Management supports host group-style groups in addition to netgroups. Really, Identity Management creates two groups, a visible host group and a shadow netgroup. **sudo** itself only supports NIS-style netgroups for group formats.

One important thing to consider is that even though **sudo** uses NIS netgroups, *it is not necessary to have a NIS server installed or a NIS client configured*. When any group is created for **sudo**, the NIS object is created in the Directory Server instance, and then the information is retrieved by NSS_LDAP or by SSSD. The client (in this case, **sudo**) then extracts the required NIS information from the information

provided by Identity Management's Directory Server.

In short, **sudo** configuration requires NIS-formatted netgroups. It does not require NIS.

The Identity Management Directory Server instance uses the standard LDAP schema for NIS objects, defined in RFC 2307.

20.1.3. Supported sudo Clients

Any system which is supported as an IdM client system can be configured as a **sudo** client in IdM.

20.2. Setting up sudo Commands and Command Groups

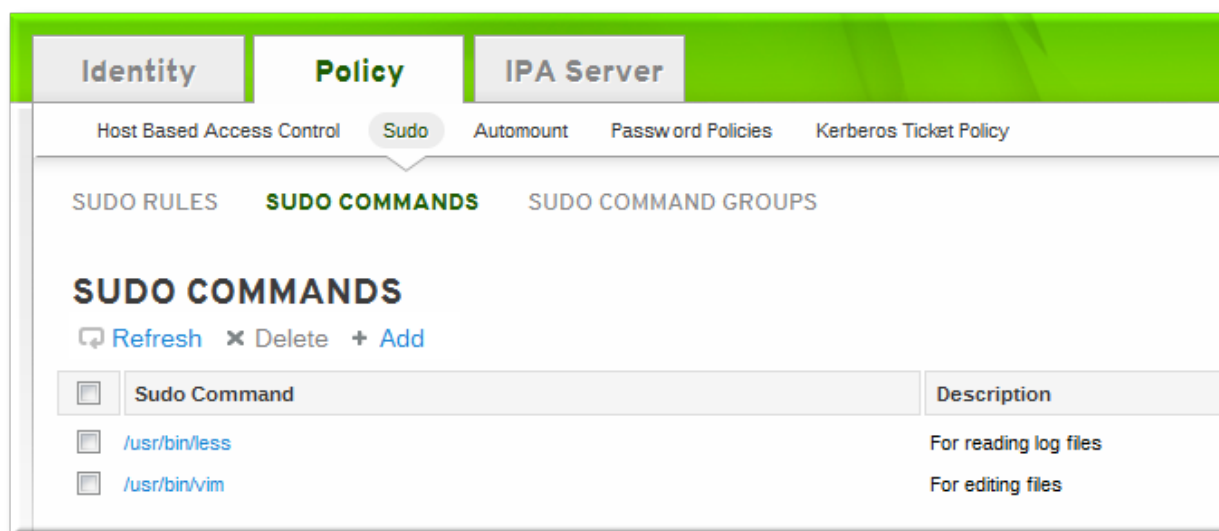
Just as in regular **sudo** configuration, any command which will be governed by **sudo** access must be listed in the configuration. Identity Management adds an extra control measure with *sudo command groups*, which allow a group of commands to be defined and then applied to the **sudo** configuration as one.

Adding a command or a command group makes it available to IdM to be defined in a **sudo** rule; simply adding a command does not automatically include it in a **sudo** rule.

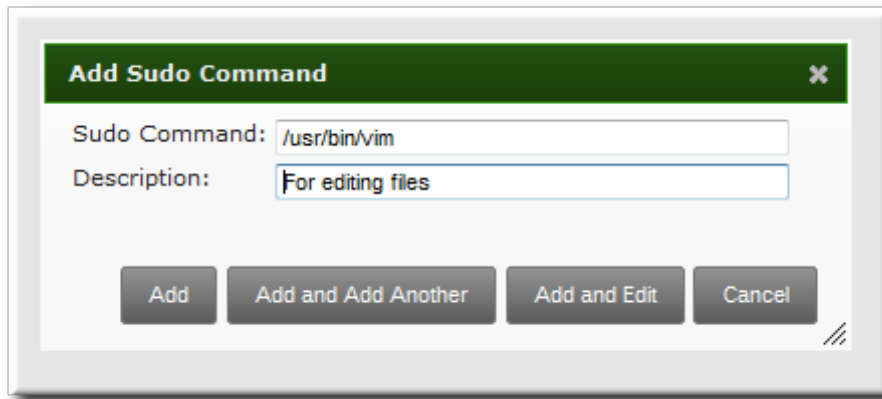
20.2.1. Adding sudo Commands

20.2.1.1. Adding sudo Commands with the Web UI

1. Click the **Policy** tab.
2. Click the **Sudo** subtab, and then select the **Sudo Commands** link.
3. Click the **Add** link at the top of the list of commands.



4. Enter the full system path and name of the command and, optionally, a description.



5. Click the **Add and Edit** button to go immediately to the edit pages for the command.
6. In the **Sudo Command Groups** tab, click the **Add** button to add the sudo command to a command group.
7. Click the checkbox by the groups for the command to join, and click the right arrows button, **>>**, to move the group to the selection box.
8. Click the **Add** button.

20.2.1.2. Adding sudo Commands with the Command Line

To add a single command, use the **sudocmd-add** command. This requires the full, local path to the command executable and a description of the command:

```
$ ipa sudocmd-add --desc "description" /local/path/to/command
```

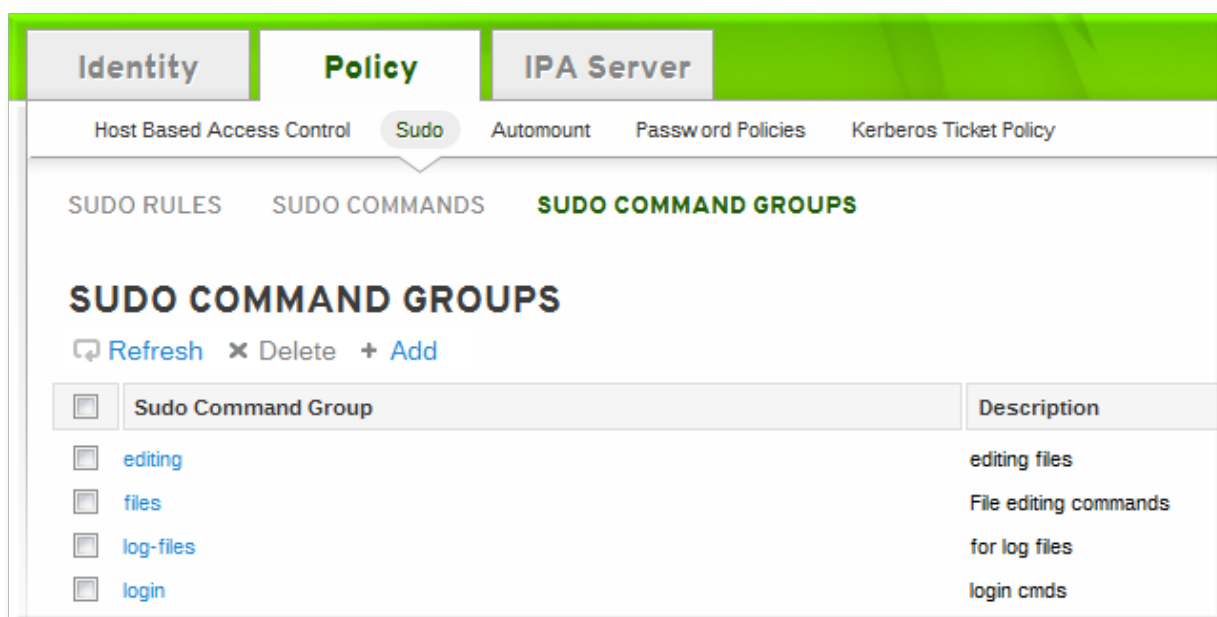
For example:

```
$ ipa sudocmd-add --desc 'For reading log files' '/usr/bin/less'
-----
Added sudo command "/usr/bin/less"
-----
sudo Command: /usr/bin/less
Description: For reading log files
```

20.2.2. Adding sudo Command Groups

20.2.2.1. Adding sudo Command Groups with the Web UI

1. Click the **Policy** tab.
2. Click the **Sudo** subtab, and then select the **Sudo Command Groups** link.
3. Click the **Add** link at the top of the list of command groups.



4. Enter the name and description for the new command group.

Add Sudo Command Group 

Sudo Command Group:

Description:

Add

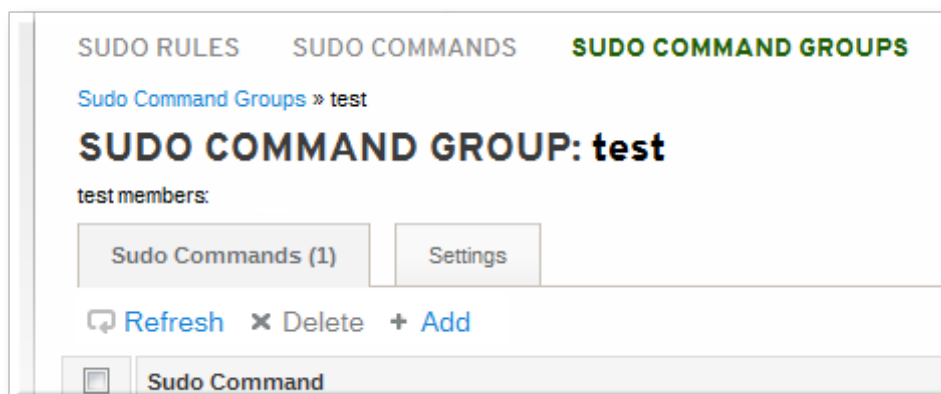
Add and Add Another

Add and Edit

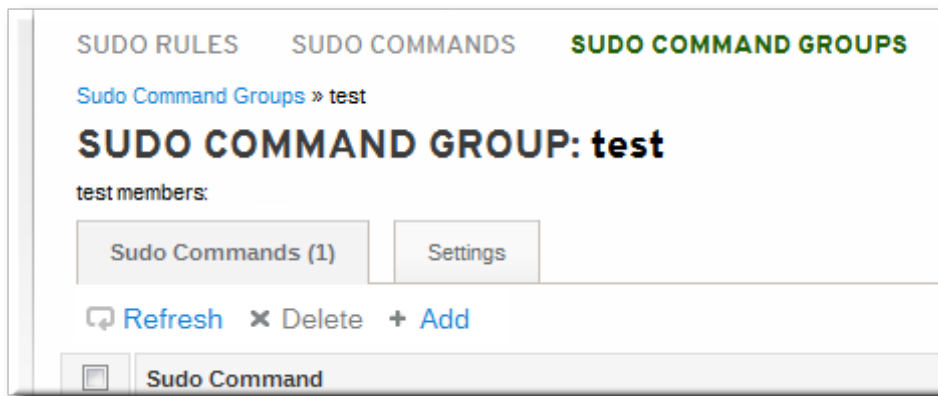
Cancel



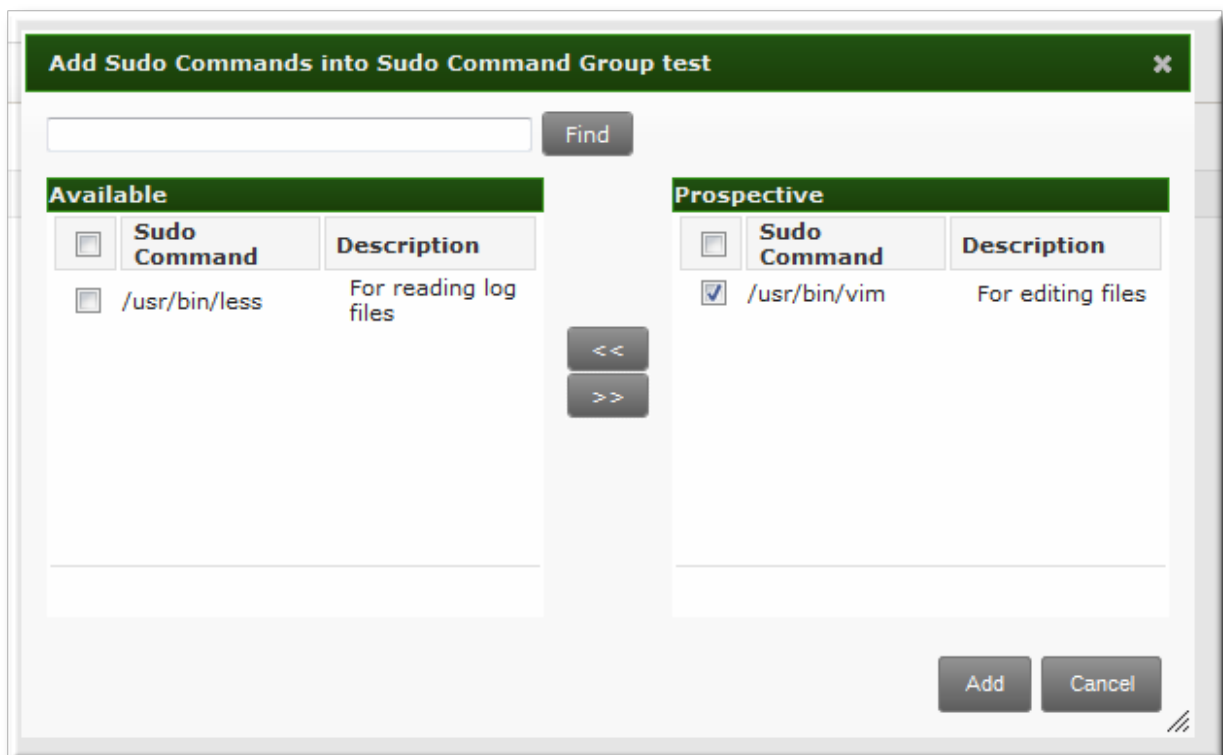
- Click the **Add and Edit** button to go immediately to the edit pages for the group.
- In the **Sudo Commands** tab, click the **Add** button to add a sudo command to the group.



7. In the **Sudo Commands** tab, click the **Add** button to add a sudo command to the group.



8. Click the checkbox by the names of the commands to add, and click the right arrows button, >>, to move the command to the selection box.



9. Click the **Add** button.

20.2.2.2. Adding sudo Command Groups with the Command Line

Creating a command group requires creating two entries, one for the group and one for the command itself:

1. Create the command group using the **sudocmdgroup-add** command:

```
$ ipa sudocmdgroup-add --desc 'File editing commands' files
-----
Added sudo command group "files"
-----
sudo Command Group: files
Description: File editing commands
```

2. Create a command entry using the **sudocmd-add** command:

```
$ ipa sudocmd-add --desc 'For editing files' '/usr/bin/vim'
-----
Added sudo command "/usr/bin/vim"
-----
sudo Command: /usr/bin/vim
Description: For editing files
```

3. Add the command, using its full directory location as its name, to the command group using the **sudocmdgroup-add-member** command:

```
$ ipa sudocmdgroup-add-member --sudocmds '/usr/bin/vim' files
sudo Command Group: files
Description: File editing commands
Member sudo commands: /usr/bin/vim
-----
Number of members added 1
-----
```

20.3. Defining sudo Rules

sudo rules are in a sense similar to access control rules: they define users who are granted access, the commands which are within the scope of the rule, and then the target hosts to which the rule applies. In IdM, additional information can be configured in the rule, such as **sudoers** options and run-as settings, but the basic elements always define who, what (services), and where (hosts).

20.3.1. About External Users

sudo rules define four elements: *who* can do *what*, *where*, and *as whom*. The *who* is the regular user, and the *as whom* is the system or other user identity which the user uses to perform tasks. Those tasks are system commands that can be run (or specifically not run) on a target machine.

Three of those elements — who, as whom, and where — are identities. They are users. Most of the time, those identities are going to be entities within the IdM domain because there will be overlap between the system users in the environment and the users and hosts belonging to the IdM domain.

However, that is not necessarily the case with all identities that a **sudo** policy may realistically cover. For example, **sudo** rules could be used to grant root access to member of the IT group in IdM, and that root user is not a user in IdM. Or, for another example, administrators may want to block access to certain hosts that are on a network but are not part of the IdM domain.

The **sudo** rules in Identity Management support the concept of *external* users — meaning, users which are stored and exist outside of the IdM configuration.

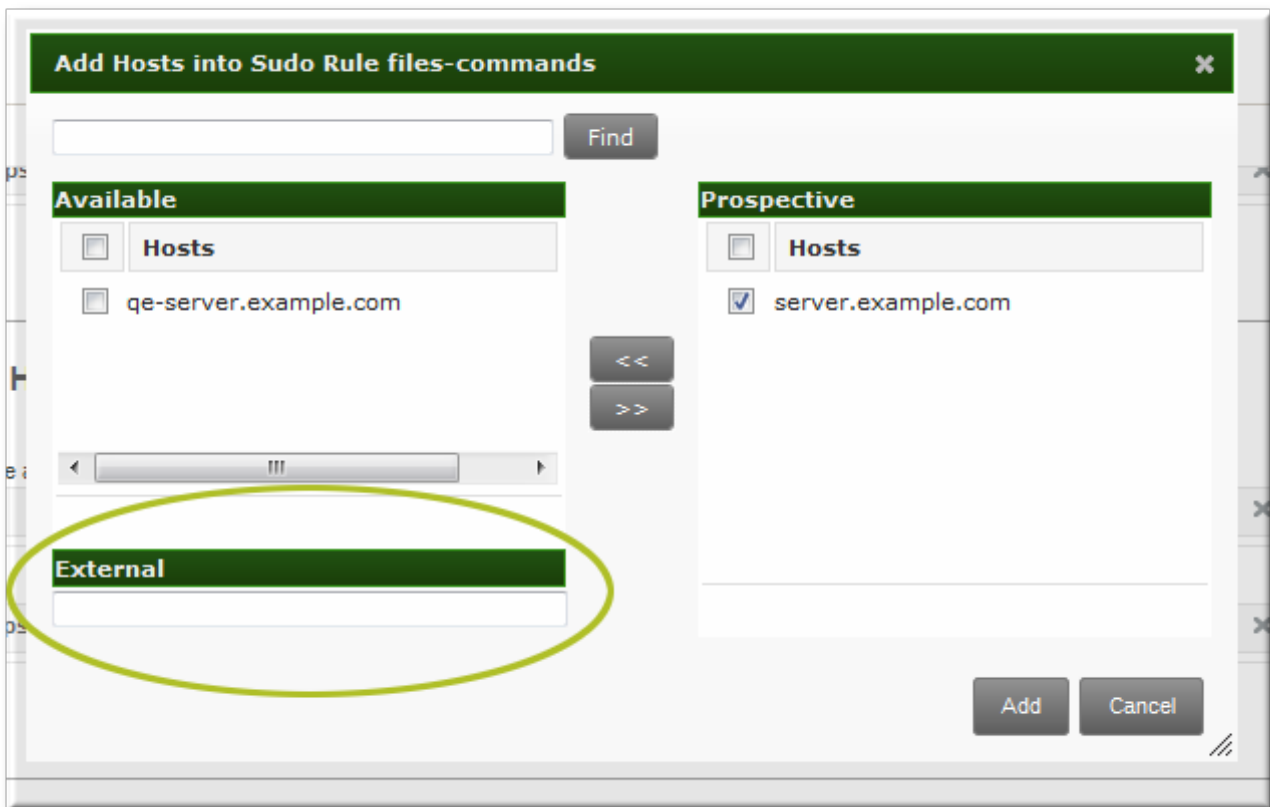


Figure 20.1. External Entities

When configuring a **sudo** rule, the user and run-as settings can point to an external identity to be included and evaluated in the **sudo** rule.

20.3.2. About sudo Options Format

The **sudo** rule can be configured to use any supported **sudoers** options. (The complete list of options is in the **sudoers** manpage and at http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options.)

However, the **sudo** rule configuration in Identity Management *does not* allow the same formatting as the configuration in the **/etc/sudoers** file. Specifically, Identity Management does not allow whitespaces in the options parameter, whether it is set in the UI or the CLI.

For example, in the **/etc/sudoers** file, it is permissible to list options in a comma-separated list with spaces between:

```
mail_badpass, mail_no_host, mail_no_perms, syslog = local2
```

However, in Identity Management, that same configuration would be interpreted as different arguments — including the equals sign (=) since it has spaces around it.

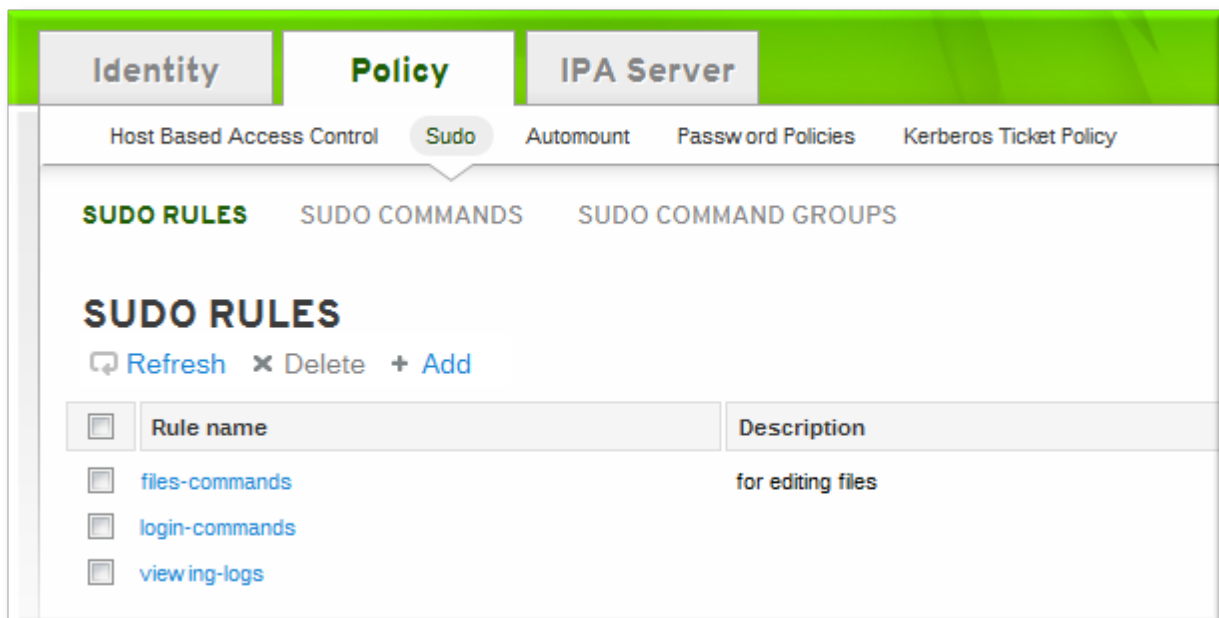
Likewise, linebreaks that are ignored in the **/etc/sudoers** file are not allowed in the Identity Management configuration:


```
env_keep = "COLORS DISPLAY EDITOR HOSTNAME HISTSIZE INPUTRC
            KDEDIR LESSSECURE LS_COLORS MAIL PATH PS1 PS2
            QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE LC_COLLATE
            LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES
            LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE
            LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
            XAUTHORITY"
```

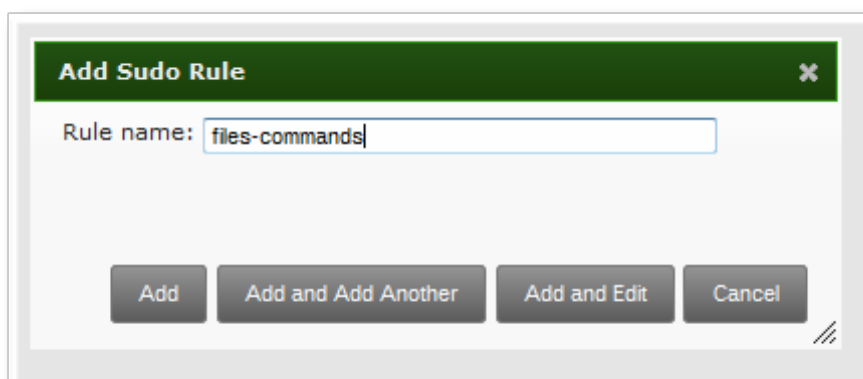
To use multiple **sudoers** options in Identity Management, configure each one as a separate option setting, rather than all on one line, as is allowed in the **/etc/sudoers** file.

20.3.3. Defining sudo Rules in the Web UI

1. Click the **Policy** tab.
2. Click the **Sudo** subtab, and then select the **Sudo Rules** link.
3. Click the **Add** link at the top of the list of sudo rules.



4. Enter the name for the rule.



5. Click the **Add and Edit** button to go immediately to set the configuration for the rule.
There are a number of configuration areas for the rule. The most basic elements are set in the **Who**, **Access This Host**, and **Run Commands** areas; the others are optional and are used to refine the rule.
6. *Optional.* In the **Options** area, add any **sudoers** options. The complete list of options is in the

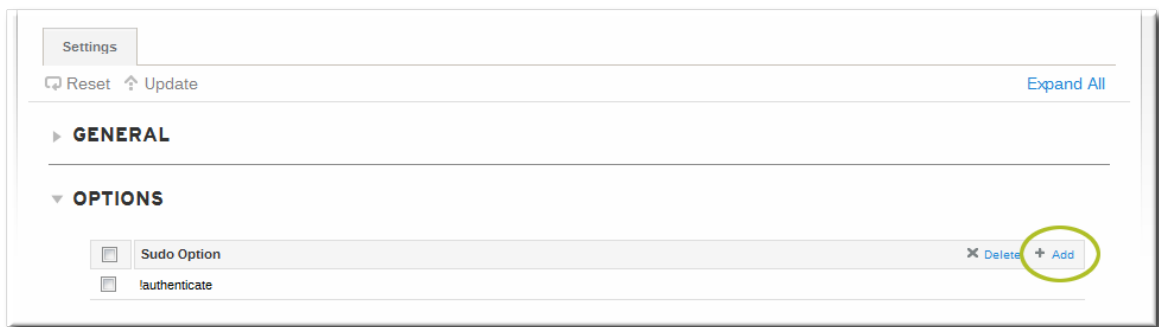
sudoers manpage and at http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options.



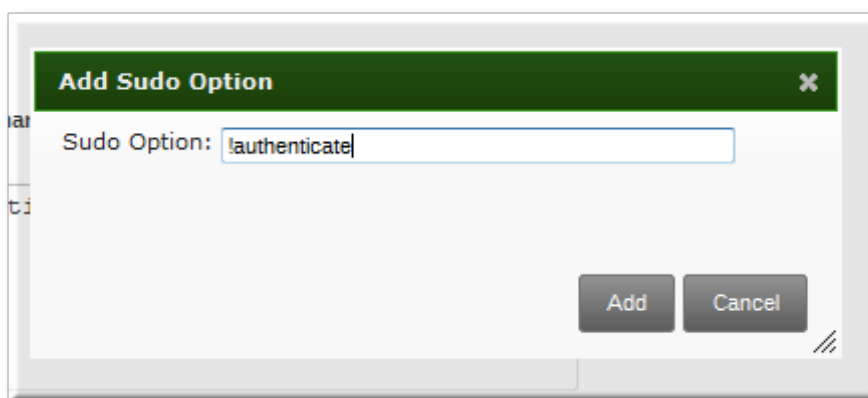
NOTE

As described in [Section 20.3.2, “About sudo Options Format”](#), do not use options with whitespace in the values. Rather than adding a list of options in one line, add a single option setting for each desired option.

- a. Click the **+ Add** link at the right of the options list.



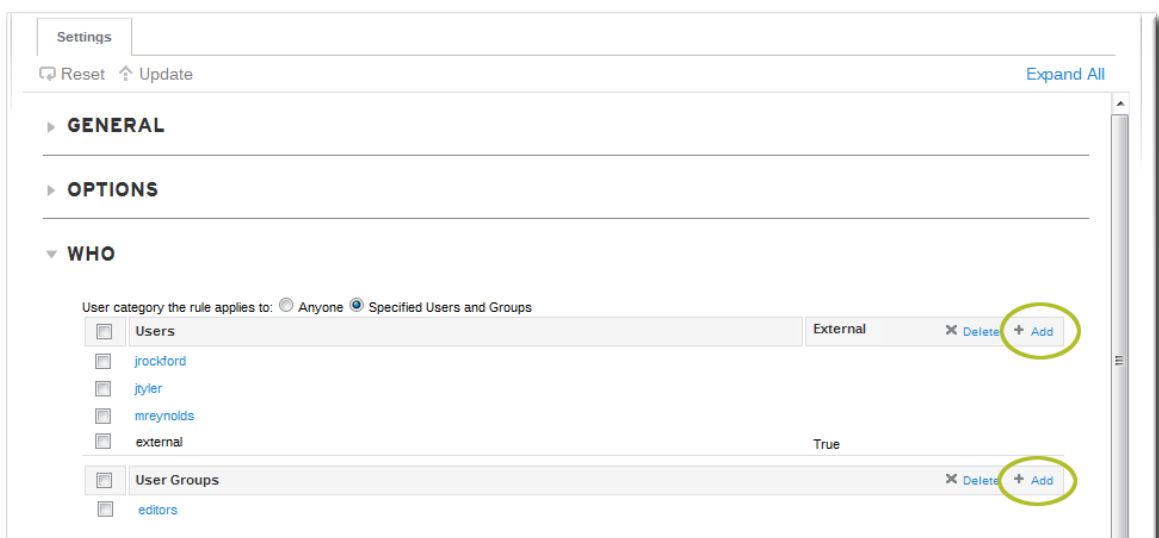
- b. Enter the **sudoers** option.



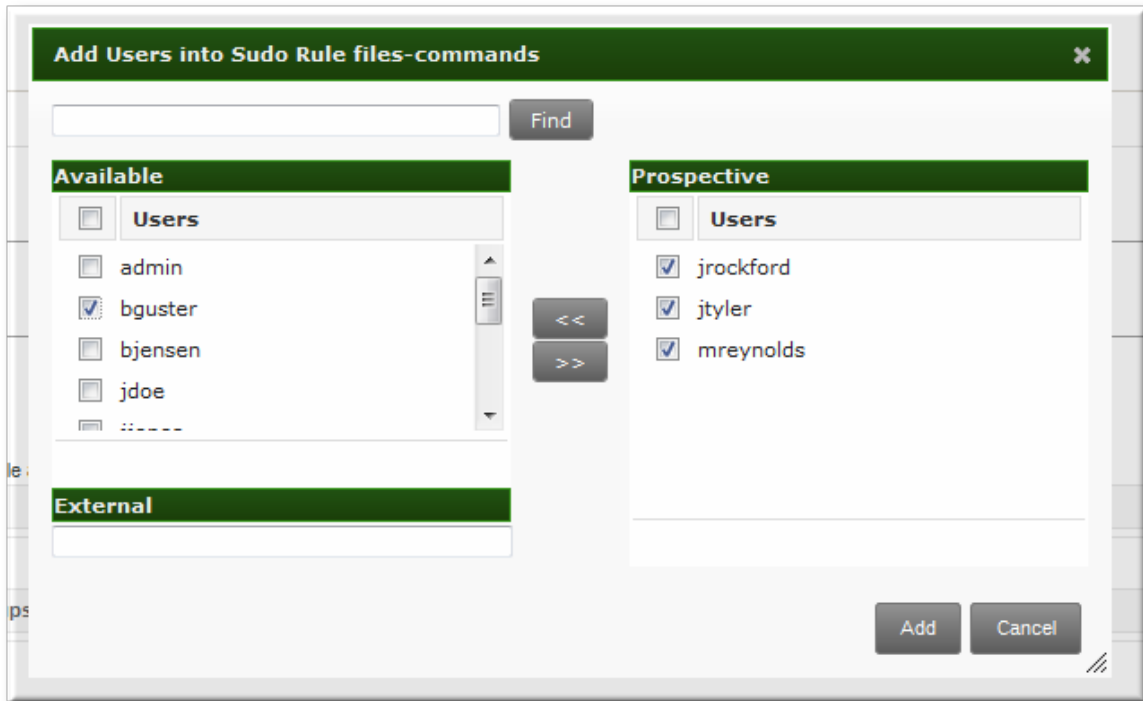
- c. Click **Add**.

7. In the **who** area, select the users or user groups to which the sudo rule is applied.

- a. Click the **+ Add** link at the right of the users list.



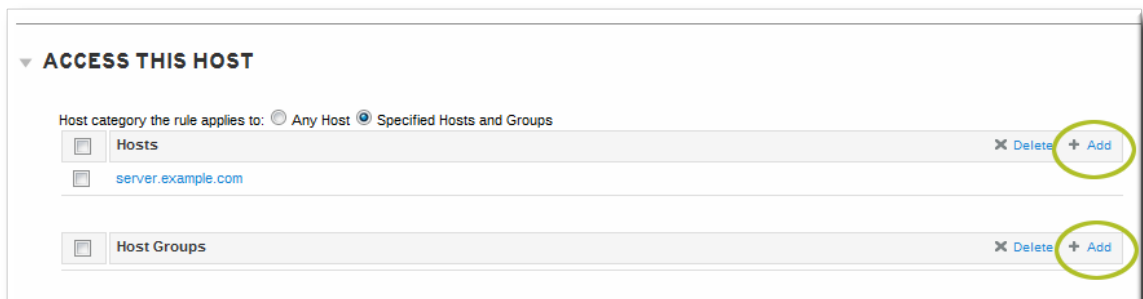
- b. Click the checkbox by the users to add to the rule, and click the right arrows button, **>>**, to move the users to the selection box.



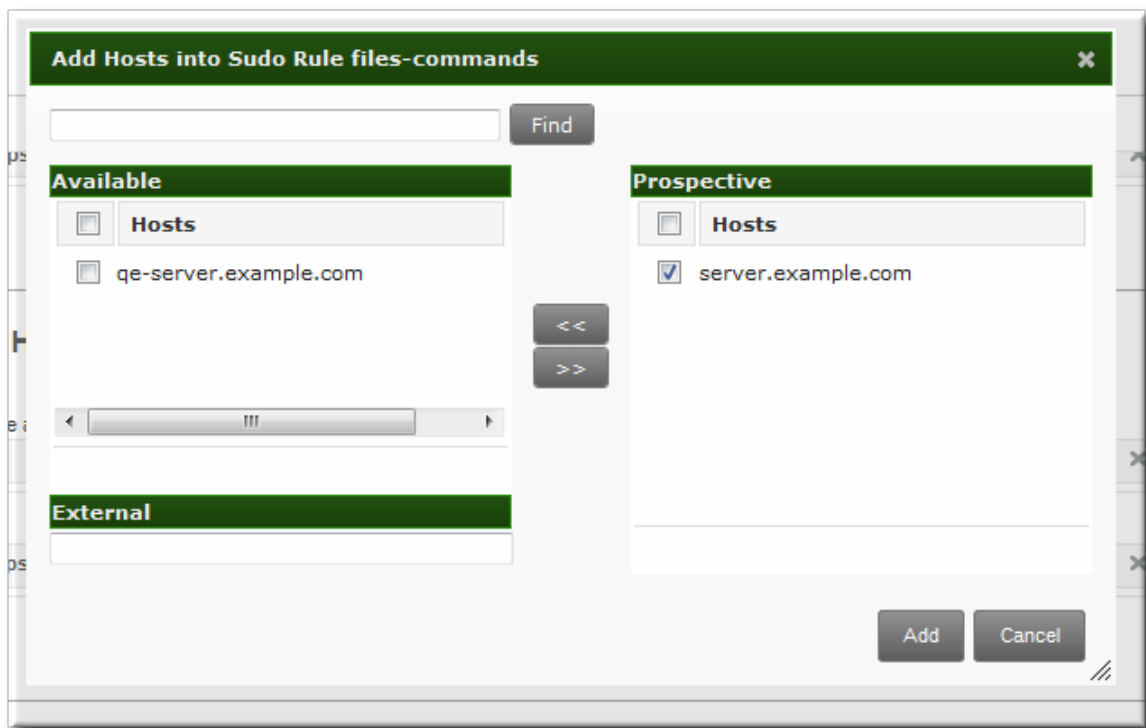
- c. Click **Add**.

It is possible to configure both IdM users and external system users ([Section 20.3.1, “About External Users”](#)).

8. In the **Access This Host** area, select the hosts on which the sudo rule is in effect.
- Click the + **Add** link at the right of the hosts list.



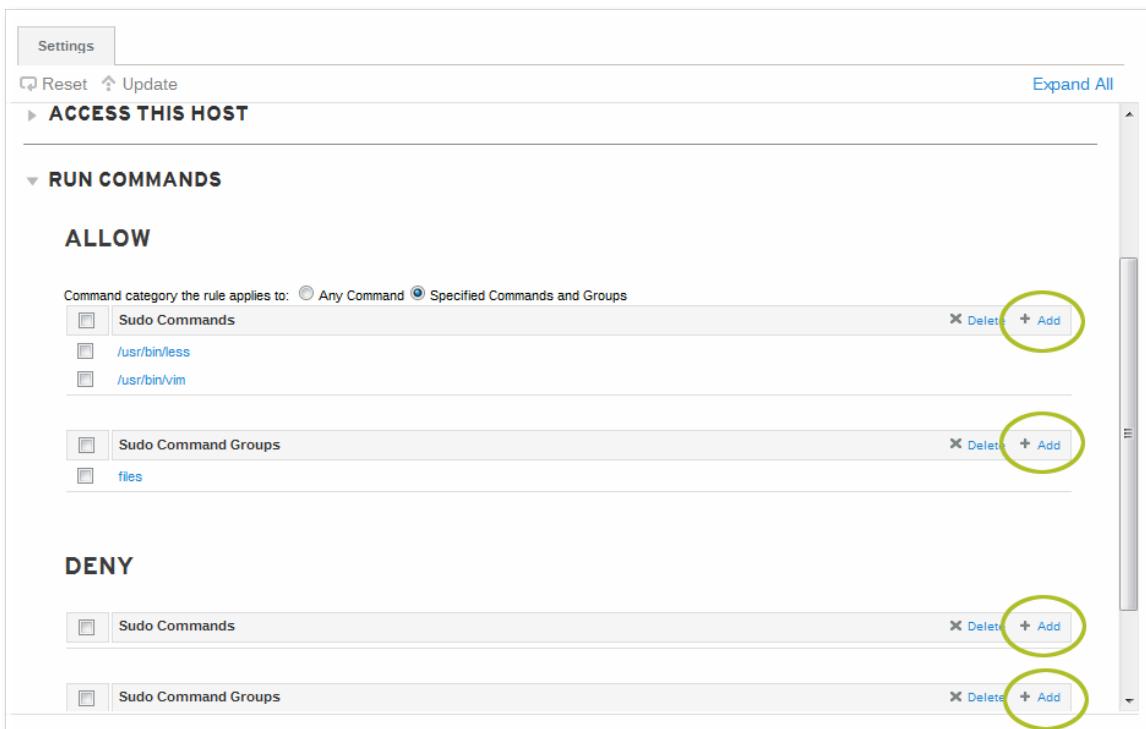
- Click the checkbox by the hosts to include with the rule, and click the right arrows button, **>>**, to move the hosts to the selection box.



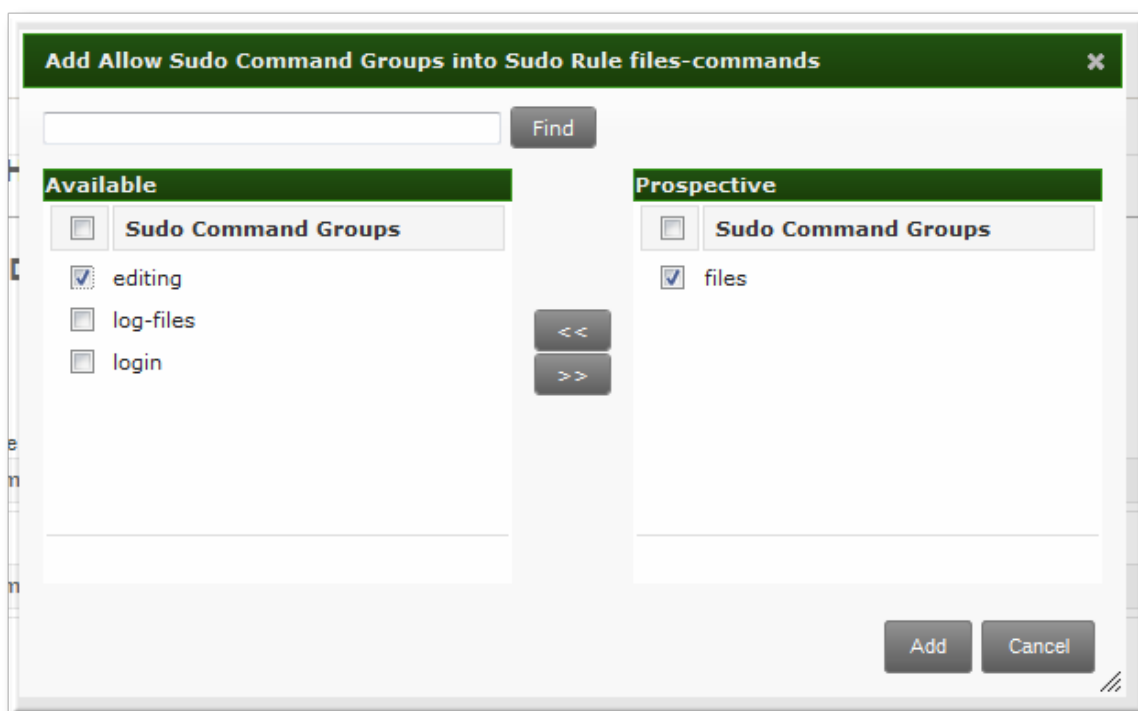
c. Click **Add**.

9. In the **Run Commands** area, select the commands which are included in the sudo rule. The **sudo** rule can grant access or deny access to commands, and it can grant allow access to one command and also deny access to another.

a. In the **Allow/Deny** area, click the + **Add** link at the right of the commands list.



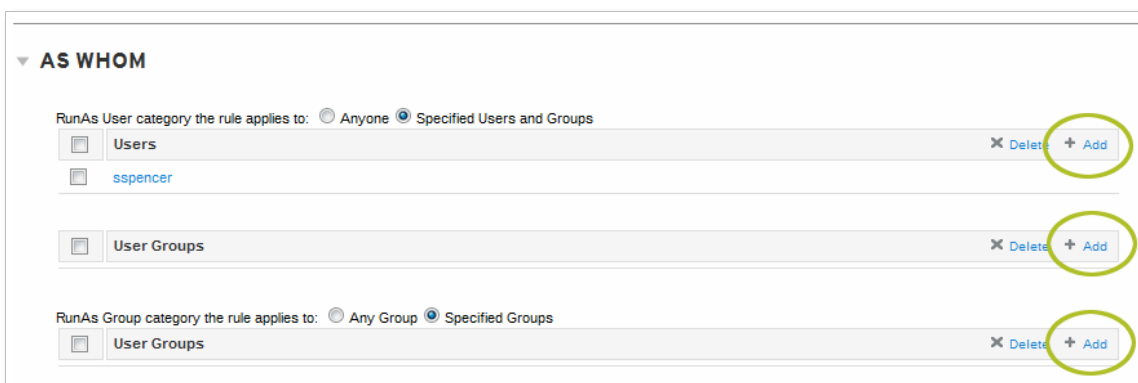
b. Click the checkbox by the commands or command groups to include with the rule, and click the right arrows button, **>>**, to move the commands to the selection box.



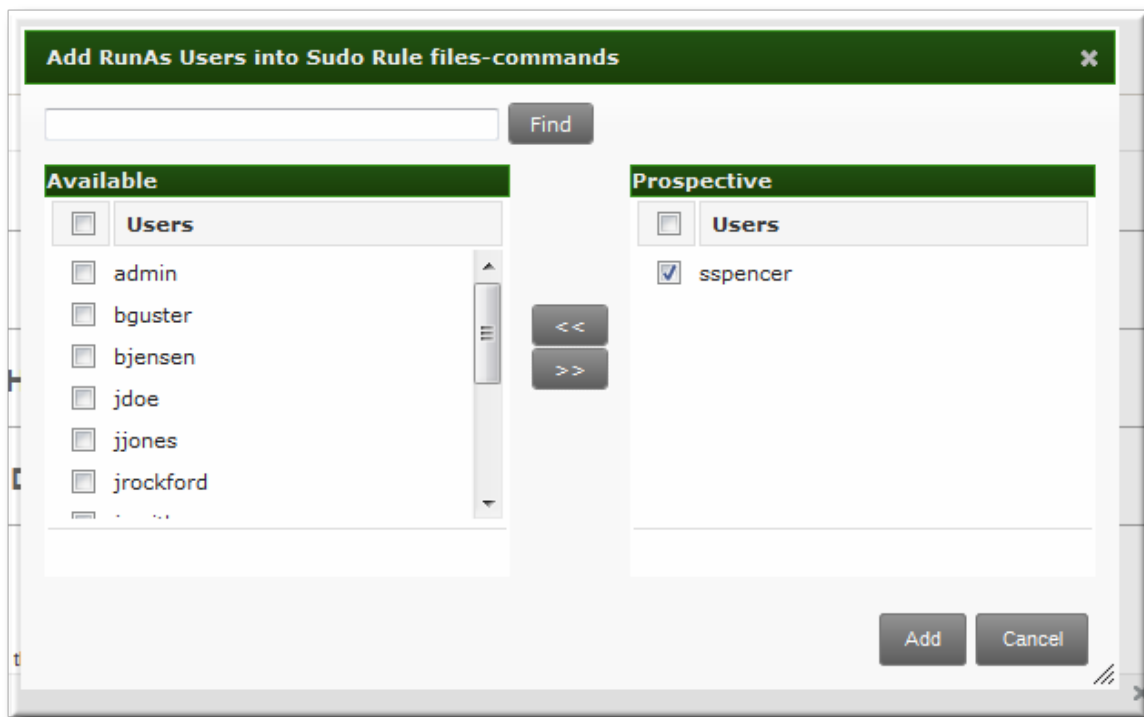
c. Click **Add**.

10. *Optional.* The sudo rule can be configured to run the given commands as a specific, non-root user.

a. In the **As Whom** area, click the + **Add** link at the right of the users list.



b. Click the checkbox by the users to run the command as, and click the right arrows button, **>>**, to move the users to the selection box.



c. Click **Add**.

20.3.4. Defining sudo Rules in the Command Line

Each element is added to the rule command using a different command (listed in [Table 20.1, “sudo Commands”](#)).

The basic outline of a **sudo** rule command is:

```
$ ipa sudorule-add* options ruleName
```

Example 20.1. Creating Basic sudo Rules

In the most basic case, the **sudo** configuration is going to grant the right to one user for one command on one host.

The first step is to add the initial rule entry.

```
$ ipa sudorule-add files-commands
-----
Added sudo rule "files-commands"
-----
Rule name: files-commands
Enabled: TRUE
```

Next, add the commands to *grant* access to. This can be a single command, using **--sudocmd**, or a group of commands, using **--sudocmdgroups**.

```
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/vim" files-commands
Rule name: files-commands
Enabled: TRUE
sudo Commands: /usr/bin/vim
-----
Number of members added 1
-----
```

Add a host or a host group to the rule.

```
$ ipa sudorule-add-host --host server.example.com files-commands
Rule name: files-commands
Enabled: TRUE
Hosts: server.example.com
sudo Commands: /usr/bin/vim
-----
Number of members added 1
-----
```

Last, add the user or group to the rule. This is the user who is allowed to use **sudo** as defined in the rule; if no "run-as" user is given, then this user will run the **sudo** commands as root.

```
$ ipa sudorule-add-user --user jsmith files-commands
Rule name: files-commands
Enabled: TRUE
Users: jsmith
Hosts: server.example.com
sudo Commands: /usr/bin/vim"
-----
Number of members added 1
-----
```

Example 20.2. Allowing and Denying Commands

The **sudo** rule can grant access or deny access to commands. For example, this rule would allow read access to files but prevent editing:

```
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/less" readfiles
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/tail" readfiles
$ ipa sudorule-add-deny-command --sudocmd "/usr/bin/vim" readfiles
```

Example 20.3. Using sudoers Options

The **sudoers** file has a lot of potential flags that can be set to control the behavior of **sudo** users, like requiring (or not requiring) passwords to offer a user to authenticate to **sudo** or using fully-qualified domain names in the **sudoers** file. The complete list of options is in the **sudoers** manpage and at http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options.

Any of these options can be set for the IdM **sudo** rule using the **sudorule-add-option** command. When the command is run, it prompts for the option to add:

```
$ ipa sudorule-add-option readfiles
Sudo Option: !authenticate
-----
Added option "!authenticate" to Sudo rule "readfiles"
-----
```



NOTE

As described in [Section 20.3.2, “About sudo Options Format”](#), do not use options with whitespace in the values. Rather than adding a list of options in one line, add a single option setting for each desired option.

Example 20.4. Running as Other Users

The **sudo** rule also has the option of specifying a non-root user or group to run the commands as. The initial rule has the user or group specified using the **--sudorule-add-runasuser** or **--sudorule-add-runasgroup** command, respectively.

```
$ ipa sudorule-add-runasuser --users=jsmith readfiles
$ ipa sudorule-add-runasgroup --groups=ITadmins readfiles
```

When creating a rule, the **sudorule-add-runasuser** or **sudorule-add-runasgroup** command can only set *specific* users or groups. However, when editing a rule, it is possible to run **sudo** as all users or all groups by using the **--runasusercat** or **--runasgroupcat** option. For example:

```
$ ipa sudorule-mod --runasgroupcat=all ruleName
```


**NOTE**

The **--sudorule-add-runasuser** and **--sudorule-add-runasgroup** commands do not support an **all** option, only specific user or group names. Specifying all users or all groups can only be used with options with the **sudorule-mod** command.

Example 20.5. Referencing External Users

The "who" in a **sudo** rule can be an IdM user, but there are many logical and useful rules where one of the referents is a system user. Similarly, a rule may need to grant or deny access to a host machine on the network which is not an IdM client.

In those cases, the **sudo** policy can refer to an *external* user — an identity created and stored outside of IdM ([Section 20.3.1, "About External Users"](#)).

The options to add an external identity to a **sudo** rule are:

- ▶ **--externaluser**
- ▶ **--runasexternaluser**

For example:

```
$ ipa sudorule-add-user --externaluser=ITAdmin readfiles
$ ipa sudorule-add-runasuser --runasexternaluser=root readfiles
```

Table 20.1. sudo Commands

Command	Description
sudorule-add	Adds a sudo rule entry.
sudorule-add-user	Adds a user or a user group to the sudo rule. This user (or every member of the group) is then entitled to sudo any of the commands in the rule.
sudorule-add-host	Adds a target host for the rule. These are the hosts where the users are granted sudo permissions.
sudorule-add-runasgroup	Sets a group to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using sudo-rule .
sudorule-add-runasuser	Sets a user to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using sudo-rule .
sudorule-add-allow-command	Adds a command that users in the rule have sudo permission to run.
sudorule-add-deny-command	Adds a command that users in the rule are explicitly <i>denied</i> sudo permission to run.
sudorule-add-option	Adds a sudoers flag to the sudo rule.

20.3.5. Suspending and Removing sudo Rules

sudo rules that are defined through Identity Management must be deleted. Local rules can be enabled or disabled so that they are temporarily suspended, but this is not possible in Identity Management because of how the rules are stored and applied.

The only way to suspend a **sudo** rule is to (permanently) delete it. For example:

```
$ ipa sudorule-remove files-commands
```



NOTE

It is not possible to enable or disable a **sudo** rule in Identity Management **even though there are enable/disable buttons in the IdM web UI**. These buttons do not work and no modifications to enable or disable a **sudo** rule are saved.

20.4. Configuring Hosts to Use IdM sudo Policies

Actually implementing **sudo** policies is more complicated than simply creating the rules in IdM. Those rules need to be applied to every local machine, which means that each system in the IdM domain has to be configured to refer to IdM (as an LDAP server) for its policies.

20.4.1. Applying the sudo Policies to Hosts Using LDAP

This example specifically configures a Red Hat Enterprise Linux 6 client for sudo rules. The NSLCD configuration is different, depending on the platform.

1. Set up the host and sudo entries in IdM.

- a. *Optional.* Set up a host group, as described in [Section 10.6, “Managing Host Groups”](#).
- b. *Optional.* Create a user group and add the users, as described in [Section 9.10.1, “Creating User Groups”](#).
- c. Set up the sudo commands and command groups, as described in [Section 20.2, “Setting up sudo Commands and Command Groups”](#).
- d. Set up the sudo rules, as described in [Section 20.3, “Defining sudo Rules”](#).

2. Set up a bind (authenticated) user by setting a password for the default IdM **sudo** user. The user must be able to authenticate to the server; anonymous access is not supported for **sudo** policies.

Using LDAP tools, set the password for the **sudo** user,

uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com. For example:

```
[jsmith@server ~]$ ldappasswd -Y GSSAPI -S -h ipaserver.example.com
uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com
New password:
Re-enter new password:
Enter LDAP Password:
```

3. Configure every system in the IdM domain to use SSSD for sudo rules.

- a. Configure **sudo** to look to LDAP for the **sudoers** file.

```
vim /etc/nsswitch.conf

sudoers: files ldap
```

Leaving the **files** option in place allows **sudo** to check its local configuration before checking the LDAP-based IdM configuration.

- b. Enable debug logging for **sudo** operations in the `/etc/ldap.conf` file. If this file does not exist, it can be created.

```
vim /etc/ldap.conf

sudoers_debug: 1
```



TIP

Adding the **sudoers_debug** parameter helps with troubleshooting. Valid values for this parameter are 0, 1, and 2. The **sudo** documentation at http://www.gratisoft.us/sudo/readme_ldap.html has more information on debugging the process.

- c. Edit the NSS/LDAP configuration file and add the following **sudo**-related lines to the `/etc/sudo-ldap.conf` file:

```
binddn uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com
bindpw sudo_password

ssl start_tls
tls_cacertfile /etc/ipa/ca.crt
tls_checkpeer yes

bind_timelimit 5
timelimit 15

uri ldap://ipaserver.example.com ldap://backup.example.com:3890
sudoers_base ou=SUDOers,dc=example,dc=com
```

Multiple LDAP servers can be configured in a space-separated list, and other options (like SSL and non-standard ports) can be used with the LDAP URL. The **sudo** LDAP configuration is covered in the **sudo** manpages, <http://www.sudo.ws/sudo/man/1.8.2/sudoers.ldap.man.html>.



IMPORTANT

The **uri** directive must give the fully-qualified domain name of the LDAP server, not an IP address. Otherwise, **sudo** fails to connect to the LDAP server.

- d. Optionally, enable debugging in SSSD to show what LDAP settings it is using.

```
[root@server ~]# vim /etc/sss/sss.conf

[domain/LDAPDOMAIN]
debug_level = 6
....
```

The LDAP search base used by SSSD for operations is recorded in the

sssd_DOMAINNAME.log log.

- e. Set a name for the NIS domain in the **sudo** configuration. **sudo** uses NIS-style netgroups, so the NIS domain name must be set in the system configuration for **sudo** to be able to find the host groups used in the IdM **sudo** configuration.
 - a. Set the NIS domain name to use with the **sudo** rules.

```
[root@server ~]# nisdomainname example.com
```

- b. Configure the system authentication settings to persist the NIS domain name. For example:

```
[root@server ~]# authconfig --nisdomain=example.com --update
```

This updates the **/etc/sysconfig/network** and **/etc/yp.conf** files with the NIS domain.



IMPORTANT

Even though **sudo** uses NIS-style netgroups, **it is not necessary to have a NIS server installed**. Netgroups require that a NIS domain be named in their configuration, so **sudo** requires that a NIS domain be named for netgroups. However, that NIS domain does not actually need to exist.

20.4.2. Applying the Configured sudo Policies to Hosts Using SSSD

1. **Set up the host and sudo entries in IdM.**
 - a. *Optional.* Set up a host group, as described in [Section 10.6, “Managing Host Groups”](#).
 - b. *Optional.* Create a user group and add the users, as described in [Section 9.10.1, “Creating User Groups”](#).
 - c. Set up the sudo commands and command groups, as described in [Section 20.2, “Setting up sudo Commands and Command Groups”](#).
 - d. Set up the sudo rules, as described in [Section 20.3, “Defining sudo Rules”](#).
2. **Configure every system in the IdM domain to use SSSD for sudo rules.**
 - a. Configure **sudo** to look to SSSD for the **sudoeers** file.

```
vim /etc/nsswitch.conf

sudoers: files sss
```

Leaving the **files** option in place allows **sudo** to check its local configuration before checking SSSD for the IdM configuration.

- b. Add **sudo** to the list of services managed by the local SSSD client.

```
[root@server ~]# vim /etc/sss/sss.conf

[sss]
config_file_version = 2
services = nss, pam, sudo
domains = IPADOMAIN
```

- c. Optionally, enable debugging in SSSD to show what LDAP settings it is using.

```
[domain/IPADOMAIN]
debug_level = 6
....
```

The LDAP search base used by SSSD for operations is recorded in the **sssd_DOMAINNAME.log** log.

- d. Set a name for the NIS domain in the **sudo** configuration. **sudo** uses NIS-style netgroups, so the NIS domain name must be set in the system configuration for **sudo** to be able to find the host groups used in the IdM **sudo** configuration.
 - a. Set the NIS domain name to use with the **sudo** rules.

```
[root@server ~]# nisdomainname example.com
```

- b. Configure the system authentication settings to persist the NIS domain name. For example:

```
[root@server ~]# authconfig --nisdomain=example.com --update
```

This updates the **/etc/sysconfig/network** and **/etc/yp.conf** files with the NIS domain.



IMPORTANT

Even though **sudo** uses NIS-style netgroups, **it is not necessary to have a NIS server installed**. Netgroups require that a NIS domain be named in their configuration, so **sudo** requires that a NIS domain be named for netgroups. However, that NIS domain does not actually need to exist.

Chapter 21. Policy: Configuring Host-Based Access Control

IdM can control access to both machines and the services on those machines within the IdM domain. The rules define who can access what within the domain, not the level of access (which are defined by system or application settings). These access control rules grant access, with all other users and hosts implicitly denied.

This is called *host-based access control* because the rule defines what hosts (*targets*) within the domain a user is allowed to access. This access can be further broken down to users and services on those hosts.



NOTE

Using host-based access control requires SSSD to be installed and configured on the IdM client machine.

21.1. About Host-Based Access Control

Host-based access control rules (which are described in [Chapter 21, Policy: Configuring Host-Based Access Control](#)) can be applied to individual hosts. However, using host groups allows centralized, and potentially simplified, access control management because an access control rule only needs to be defined once and then it is applied immediately and consistently to all the hosts within the group.

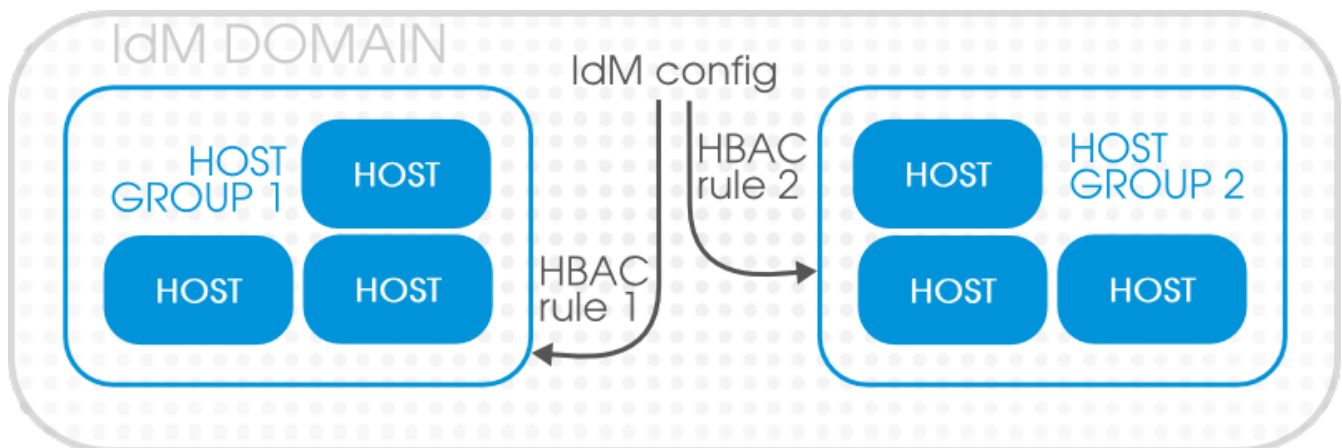


Figure 21.1. Host Groups and Host-Based Access Control



NOTE

While access must be explicitly granted to users and hosts within the IdM domain, IdM servers are configured by default with an **allow all** access control rule which allows access for every host within the domain to every host within the domain.

To create an IdM server without the default **allow all** rule, run **ipa-server-install** with the **--no_hbac_allow** option.

The *rule* first defines things that can be accessed, and there are two types of entities:

- *Hosts*, or target hosts, within the IdM domain.

- **Services on the target hosts.** Multiple services can be combined into *service groups*. The service group can be modified without having to edit the access control rule itself.

The rule also sets *who can have access* (the IdM domain user).



TIP

It is possible to use categories for users and target hosts instead of adding each one individually to the access control rule. The only supported category is **all**.

The entities in host-based access control rules follow the Kerberos principal entries: users, hosts (machines), and services. Users and target hosts can be added directly to host-based access control rules. However, services must be flagged first and then added to the access control rules.

21.2. Creating Host-Based Access Control Entries for Services and Service Groups

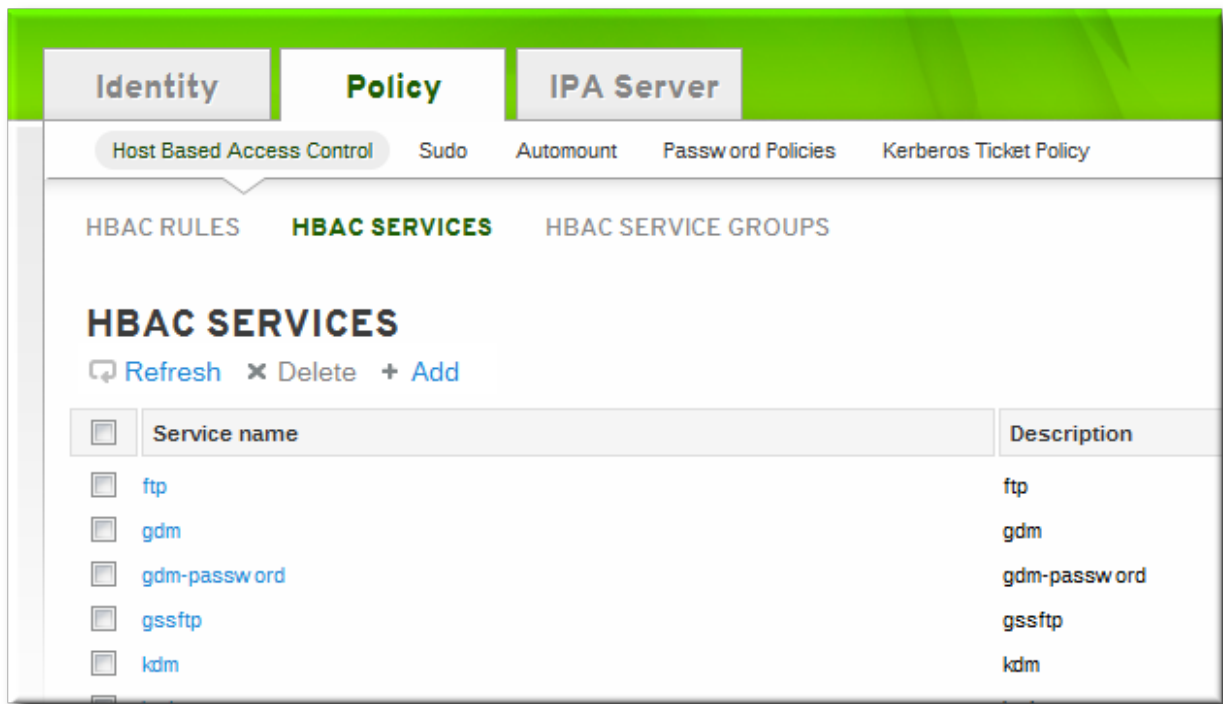
Any PAM service can be identified as to the host-based access control (HBAC) system in IdM. The service entries used in host-based access control are separate from adding a service to the IdM domain. Adding a service to the domain makes it a recognized resource which is available to other resources. Adding a domain resource to the host-based access control configuration allows administrators to exert defined control over what domain users and what domain clients can access that service.

Some common services are already configured as HBAC services, so they can be used in host-based access control rules. Additional services can be added, and services can be added into service groups for simpler management.

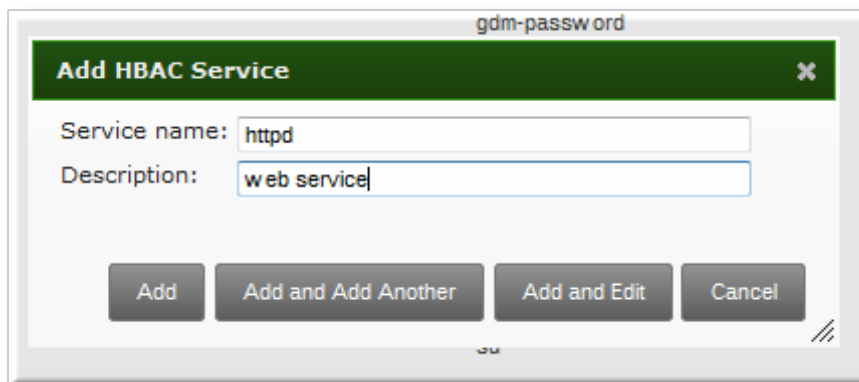
21.2.1. Adding HBAC Services

21.2.1.1. Adding HBAC Services in the Web UI

1. Click the **Policy** tab.
2. Click the **Host-Based Access Control** subtab, and then select the **HBAC Services** link.
3. Click the **Add** link at the top of the list of services.



4. Enter the service name and a description.



5. Click the **Add** button to save the new service.

6. If a service group already exists, then add the service to the desired group, as described in [Section 21.2.2.1, “Adding Service Groups in the Web UI”](#).

21.2.1.2. Adding Services in the Command Line

The service is added to the access control system using the **hbacsvc-add** command, specifying the service by the name that PAM uses to evaluate the service.

For example, this adds the **tftp** service:

```
# ipa hbacsvc-add --desc="TFTP service" tftp
-----
Added HBAC service "tftp"
-----
```

If a service group already exists, then the service can be added to the group using the **hbacsvcgroup-add-member** command, as in [Section 21.2.2.2, “Adding Service Groups in the Command Line”](#).

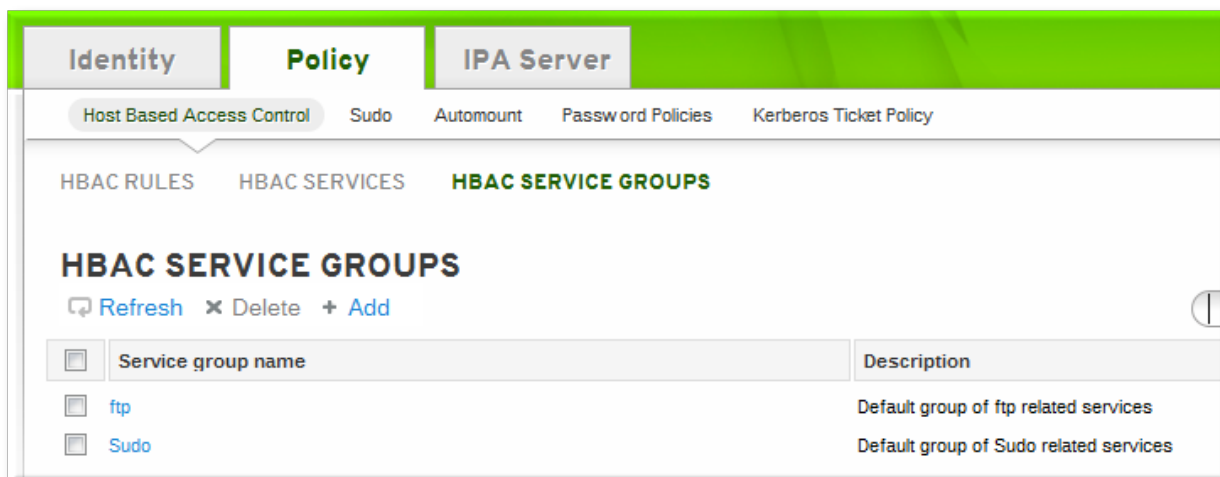
21.2.2. Adding Service Groups

Once the individual service is added, it can be added to the access control rule. However, if there a large

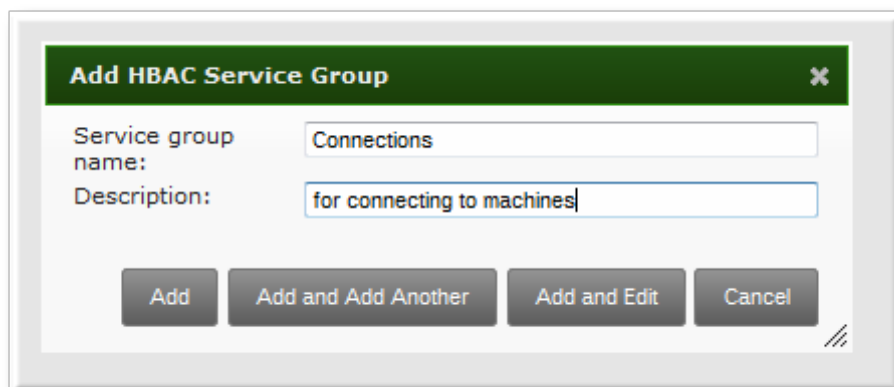
number of services, then it can require frequent updates to the access control rules as services change. Identity Management also allows groups of services to be added to access control rules. This makes it much easier to manage access control, because the members of the service group can be changed without having to edit the rule itself.

21.2.2.1. Adding Service Groups in the Web UI

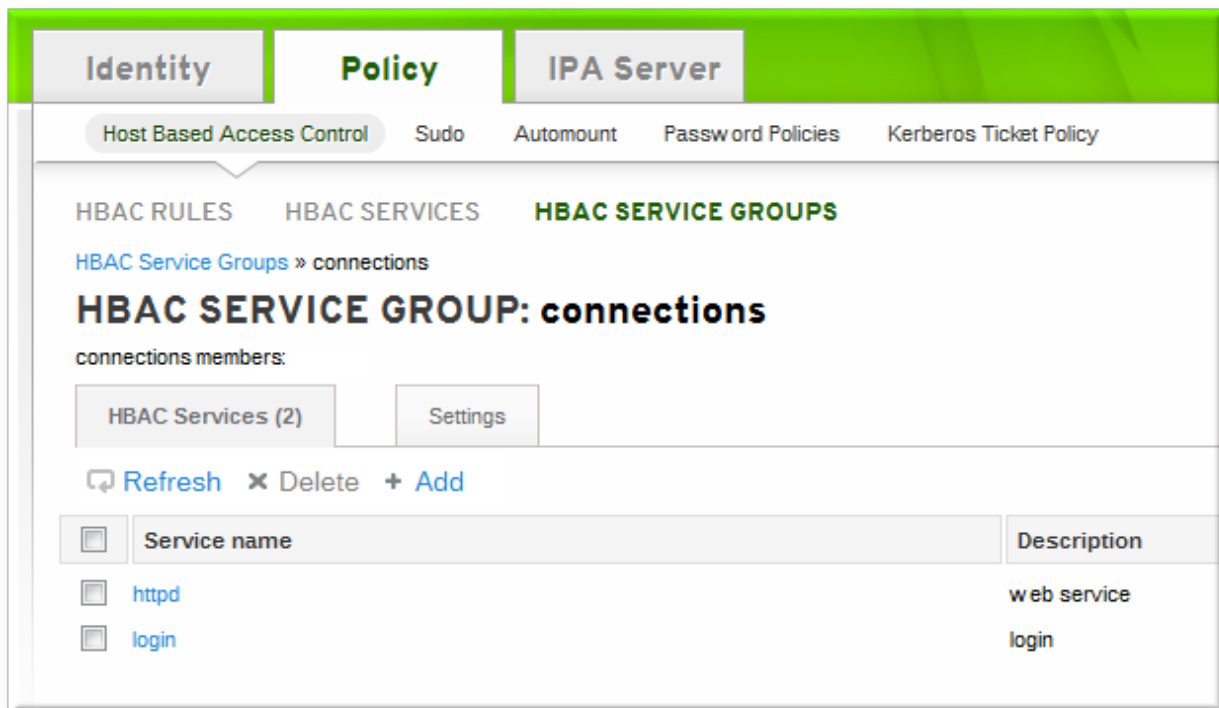
1. Click the **Policy** tab.
2. Click the **Host-Based Access Control** subtab, and then select the **HBAC Service Groups** link.
3. Click the **Add** link at the top of the list of service groups.



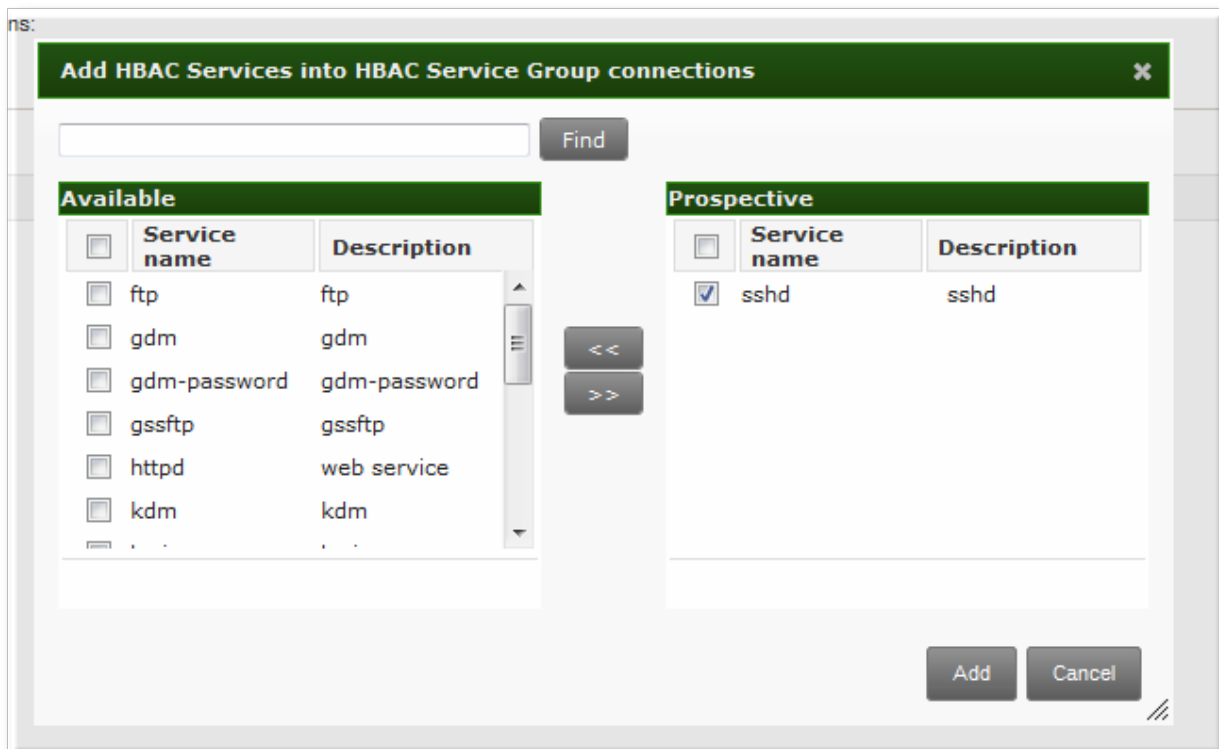
4. Enter the service group name and a description.



5. Click the **Add and Edit** button to go immediately to the service configuration page.
6. At the top of the **HBAC Services** tab, click the **Add** link.



7. Click the checkbox by the names of the services to add, and click the right arrows button, >>, to move the command to the selection box.



8. Click the **Add** button to save the group membership.

21.2.2.2. Adding Service Groups in the Command Line

First create the service group entry, then create the service, and then add that service to the service group as a member. For example:

```
$ ipa hbacsvgroup-add --desc="login services" login
-----
Added HBAC service group "login"
-----
Service group name: login
Description: login services

$ ipa hbacsvc-add --desc="SSHD service" sshd
-----
Added HBAC service "sshd"
-----

$ ipa hbacsvgroup-add-member --hbacsvcs=sshd login
Service group name: login
Description: login services
-----
Number of members added 1
-----
```



NOTE

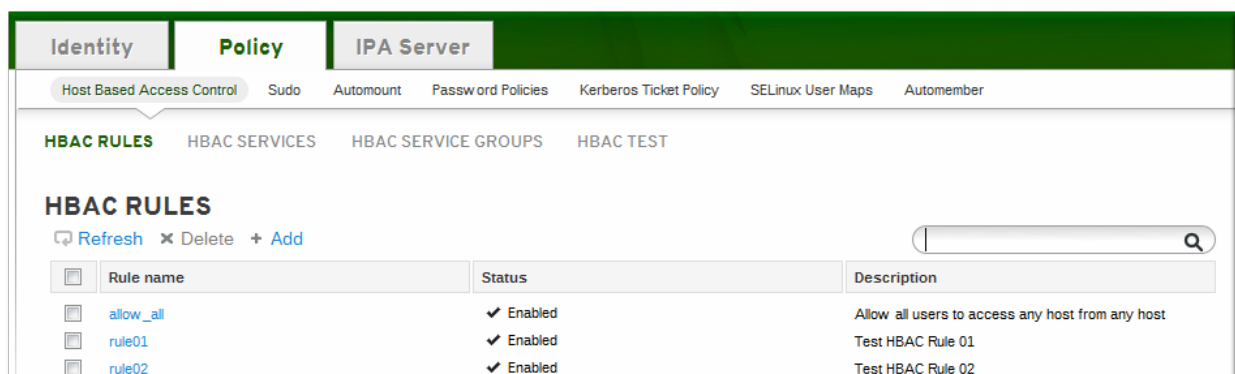
IdM defines two default service groups: **SUDO** for sudo services and **FTP** for services which provide FTP access.

21.3. Defining Host-Based Access Control Rules

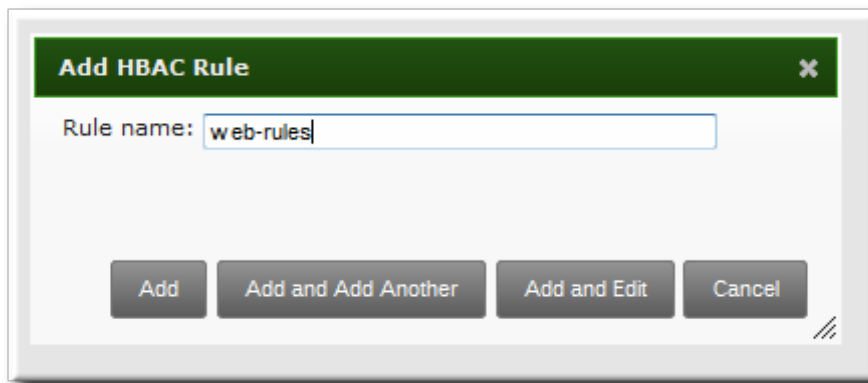
Access controls, at a high level, define *who* has access to *what*. The *who* is an IdM user, and the *what* can be either a host (target host), service, or service group, or a combination of the three.

21.3.1. Setting Host-Based Access Control Rules in the Web UI

1. Click the **Policy** tab.
2. Click the **Host-Based Access Control** subtab, and then select the **HBAC Rules** link.
3. Click the **Add** link at the top of the list of host-based access control rules.



4. Enter the name for the rule.



5. Click the **Add and Edit** button to go immediately to set the configuration for the rule.

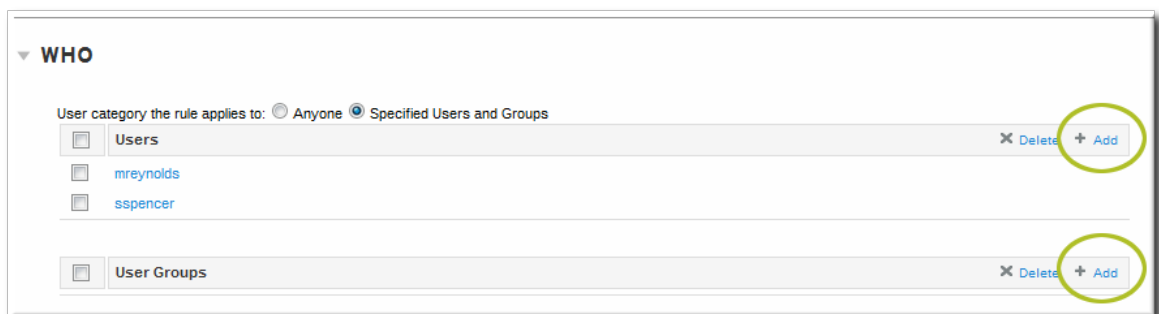
There are a number of configuration areas for the rule. The four basic elements are *who* the rule applies to, what hosts allow access (the target), and, optionally, what services can be accessed.

6. In the **who** area, select the users or user groups to which the access control rule is applied.

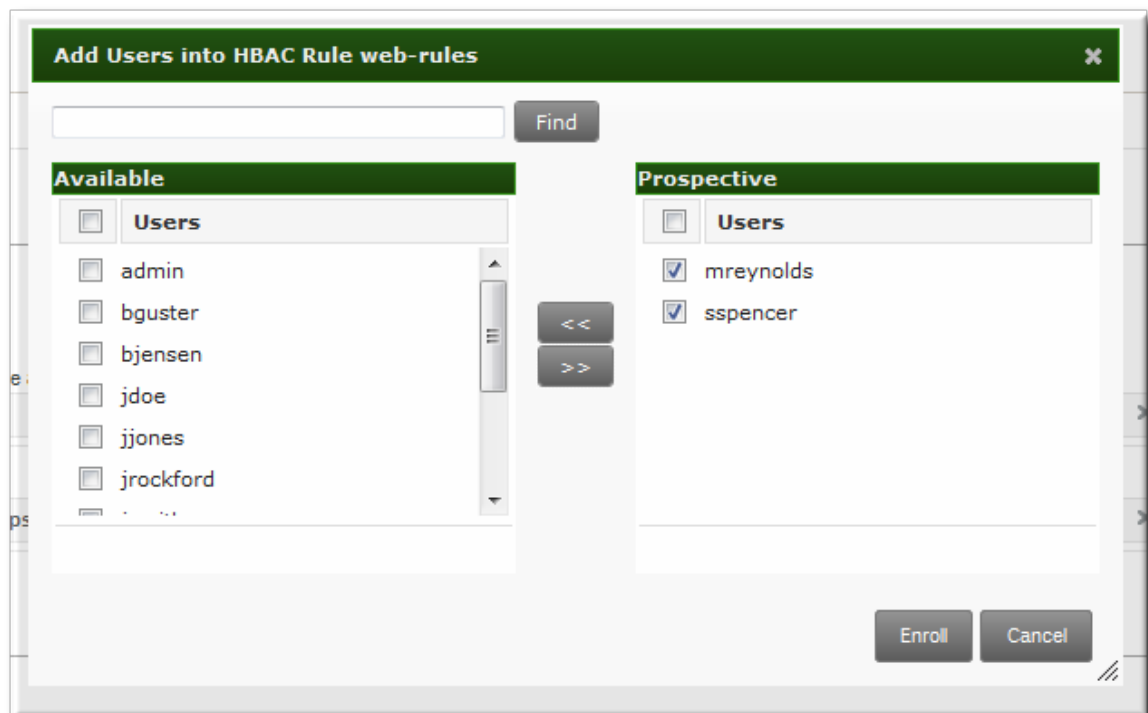
To apply the rule to all IdM users, select the **Anyone** radio button.

To apply the rule to a specific set of users or user groups:

- Select the **Specified Users and Groups** radio button.
- Click the **+ Add** link at the right of the users list.



- Click the checkbox by the users to add to the rule, and click the right arrows button, **>>**, to move the users to the selection box.



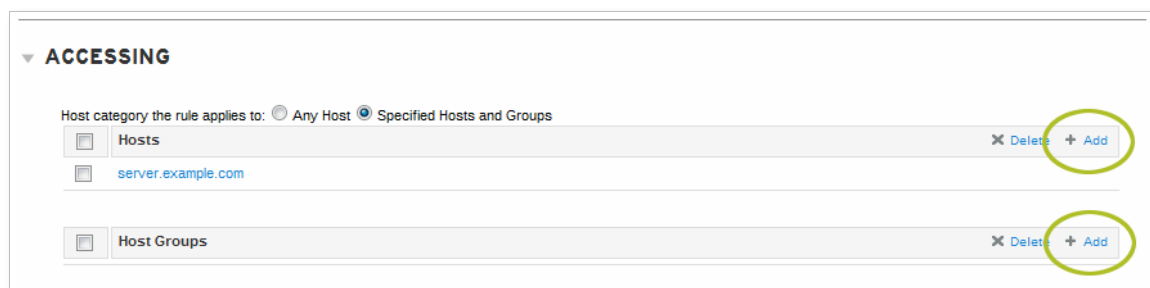
d. Click **Add**.

7. In the **Accessing** area, select the target hosts which can be accessed through this access control rule.

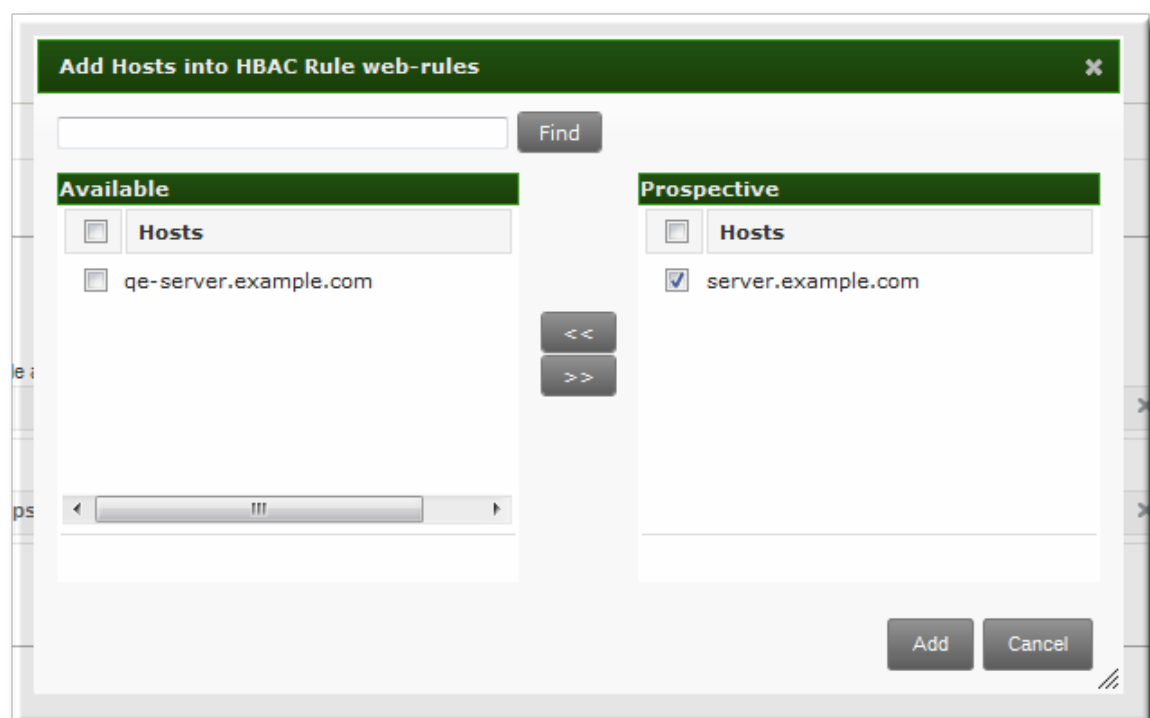
To apply the rule to all IdM hosts, select the **Any Host** radio button.

To apply the rule to a specific set of hosts or host groups:

- Select the **Specified Hosts and Groups** radio button.
- Click the **+ Add** link at the right of the hosts list.



- Click the checkbox by the hosts to include with the rule, and click the right arrows button, **>>**, to move the hosts to the selection box.



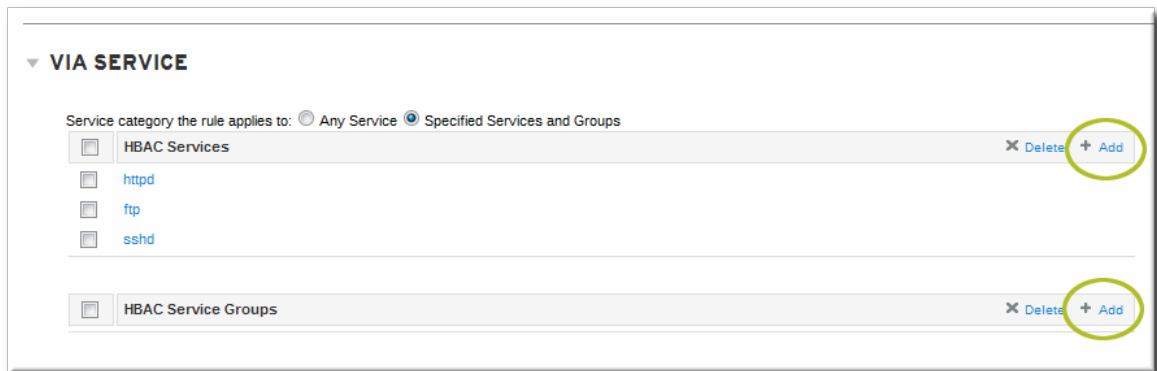
d. Click **Add**.

8. In the **Via Service** area, select specific services on the target hosts which the users are allowed to use to access target machines.

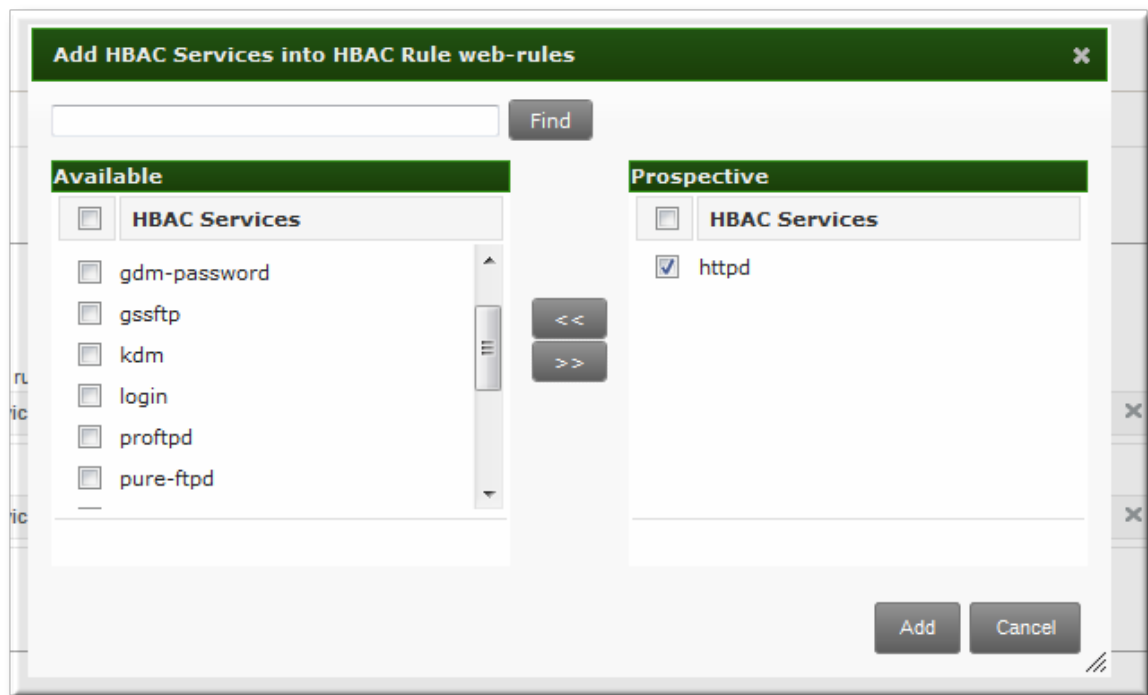
To apply the rule to all IdM hosts, select the **Any Service** radio button.

To apply the rule to a specific set of hosts or host groups:

- Select the **Specified Services and Groups** radio button.
- Click the **+ Add** link at the right of the commands list.



- c. Click the checkbox by the services or groups to include with the rule, and click the right arrows button, >>, to move the services to the selection box.



- d. Click **Add**.

21.3.2. Setting Host-Based Access Control Rules in the Command Line

Access control rules are created using the **hbacrule-*** commands (listed in [Table 21.1, “Host-Based Access Control Command and Options”](#)). The first step is to create a container entry; from there, users, hosts, and services can be added to the access control entry.

The basic outline of all the access control commands is:

```
$ ipa hbacrule-add* options ruleName
```



TIP

To set every user or every host as a target, use the category options, such as **--usercat=all**.

Example 21.1. Granting All Access to One Host

One simple rule is to grant every user access to a single server. The first command creates the entry and uses the category options to apply every user.

```
$ ipa hbacrule-add --usercat=all allGroup
-----
Added HBAC rule "allGroup"
-----
Rule name: allGroup
User category: all
Enabled: TRUE
```

The second rule adds the target host, using the **hbacrule-add-host** command:

```
$ ipa hbacrule-add-host --hosts=server.example.com allGroup
Rule name: allGroup
User category: all
Enabled: TRUE
Successful hosts/hostgroups:
  member host: server.example.com
-----
Number of members added 1
-----
```

Example 21.2. Adding Control for a Single User to a Service

Another access control method is to specify which services users are allowed to use to access the target hosts.

First, for the user to have access to every machine, every host must be added as both a host and target. This can be done using the category options:

```
$ ipa hbacrule-add --hostcat=all sshd-jsmith
```

Since the access control rule applies to a specific user, the user is added to the rule using the **hbacrule-add-user** command:

```
$ ipa hbacrule-add-user --users=jsmith sshd-jsmith
```

Then, the service is added to the access control rule. (The service should have already been added to the access control system using the **hbacsvc-add** command.) This is the service that the user can use to connect to the machine.

```
$ ipa hbacrule-add-service --hbacsvcs=sshd sshd-jsmith
```

Example 21.3. Adding a Service Group to the Rule

While a single service can be added to a rule, it is also possible to add an entire service group. Like a single service, this uses the **hbacrule-add-service** command, only with the **--hbacsvcgroups** option that specifies the group name.

```
$ ipa hbacrule-add-service --hbacsvcgroups=login loginRule
```


Table 21.1. Host-Based Access Control Command and Options

Command	Description	Arguments	Source or Target Entry
hbacrule-add	Adds a new host-based access control rule.	▸ <code>--usercat=all</code>, which applies the rule to every user ▸ <code>--hostcat=all</code>, which sets every host as an allowed target server ▸ <code>--servicecat=all</code>, which sets every configured service as an allowed target service ▸ <i>ruleName</i>, which is the required unique identifier for the new rule	
hbacrule-add-host	Adds a target host to the access control rule. A target host can be accessed <i>by</i> other servers and users in the domain.	▸ <code>--hosts</code>, which adds an individual server or command-separated list of servers as an allowed target server ▸ <code>--hostgroups</code>, which adds a host group to the rule and every host within the host group is an allowed target server ▸ <i>ruleName</i>, which is the rule to which to add the target server	Target
hbacrule-add-service	Adds a service type to the rule.	▸ <code>--hbacsvcs</code>, which adds an individual service type or a comma-separated list of service type as an allowed target service ▸ <code>--hbacsvcgroups</code>, which adds a service group to the rule and every	Target

		service within the service group is an allowed target service ▶ <i>ruleName</i>, which is the rule to which to add the target service	
hbacrule-add-user	Adds a user to the access control rule. The user is then able to access any allowed target host or service within the domain.	▶ <i>--users</i>, which adds an individual user or command-separated list of users to the rule ▶ <i>--groups</i>, which adds a user group to the rule and, thus, every user within the group ▶ <i>ruleName</i>, which is the rule to which to add the user	Source
hbacrule-disable hbacrule-enable	Disables or enables a host-based access control rule. Rules can be disabled if their behavior needs to be evaluated (for troubleshooting or to test a new rule).	<i>ruleName</i>, which is the rule to disable or enable	

21.4. Testing Host-Based Access Control Rules

Implementing host-based access controls effectively can be tricky because it requires that all of the hosts be properly configured and the access is properly applied to users and services.

The **hbactest** command can test different host-based access control scenarios to make sure that the rules are working as expected.



NOTE

The **hbactest** command does not work with trusted Active Directory users. Active Directory user/group associations are determined dynamically, as a user logs in, and those data are not stored in the IdM LDAP directory. The **hbactest** command, then, is unable to resolve the group memberships to check how access control rules will be applied.

21.4.1. The Limits of Host-Based Access Control Configuration

The access control configuration should always be tested before it is implemented to prevent

authorization failures.

Host-based access control rules depend on a lot of interactions — between hosts, services, DNS lookups, and users. If any element is misconfigured, then the rule can behave in unexpected ways.

Identity Management includes a testing tool to verify that access control rules are behaving in the expected way by testing the access in a defined scenario. There are several situations where this testing is useful:

- ▶ A new rule needs to be tested before it is implemented.
- ▶ There are problems with the existing rules, and the testing tool can identify what rule is behaving badly.
- ▶ A subset of existing rules can be tested to see how they are performing.

21.4.2. Test Scenarios for Host-Based Access Control (CLI-Based)



NOTE

The **hbactest** command does not work with trusted Active Directory users. Active Directory user/group associations are determined dynamically, as a user logs in, and those data are not stored in the IdM LDAP directory. The **hbactest** command, then, is unable to resolve the group memberships to check how access control rules will be applied.

The **hbactest** command tests configured host-based access control rules in very specific situations. A test run defines:

- ▶ The user to run the operation as to test the rule performance for that user (**--user**).
- ▶ Using the login client Y (**--service**).
- ▶ To target host Z (**--host**).
- ▶ The rule to test (**--rules**); if this is not used, then all enabled rules are tested.
- ▶ *Optional* The **hbactest** returns detailed information about which rules were matched, not matched, or invalid. This detailed rule output can be disabled using **--nodetail**, so the test simply runs and returns whether access was granted.



NOTE

The **hbactest** script does not actually connect to the target host. Instead, it uses the rules within the IdM database to simulate how those rules would be applied in a specific situation as if an SSSD client were connecting to the IdM server.

More briefly, it performs a simulated test run based on the given information and configuration, but it does not actually attempt a service request against the target host.

Example 21.4. Testing All Active Rules

The most basic command checks all active rules. It requires a specific connection scenario, so the user, login service and target host have to be given, and the testing tool checks the connection.

```
$ ipa hbactest --user=jsmith --host=target.example.com --service=ssh
-----
Access granted: True
-----
  notmatched: my-second-rule
  notmatched: my-third-rule
   matched: myrule
   matched: allow_all
```

Example 21.5. Testing a Specific Rule

It is possible to check a specific rule (or several rules).

```
$ ipa hbactest --user=jsmith --host=target.example.com --service=ssh --
rules=myrule
-----
Access granted: True
-----
  notmatched: myrule
```

Example 21.6. Testing Specific Rules Plus All Enabled

The **--rules** option lists specific rules to test, but it may be useful to test the specified rules against all of the enabled rules in the domain. This can be done by adding the **--enabled** option, which includes the (unspecified) enabled rules along with the specified rules.

```
$ ipa hbactest --user=jsmith --host=target.example.com --service=ssh --
rules=myrule --enabled
-----
Access granted: True
-----
  matched: my-second-rule
  notmatched: my-third-rule
   matched: myrule
   matched: allow_all
```

It is possible to run a similar comparison against *disabled* rules by using the **--disable** option. With the **--rules** option, the specified rule plus all of the disabled rules are checked. With the **--disabled** option, all disabled rules are checked.

21.4.3. Testing Host-Based Access Control Rules in the UI

As [Section 21.4.1, “The Limits of Host-Based Access Control Configuration”](#) details, misconfiguring a host-based access-control rule can result in unpredictable behavior when users or services attempt to connect to a remote host.

Testing host-based access control can help confirm that the rule performs as expected before it is

deployed or to troubleshoot a rule once it is already active.



NOTE

The **hbactest** command does not work with trusted Active Directory users. Active Directory user/group associations are determined dynamically, as a user logs in, and those data are not stored in the IdM LDAP directory. The **hbactest** command, then, is unable to resolve the group memberships to check how access control rules will be applied.

By the nature of host-based access control rules, a test must define and verify a very specific set of criteria. A test run defines:

- ▶ The user to run the operation as to test the rule performance for that user (**who**).
- ▶ To target host Z (**Accessing**).
- ▶ Using the login client Y (**Via Service**).
- ▶ The rule to test; if this is not used, then all enabled rules are tested (**Rules**).

The test environment is defined on the **HBAC TEST** page in the **Host Based Access Control** tab under **Policy**. A series of tabs is set up for each configuration step.

The screenshot shows the 'HBAC TEST' configuration page in the Red Hat Identity Management web interface. The 'Policy' tab is selected, and the 'HBAC TEST' sub-tab is active. The 'FROM' tab is selected, showing a table of hosts to test against. The table has columns for Host name, Description, and Enrolled?. Two hosts are listed: dev.example.com and test.example.com, both with 'True' in the Enrolled? column. Navigation buttons for 'Prev' and 'Next' are at the bottom right.

Host name	Description	Enrolled?
dev.example.com		True
test.example.com		

Showing 1 to 2 of 2 entries. [Prev](#) [Next](#) Page: 1 / 1

☐ Specify external Host:

Figure 21.2. The From Tab to Set up an HBAC Test

Once the environment is defined, then the test is run simply by clicking a button on the **Run Test** page. The results show clearly whether access was granted or denied to the users, and then runs through the rules which matched the given parameters.

Identity **Policy** **IPA Server**

Host Based Access Control Sudo Automount Password Policies Kerberos Ticket Policy SELinux User Maps Automember

HBAC RULES HBAC SERVICES HBAC SERVICE GROUPS **HBAC TEST**

RUN TEST

Who Accessing Via Service From Rules Run Test

Run Test **ACCESS GRANTED**

RULES

☒ Matched ☒ Unmatched

Rule name	Matched	Status	Description
allow_all	True	✓ Enabled	Allow all users to access any host from any host
rule01	True	✓ Enabled	Test HBAC Rule 01
rule02	True	✓ Enabled	Test HBAC Rule 02
rule03	True	✓ Enabled	Test HBAC Rule 03
rule04	True	— Disabled	Test HBAC Rule 04
rule05	True	✓ Enabled	Test HBAC Rule 05
rule06	True	✓ Enabled	
rule07	True	✓ Enabled	
rule08	True	✓ Enabled	
rule09	True	✓ Enabled	
rule10	True	✓ Enabled	
rule11		✓ Enabled	
rule12		✓ Enabled	
rule13		✓ Enabled	
rule14		✓ Enabled	
rule15		✓ Enabled	

Showing 1 to 20 of 21 entries.

Prev Next Page: 1 / 2

Prev New Test

Figure 21.3. HBAC Test Results

**NOTE**

To change some of the parameters and check for other results, click the **New Test** button at the bottom of the test results page. If that button is not selected, the form is not reset, so a new test will not run, even if test settings are changed.

Chapter 22. Policy: Defining SELinux User Maps

Security-enhanced Linux (SELinux) sets rules over what system users can access processes, files, directories, and system settings. Both the system administrator and applications themselves can define *security contexts* that restrict or allow user access and even access from other applications.

As part of defining centralized security policies in the Identity Management domain, Identity Management provides a way to map IdM users to SELinux users and automatically grant or restrict access to clients and services within the IdM domain, per host, based on the defined SELinux policies.

22.1. About Identity Management, SELinux, and Mapping Users

Security-enhanced Linux defines kernel-level, mandatory access controls for how users, processes, and applications can interact with other resources on a system. These rules for interactions, called *contexts*, look at the data and behavior characteristics of different objects on the system and then set rules, called *policies*, which create contexts based on the security implications of each specific object. This is in contrast to higher-level discretionary access controls which are concerned primarily with file ownership and user identity, without accounting for data criticality or application behavior.

System users are associated with an SELinux *role*. The role is assigned both a multi-layer security context (MLS) a multi-category security context (MCS). The MLS/MCS contexts *confine* users to what processes, files, and operations they can access on the system.

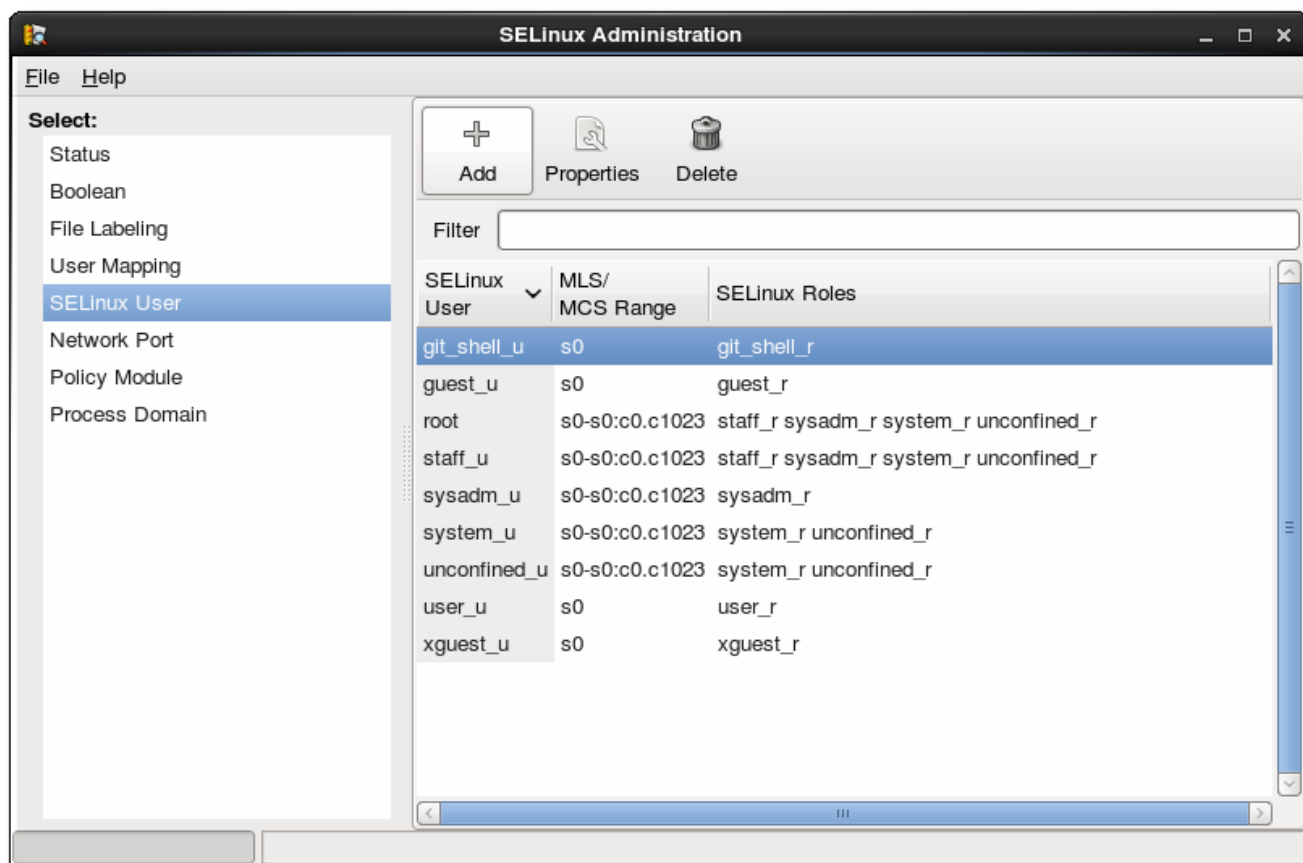


Figure 22.1. SELinux Users in the SELinux Manager

This is all described in detail in [Red Hat Enterprise Linux 6 Security-Enhanced Linux](#).

SELinux users and policies function at the system level, not the network level. This means that SELinux

users are configured independently on each system. While this is acceptable in many situations — SELinux has common defined system users and SELinux-aware services define their own policies — it has some issues when dealing with remote users and systems that access local resources. Remote users and services can get shuffled into a default guest context without a lot of intelligence about what their actual SELinux user and role should be.

This is how Identity Management can cleanly integrate an identity domain with local SELinux services. Identity Management can map IdM users to configured SELinux roles *per host*. Mapping SELinux and IdM users improves user administration:

- ▶ Remote users can be granted appropriate SELinux user contexts based on their IdM group assignments. This also allows administrators to consistently apply the same policies to the same users without having to create local accounts or reconfigure SELinux.
- ▶ SELinux users are automatically updated as hosts are added to the IT environment or as users are added, removed, or changed, without having to edit local systems.
- ▶ SELinux policies can be planned and related to domain-wide security policies through settings like IdM host-based access control rules.
- ▶ Administrators gain environment-wide visibility and control over how users and systems are assigned in SELinux.

SELinux user maps are comprised of three parts: the SELinux user for the system, an IdM user, and an IdM host. These define two separate relationships. First, it defines a map for the SELinux user on a specific host (the local or target system). Second, it defines a map for the SELinux user and the IdM user.

This arrangement allows administrators to set different SELinux users for the same IdM users, depending on which host they are accessing.

SELinux user maps work with the System Security Services Daemon (SSSD) and the **pam_selinux** module. When a remote user attempts to log into a machine, SSSD checks its IdM identity provider to collect the user information, including any SELinux maps. The PAM module then processes the user and assigns it the appropriate SELinux user context.

The core of an SELinux mapping rule is the SELinux system user. Each map is associated with the SELinux user first. The SELinux users which are available for mapping are configured in the IdM server, so there is a central and universal list. These are SELinux users which are configured on every host in the IdM domain. By default, there are five common SELinux users defined:

- ▶ `unconfined_u` (also used as a default for IdM users)
- ▶ `guest_u`
- ▶ `xguest_u`
- ▶ `user_u`
- ▶ `staff_u`

In the IdM server configuration, each SELinux user is configured with both its username and its MLS/MCS range, `SELinux_username:MLS[:MCS]`, and this format is used to identify the SELinux user when configuring maps.

The IdM user and host configuration is very flexible. Users and hosts can be explicitly and individually assigned to an SELinux user map individually, or user groups or host groups can be explicitly assigned to the map.

An extra layer of security is possible by using host-based access control rules. As long as the host-based access control rule defines a user and a host, it can be used for an SELinux user map. Host-

based access control rules (described in [Chapter 21, Policy: Configuring Host-Based Access Control](#)) help integrate SELinux user maps with other access controls in IdM and can help limit or allow host-based user access for remote users, as well as defining local security contexts.



NOTE

If a host-based access control rule is associated with an SELinux user map, the host-based access control rule cannot be deleted until it is removed from the SELinux user map configuration.

22.2. Configuring SELinux Users in IdM

SELinux user maps, as the name implies, creates an association between an SELinux user and an IdM user. Before that association can be established, the IdM server has to be aware of what SELinux users are configured on the systems it manages.

The available SELinux users are part of the IdM server configuration. This is a list, in order from most to least confined, of the SELinux users. The SELinux user entry itself has this format:

```
SELinux_username:MLS[:MCS]
```

The individual user entries are separated with a dollar sign (\$).

Since there is no requirement on user entries to have an SELinux map, many entries may be unmapped. The IdM server configuration can also set a default SELinux user (which is part of the larger SELinux map list) to use for otherwise unmapped IdM user entries.

22.2.1. In the Web UI

1. In the top menu, click the **IPA Server** main tab and the **Configuration** subtab.
2. Scroll to the bottom of the list of server configuration areas, to **SELINUX OPTIONS**.
3. Set the SELinux user configuration.

There are two areas that can be edited: the prioritized list of SELinux users and the default SELinux user to use for unmapped IdM users.

The **SELinux user map order** gives the list of SELinux users, defined on the local Linux system, which are available for configuring mapping rules. This is a prioritized list, from most to least confined. Each SELinux user has the format *SELinux_user:MLS*.

The **Default SELinux user** field sets the SELinux user to use for *unmapped* IdM users.

The screenshot shows the 'IPA Server' configuration page with the 'Configuration' tab selected. Under the 'CONFIGURATION' heading, there are links for 'Refresh', 'Reset', and 'Update'. Below these are expandable sections for 'SEARCH OPTIONS', 'USER OPTIONS', 'GROUP OPTIONS', and 'SELINUX OPTIONS'. The 'SELINUX OPTIONS' section is expanded, showing two input fields: 'SELinux user map order' with the value 'guest_u:s0\$guest_u:s0\$user_u:s0-s0:c0.c' and 'Default SELinux user' with the value 'guest_u:s0'.

4. Click the **Update** link at the top of the page to save the changes.

22.2.2. In the CLI

Before SELinux mapping rules can be created, there has to be a defined and universal list of SELinux users which are available to be mapped. This is set in the IdM server configuration:

```
[jsmith@server ~]$ ipa config-show
...
SELinux user map order: unconfined_u:s0-
s0:c0.c1023$guest_u:s0$guest_u:s0$user_u:s0-s0:c0.c1023$staff_u:s0-s0:c0.c1023
Default SELinux user: unconfined_u:s0-s0:c0.c1023
```

The SELinux user settings can be edited using the **config-mod** command.

Example 22.1. List of SELinux Users

The complete list of SELinux users is passed in the **--ipaselinuusermaporder** option. This list sets a priority order, from most to least confined users.

The SELinux user entry itself has this format:

```
SELinux_user:MLS:MCS
```

The individual user entries are separated with a dollar sign (\$).

For example:

```
[jsmith@server ~]$ ipa config-mod --ipaselinuusermaporder="unconfined_u:s0-  
s0:c0.c1023$guest_u:s0$xguest_u:s0$user_u:s0-s0:c0.c1023$staff_u:s0-s0:c0.c1023"
```



NOTE

The default SELinux user, used for unmapped entries, must be included in the user map list or the edit operation fails. Likewise, if the default is edited, it must be changed to a user in the SELinux map list or the map list must be updated first.

Example 22.2. Default SELinux User

IdM users are not required to have a specific SELinux user mapped to their account. However, the local system still checks the IdM entry for an SELinux user to use for the IdM user account. The default SELinux user sets the fallback user to use for unmapped IdM user entries; this is, by default, the default SELinux user for system users on Red Hat Enterprise Linux, **unconfined_u**.

This default user can be changed with the **--ipaselinuusermapdefault**. For example:

```
[jsmith@server ~]$ ipa config-mod --ipaselinuusermapdefault="guest_u:s0"
```

22.3. Mapping SELinux Users and IdM Users

An SELinux map associates an SELinux user with an IdM user (or users). However, SELinux settings are local to each host system, so a map not only needs to map the SELinux user with an IdM user but also with a host system.

The rule definition primarily identifies the SELinux user; the SELinux user is the basis of the rule.

The other half of the map is comprised of defined IdM users and defined IdM hosts. (There can be one single user or host or multiple users and hosts or user and host groups in the map.) The users and hosts can be defined either by explicitly listing users and hosts or by referencing a host-based access control rule.

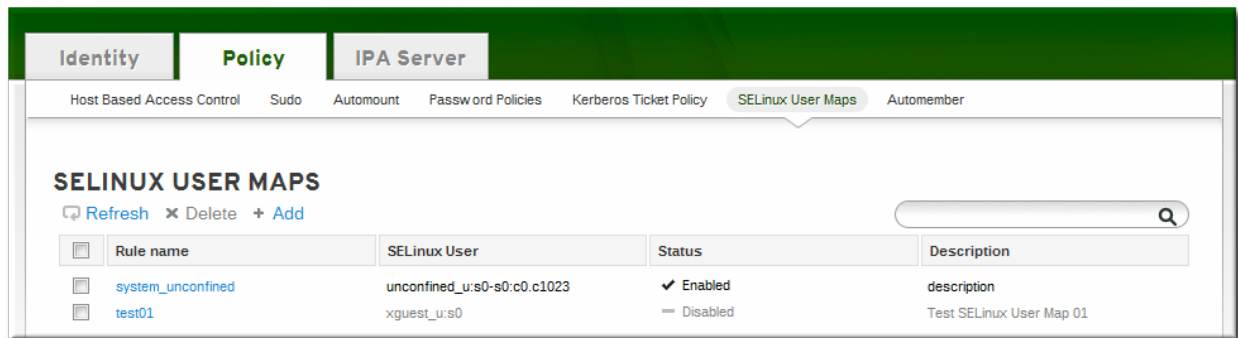


NOTE

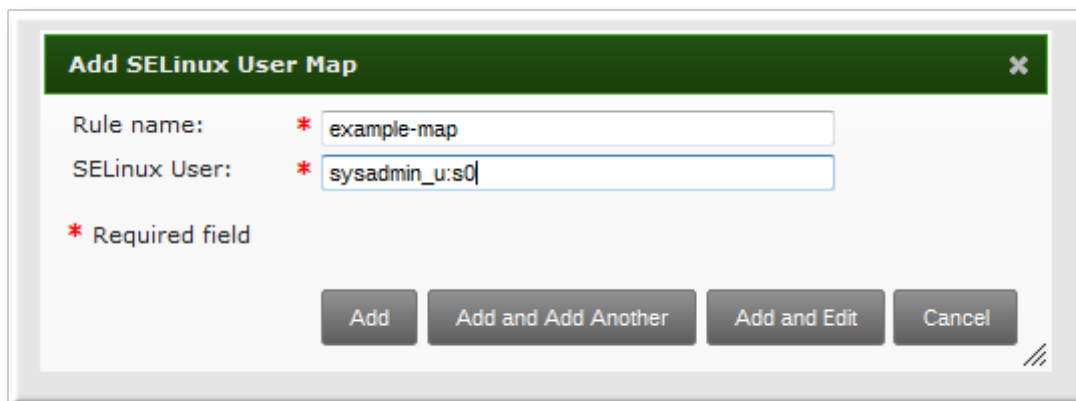
The host-based access control rule must contain users and hosts, not just services.

22.3.1. In the Web UI

1. In the top menu, click the **Policy** main tab and the **SELinux User Mappings** subtab.
2. In the list of mappings, click the **Add** button to create a new map.



3. Enter the name for the map and the SELinux user *exactly as it appears in the IdM server configuration*. SELinux users have the format `SELinux_username:MLS[:MCS]`.



4. Click **Add and Edit** to add the IdM user information.
5. As described in the introduction, an SELinux map has three parts: the SELinux user and an IdM user/host pairing. That IdM user/host pair can be defined in one of two ways: it can be set for explicit users on explicit hosts, or it can be defined using a host-based access control rule. To set a host-based access control rule, select the rule from the drop-down menu in the **General** area of the configuration. Using a host-based access control rule also introduces access controls on what hosts a remote user can use to access a target machine. **Only one host-based access control rule can be set.**

SELinux User Maps » user

SELINUX USER MAP: user

Settings

Refresh Reset Update

GENERAL

Rule name: **user**

Description:

SELinux User: *

HBAC Rule: undo

Status: ☒ Enabled ☐ Disabled

Alternatively, scroll down the **Users** and **Hosts** areas, and click the **Add** link to assign users, user groups, hosts, or host groups to the SELinux map.

Add Users into SELinux User Map user

Find

Available	Prospective
☐ Users ☐ bguster ☐ bjensen ☐ jrockford ☐ ljones ☒ mreynolds ☒ sspencer	☐ Users

>> <<

Add Cancel

Select the users (or hosts or groups) on the left, click the right arrows button (>>) to move them to

the **Prospective** column, and click the **Add** button to add them to the rule.

SELinux User Maps » user_u

SELINUX USER MAP: user_u

Settings

Refresh Reset Update

Status: ☒ Enabled ☐ Disabled

▼ USER

User category the rule applies to: ☐ Anyone ☒ Specified Users and Groups

☐ Users	✕ Delete	+ Add
☐ jsmith		
☐ User Groups	✕ Delete	+ Add

▼ HOST

Host category the rule applies to: ☐ Any Host ☒ Specified Hosts and Groups

☐ Hosts	✕ Delete	+ Add
☐ test.example.com		
☐ Host Groups	✕ Delete	+ Add



NOTE

Either a host-based access control rule can be given or the users and hosts can be set manually. Both options cannot be used at the same time.

6. Click the **Update** link at the top to save the changes to the SELinux user map.

22.3.2. In the CLI

An SELinux map rule has three fundamental parts:

- » The SELinux user (**--selinuxuser**)
- » The user or user groups which are associated with the SELinux user (**--users** or **--groups**)
- » The host or host groups which are associated with the SELinux user (**--hosts** or **--hostgroups**)
- » Alternatively, a host-based access control rule which specifies both hosts and users in it (**--hbacrule**)

A rule can be created with all information at once using the **selinuxusermap-add** command. Users and hosts can be added to a rule after it is created by using the **selinuxusermap-add-user** and **selinuxusermap-add-host** commands, respectively.

Example 22.3. Creating a New SELinux Map

The **--selinuxuser** value must be the SELinux user name exactly as it appears in the IdM server configuration. SELinux users have the format *SELinux_username:MLS[:MCS]*.

Both a user and a host (or appropriate groups) must be specified for the SELinux mapping to be valid. Users, hosts, or groups can be specified in comma-separated lists.

```
[jsmith@server ~]$ ipa selinuxusermap-add --users=jsmith,bjensen,jrockford --
hosts=server.example.com,test.example.com --selinuxuser="xguest_u:s0" selinux1
```

Example 22.4. Creating an SELinux Map with a Host-Based Access Control Rule

The **--hbacrule** value identifies the host-based access control rule to use for mapping. Using a host-based access control rule introduces access controls on what hosts a remote user can use to access a target machine, along with applying SELinux contexts after the remote user as logged into the target machine.

The access control rule must specify both users and hosts appropriately so that the SELinux map can construct the SELinux user, IdM user, and host triple.

Only one host-based access control rule can be specified.

```
[jsmith@server ~]$ ipa selinuxusermap-add --hbacrule=webserver --
selinuxuser="xguest_u:s0" selinux1
```

Host-based access control rules are described in [Chapter 21, Policy: Configuring Host-Based Access Control](#).

Example 22.5. Adding a User to an SELinux Map

While all of the users and hosts can be added to a map when it is created, users and hosts can also be added after the rule is created. This is done using a specific command, either **selinuxusermap-add-user** or **selinuxusermap-add-host**.

```
[jsmith@server ~]$ ipa selinuxusermap-add-user --users=jsmith selinux1
```

It is not necessary to use a separate command to add a host-based access control rule after the rule is configured because there can only be one. If the **selinuxusermap-mod** command is used with the **--hbacrule** option, it adds the host-based access control rule or overwrites the previous one.

A specific user or host can be removed from an SELinux map by using either the **selinuxusermap-remove-host** or **selinuxusermap-remove-user** command.

Example 22.6. Removing a User from an SELinux Map

As with adding a user to a `ion>` value identifies the host-based access control rule to use for mapping. The access control rule must specify both users and hosts appropriately so that the SELinux map can construct the SELinux user, IdM user, and host triple.

```
[jsmith@server ~]$ ipa selinuxusermap-remove-user --users=jsmith selinux1
```


Chapter 23. Policy: Defining Automatic Group Membership for Users and Hosts

Most of the policies and configuration within the Identity Management domain are based on *groups*. Settings from sudo rules to automount to access control are defined for groups, and then those settings are applied to group members.

Managing group membership is an important factor in managing users and hosts. Creating *automember groups* defines rules to add users and hosts to specified groups automatically, as soon as a new entry is added.

23.1. About Automembership

One of the most critical tasks for managing policies, identities, and security is managing group membership in Identity Management. Groups are the core of most policy configuration.

By default, hosts do not belong to any group when they are created; users are added to the catchall **ipausers** group. Even if custom groups are configured and all policy configuration is in place, users and hosts cannot take advantage of those policies until they are joined to groups. Of course, this can be done manually, but it is both more efficient and more consistent if group membership can be assigned automatically.

This is done with *automembership groups*.

Automembership is essentially an automatic, global entry filter that organizes entries, at least in part, based on specific criteria. An automember rule, then, is the way that that filter is specified.

For example, there can be a lot of different, repeatable ways that to categorize identities within the IT and organizational environment:

- ▶ Adding all hosts or all users to a single global group.
- ▶ Adding employees to specific groups based on their employee type, ID number, manager, or physical location.
- ▶ Dividing hosts based on their IP address or subnet.

Automembers provide a way to pre-sort those entries. That makes it easier to configure the actual behavior that you want to configure — like granting different sudo rules to different user types or machines on different subnets or have different automount settings for different users.



NOTE

Automembership only applies to *new* users or groups. Changing the configuration on an existing user or group does not affect group membership, either by adding or removing the user/host in the group.

Automembership is a flag or a target set on an existing user group or host group. An *automembership rule* is created as a policy. This is a sister entry to the actual group entry and it signals that the given group is used for automatic group membership.

Once the rule is created — once the group is identified as being a target — then the next step is to define *automember conditions*. Conditions are regular expression filters that are used to identify group members. Conditions can be inclusive or exclusive, meaning that matching entries can be added or

ignored based on those conditions.

There can be multiple conditions in a single rule. A user or host entry can match multiple rules and be added to multiple groups.

Automembership is a way of imposing reliable order on user and host group entries as they are created.

The key to using automember groups effectively is to plan your overall Identity Management structure — the access control policies, sudo rules, host/service management rules, host groups, and user groups.

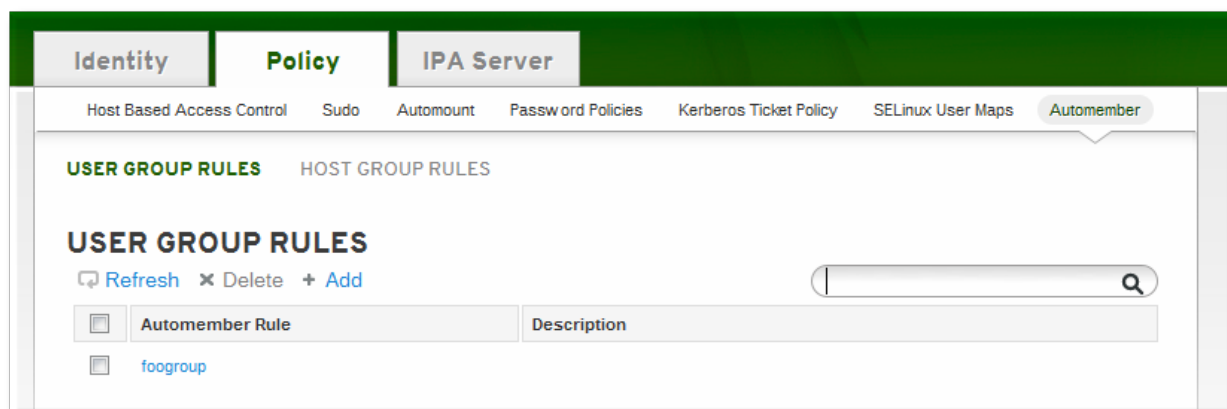
Once the structure is in place, then several things are clear:

- What groups will be used in the Identity Management
- What specific groups different types of users and hosts need to belong to to perform their designated functions
- What delineating attributes can be used to filter users and hosts into the appropriate groups

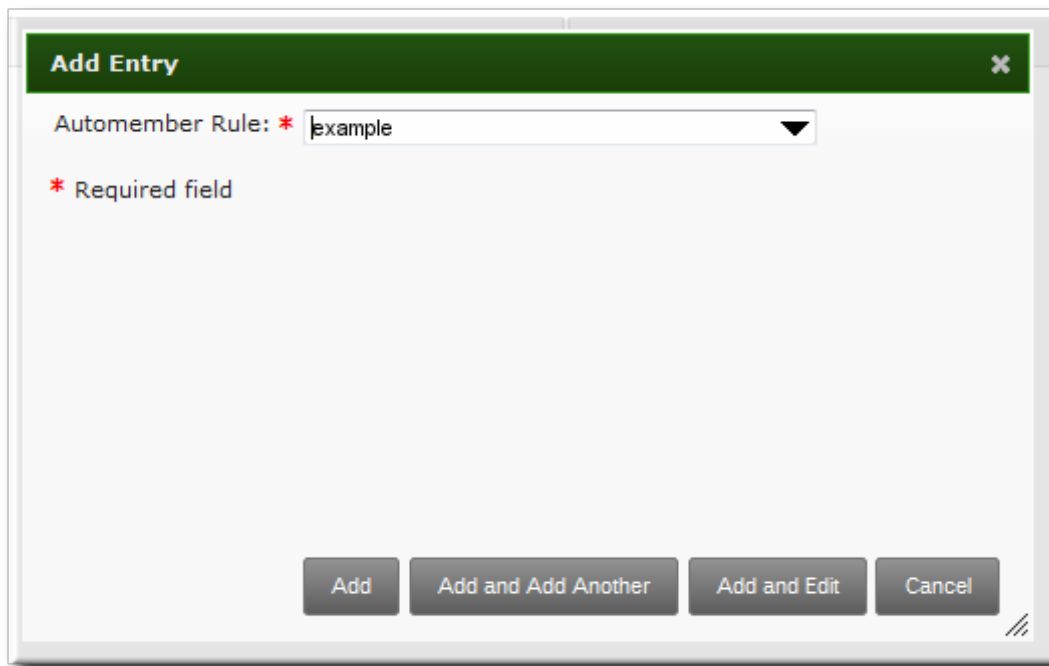
23.2. Defining Automembership Rules (Basic Procedure)

23.2.1. From the Web UI

1. Create the user group ([Section 9.10.1, “Creating User Groups”](#)) or host group ([Section 10.6.1.1, “Creating Host Groups from the Web UI”](#)).
2. Open the **Policy** tab, and select the **Automembers** subtab.
3. In the top of the **Automembers** area, select the type of autogroup to create, either **USER GROUP RULES** or **HOST GROUP RULES**.



4. In the drop-down menu, select the group for which to create the automember rule.



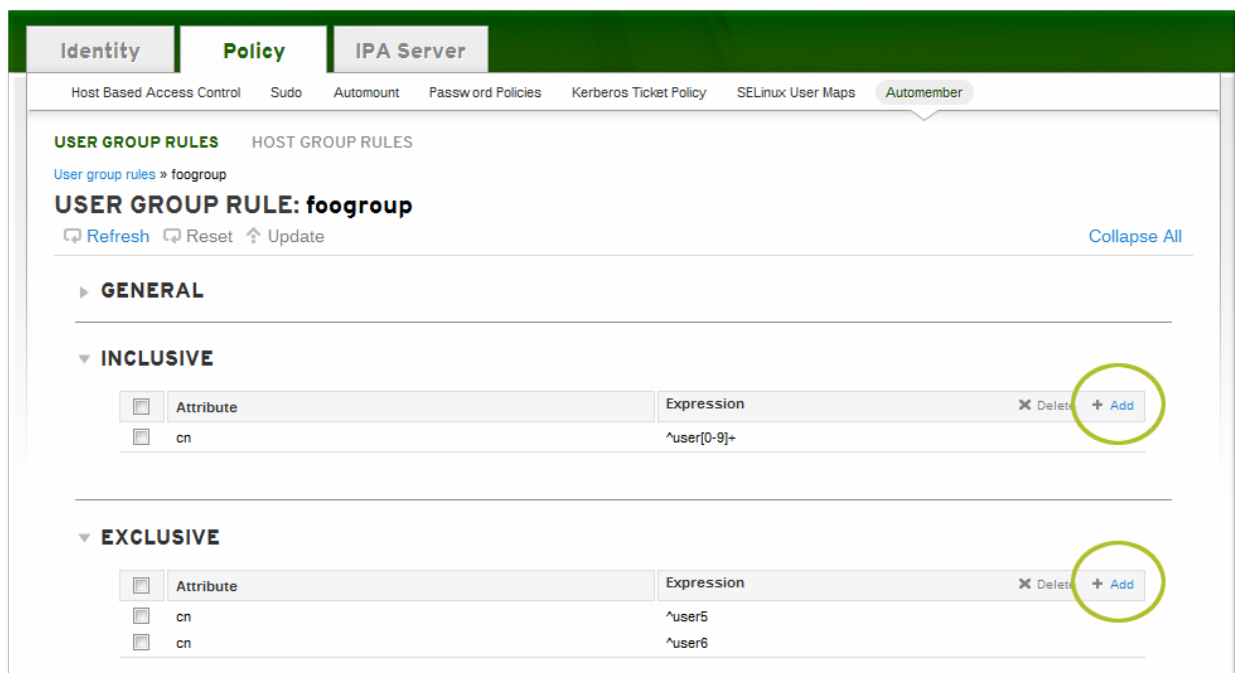
Add Entry

Automember Rule: * example

* Required field

Add Add and Add Another Add and Edit Cancel

5. Click the **Add and Edit** button.
6. In the edit page for the rule, click the + **Add** by the type of condition to create to identify entries.



Identity **Policy** **IPA Server**

Host Based Access Control Sudo Automount Password Policies Kerberos Ticket Policy SELinux User Maps Automember

USER GROUP RULES HOST GROUP RULES

User group rules » foogroup

USER GROUP RULE: foogroup

Refresh Reset Update Collapse All

GENERAL

INCLUSIVE

Attribute	Expression	Delete	+ Add
cn	^user[0-9]+		

EXCLUSIVE

Attribute	Expression	Delete	+ Add
cn	^user5		
cn	^user6		

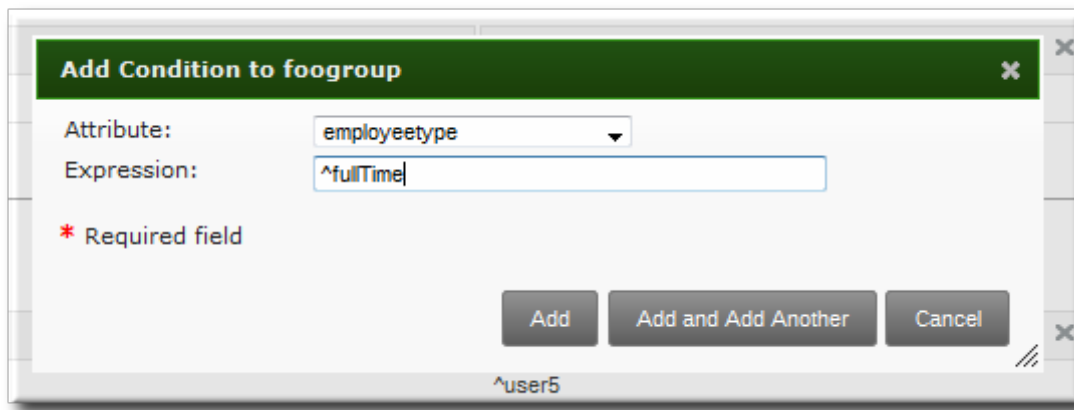
7. Select the attribute to use as the basis for the search and then set the regular expression to use to match the attribute value.

Conditions can look for entries either to *include* in the group or to explicitly *exclude* from the group. The format of a condition is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see [the `pcresyntax\(3\)` man page](#).



NOTE

Exclude conditions are evaluated first and take precedence over include conditions.



8. Click **Add and Add Another** to add another condition. A single rule can have multiple include and exclude conditions. When all conditions have been configured, click the **Add** button to save the last condition and close the dialog window.

23.2.2. From the CLI

There are two commands used to define an automember policy:

- A command to flag the group as an automember group, **automember-add**
- A command to add regular expression conditions to identify group members, **automember-add-condition**

For example:

1. Create the user group ([Section 9.10.1.2, “With the Command Line”](#)) or host group ([Section 10.6.1.2, “Creating Host Groups from the Command Line”](#)).
2. Create the automember policy entry for the group. Use the **--type** to identify whether the target group is a user group (**group**) or a host group (**hostgroup**). This command has the format:

```
ipa automember-add --type=group|hostgroup groupName
```

For example:

```
[jsmith@server ~]$ ipa automember-add --type=group exampleGroup
```

3. Create the conditions for the rule. To set multiple patterns, either give a comma-separated list of patterns in the **--inclusive-regex|--exclusive-regex** options or run the command multiple times.

This command has the format:

```
ipa automember-add-condition --type=group|hostgroup --key=attribute --  
inclusive-regex=regex | --exclusive-regex=regex groupName
```

As with the automember rule, the condition must specify the type of group (**--type**) and the name of the target group (*groupName*).

The condition must also specify the attribute (the key) and any patterns for the attribute value. The **--key** is the attribute name that is the focus of the condition. Then, there is a regular expression pattern to identify matching values; matching entries can either be included (**--inclusive-regex**) or excluded (**--exclusive-regex**) from the group. Exclusion rules take precedence.

For example, to include all employees with Barbara Jensen as a manager, but excluding the

temporary employees:

```
[jsmith@server ~]$ ipa automember-add-condition --type=group --key=manager -
-inclusive-regex=^uid=bjensen$ exampleGroup
[jsmith@server ~]$ ipa automember-add-condition --type=group --
key=employee --exclusive-regex=^temp exampleGroup
```



TIP

The regular expression can match any part of the string. Using a caret (^) means that it must match at the beginning, and using a dollar sign (\$) means that it must match at the end. Wrapping the pattern in ^ and \$ means that the string as a whole must match.

For more information on Perl-compatible regular expression (PCRE) patterns, see [the pcreyntax\(3\) man page](#).

To remove a condition for a rule, pass the full condition information, both the key and the regular expression:

```
[jsmith@server ~]$ ipa automember-remove-condition --key=fqdn --type=hostgroup --
inclusive-regex=^web[1-9]+\..example\.com webserver
```

To remove the entire rule, simply run the **automember-del** command.

23.3. Examples of Using Automember Groups



NOTE

These examples are shown using the CLI; the same configuration can be performed in the web UI.

A Note on Creating Default Groups

One common environment requirement is to have some sort of default group that users or hosts are added to. There are a couple of different ways to approach that.

- All entries can be added to a single, global group regardless of what other groups they are also added to.
- Entries can be added to specific automember groups. If the new entry does not match any autogroup, then it is added to a default or fallback group.

These strategies are mutually exclusive. If an entry matches a global group, then it does match an automember group and would, therefore, not be added to the fallback group.

23.3.1. Setting an All Users/Hosts Rule

To add all users or all hosts to a single group, use an inclusive regular expression for some attribute (such as **cn** or **fqdn**) which all entries will contain.

A regular expression to match all entries is simply **.***. For example, to add all hosts to the same host group:

```
[jsmith@server ~]$ ipa automember-add-condition --type=hostgroup allhosts --
inclusive-regex=.* --key=fqdn
-----
Added condition(s) to "allhosts"
-----
    Automember Rule: allhosts
    Inclusive Regex: fqdn=.*
-----
Number of conditions added 1
-----
```

Every host added after that is automatically added to the **allhosts** group:

```
[jsmith@server ~]$ ipa host-add test.example.com
-----
Added host "test.example.com"
-----
    Host name: test.example.com
    Principal name: host/test.example.com@EXAMPLE.COM
    Password: False
    Keytab: False
    Managed by: test.example.com

[jsmith@server ~]$ ipa hostgroup-show allhosts
    Host-group: allhosts
    Description: Default hostgroup
    Member hosts: test.example.com
```

For more information on PCRE patterns, see [the pcreyntax\(3\) man page](#).

23.3.2. Defining Default Automembership Groups

There is a special command to set a default group, **automember-default-group-set**. This sets the group name (**--default-group**) and group type (**--type**), similar to an automember rule, but there is no condition to match. By definition, default group members are unmatched entries.

For example:

```
[jsmith@server ~]$ ipa automember-default-group-set --default-group=ipaclients --
type=hostgroup
[jsmith@server ~]$ ipa automember-default-group-set --default-group=ipausers --
type=group
```

A default group rule can be removed using the **automember-default-group-remove** command. Since there is only one default group for a group type, it is only necessary to give the group type, not the group name:

```
[jsmith@server ~]$ ipa automember-default-group-remove --type=hostgroup
```

23.3.3. Using Automembership Groups with Windows Users

When a user is created in IdM, that user is automatically added as a member to the **ipausers** group (which is the default group for all new users, apart from any automember group). However, when a Windows user is synced over from Active Directory, that user is not automatically added to the **ipausers** group.

New Windows users can be added to the **ipausers** group, as with users created in Identity Management, by using an automember group. Every Windows user is added with the **ntUser** object class; that object class can be used as an inclusive filter to identify new Windows users to add to the automember group.

First, define the **ipausers** group as an automember group:

```
[jsmith@server ~]$ ipa automember-add --type=group ipausers
```

Then, use the **ntUser** object class as a condition to add users:

```
[jsmith@server ~]$ ipa automember-add-condition ipausers --key=objectclass --  
type=group --inclusive-regex=ntUser
```

Chapter 24. Configuration: Defining Access Control for IdM Users

Access control is a security system which defines who can access certain resources — from machines to services to entries — and what kinds of operations they are allowed to perform. Identity Management provides several access control areas to make it very clear what kind of access is being granted and to whom it is granted. As part of this, Identity Management draws a distinction between access controls to resources within the domain and access control to the IdM configuration itself.

This chapter details the different internal access control mechanisms that are available for users within IdM to the IdM server and other IdM users.

24.1. About Access Controls for IdM Entries

Access control defines the rights or permissions users have been granted to perform operations on other users or objects.

24.1.1. A Brief Look at Access Control Concepts

The Identity Management access control structure is based on standard LDAP access controls. Access within the IdM server is based on the IdM users (who are stored in the backend Directory Server instance) who are allowed to access other IdM entities (which are also stored as LDAP entries in the Directory Server instance).

An access control rule has three parts:

- ▶ *Who can perform the operation.* This is the entity who is being granted permission to do something; this is the actor. In LDAP access control models, this is called the *bind rule* because it defines who the user is (based on their bind information) and can optionally require other limits on the bind attempt, such as restricting attempts to a certain time of day or a certain machine.
- ▶ *What can be accessed.* This defines the entry which the actor is allowed to perform operations on. This is the *target* of the access control rule.
- ▶ *What type of operation can be performed.* The last part is determining what kinds of actions the user is allowed to perform. The most common operations are add, delete, write, read, and search. In Identity Management, all users are implicitly granted read and search rights to all entries in the IdM domain, with restrictions only for sensitive attributes like passwords and Kerberos keys. (Anonymous users are restricted from seeing security-related configuration, like **sudo** rules and host-based access control.)

The only rights which can be granted are add, delete, and write — the permissions required to *modify* an entry.

**NOTE**

Identity Management does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In IdM, read and search access are implicitly granted to any authenticated user.

Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

When any operation is attempted, the first thing that the IdM client does is send user credentials as part of the bind operation. The backend Directory Server checks those user credentials and then checks the user account to see if the user has permission to perform the requested operation.

24.1.2. Access Control Methods in Identity Management

To make access control rules simple and clear to implement, Identity Management divides access control definitions into three categories:

- ▶ *Self-service rules*, which define what operations a user can perform on his own personal entry. The access control type only allows write permissions to attributes within the entry; it does not allow add or delete operations for the entry itself.
- ▶ *Delegation rules*, which allow a specific user group to perform write (edit) operations on specific attributes for users in another user group. Like self-service rules, this form of access control rule is limited to editing the values of specific attributes; it does not grant the ability to add or remove whole entries or control over unspecified attributes.
- ▶ *Role-based access control*, which creates special access control groups which are then granted much broader authority over all types of entities in the IdM domain. Roles can be granted edit, add, and delete rights, meaning they can be granted complete control over entire entries, not just selected attributes.

Some roles are already created and available within Identity Management. Special roles can be created to manage any type of entry in specific ways, such as hosts, automount configuration, netgroups, DNS settings, and IdM configuration.

24.2. Defining Self-Service Settings

Self-service access control rules define the operations that an entity can perform on itself. These rules define only what attributes a user (or other IdM entity) can edit on their personal entries.

Two self-service rules exist by default:

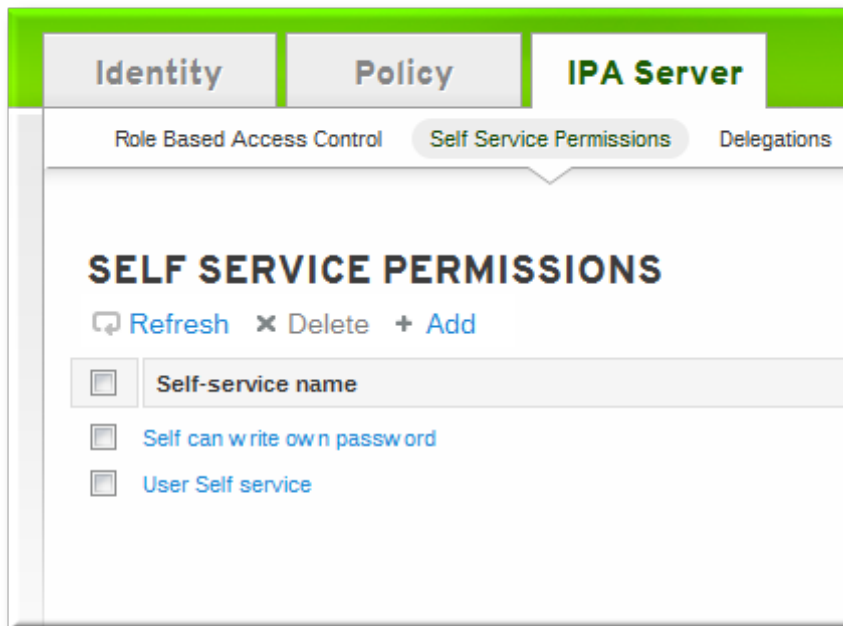
- ▶ A rule for editing some general attributes in the personal entry, including given name and surname, phone numbers, and addresses.
- ▶ A rule to edit user passwords, including two Samba passwords, the Kerberos password, and the general user password.

24.2.1. Creating Self-Service Rules from the Web UI

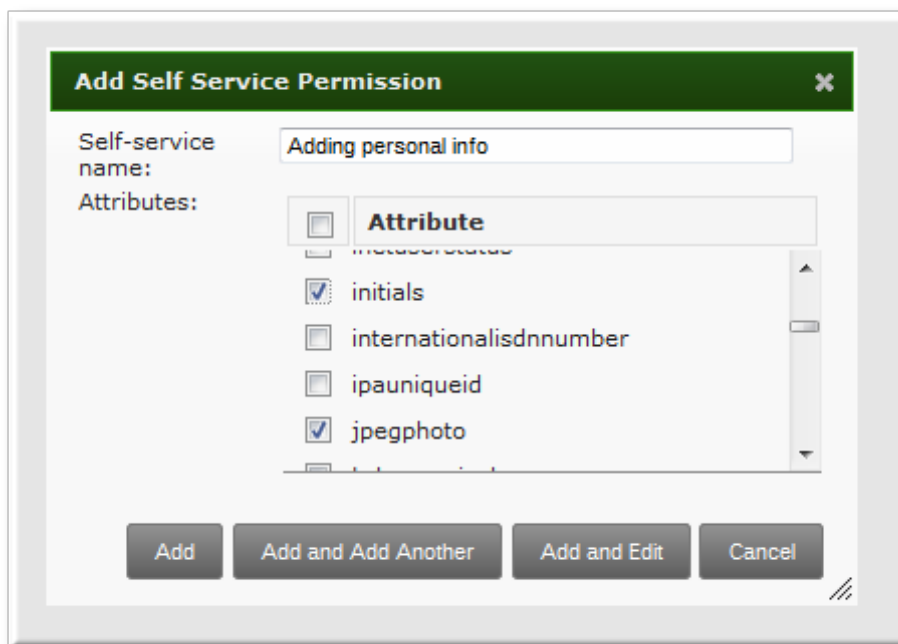
1. Open the **IPA Server** tab in the top menu, and select the **Self Service Permissions**

subtab.

- Click the **Add** link at the top of the list of self-service ACIs.



- Enter the name of the rule in the pop-up window. Spaces are allowed.



- Select the checkboxes by the attributes which this ACI will permit users to edit.
- Click the **Add** button to save the new self-service ACI.

24.2.2. Creating Self-Service Rules from the Command Line

A new self-service rule can be added using the **selfservice-add** command. There are two required options, **--permissions** to set whether the ACI grants write, add, or delete permission and **--attrs** to give the full list of attributes which this ACI grants permission to.

```
$ ipa selfservice-add "Users can manage their own name details" --
permissions=write --attrs=givenname,displayname,title,initials
-----
Added selfservice "Users can manage their own name details"
-----
Self-service name: Users can manage their own name details
Permissions: write
Attributes: givenname, displayname, title, initials
```

24.2.3. Editing Self-Service Rules

In the self-service entry in the web UI, the only element that can be edited is the list of attributes that are included in the ACL. The checkboxes can be selected or deselected.

Self Service Permissions » User Self service

SELF SERVICE PERMISSION: User Self service

Settings

Reset Update

GENERAL

Self-service name: User Self service

Attributes:

Attribute	Selected
Attribute	☐
audio	☐
businesscategory	☒
carlicense	☒
cn	☒
departmentnumber	☐

Figure 24.1. Self-Service Edit Page

With the command line, self-service rules are edited using the **ipa selfservice-mod** command. The **--attrs** option overwrites whatever the previous list of supported attributes was, so always include the complete list of attributes along with any new attributes.

```
$ ipa selfservice-mod "Users can manage their own name details" --
attrs=givenname,displayname,title,initials,surname
-----
Modified selfservice "Users can manage their own name details"
-----
Self-service name: Users can manage their own name details
Permissions: write
Attributes: givenname, displayname, title, initials
```



IMPORTANT

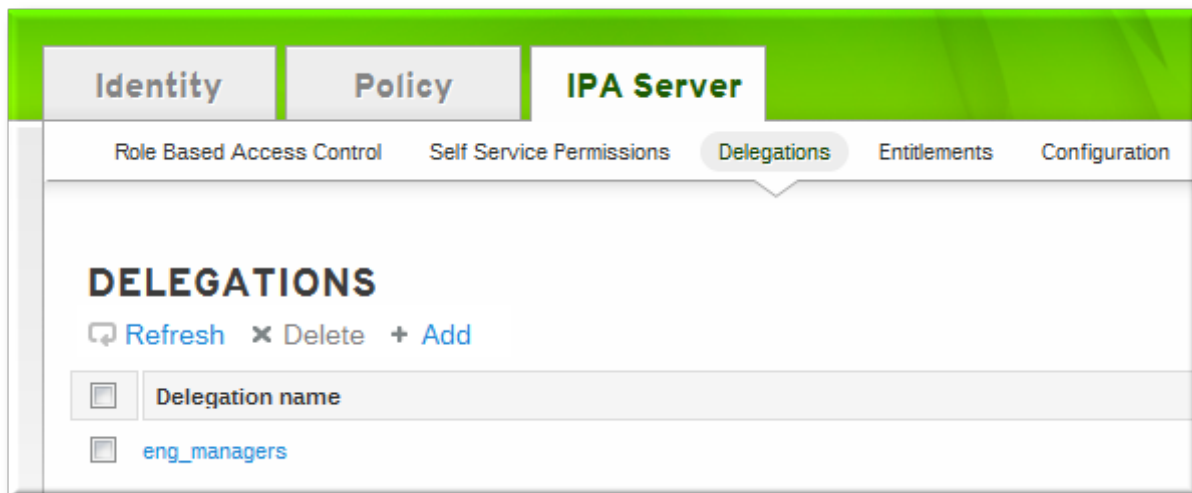
Include all of the attributes when modifying a self-service rule, including existing ones.

24.3. Delegating Permissions over Users

Delegation is very similar to roles in that one group of users is assigned permission to manage the entries for another group of users. However, the delegated authority is much more similar to self-service rules in that complete access is granted but only to specific user attributes, not to the entire entry. Also, the groups in delegated authority are existing IdM user groups instead of roles specifically created for access controls.

24.3.1. Delegating Access to User Groups in the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Delegations** subtab.
2. Click the **Add** link at the top of the list of delegation ACIs.



3. Name the new delegation ACI.
4. In the **User group** drop-down menu, select the group *whose entries can be edited* by members of the delegation group.

5. In the **Member user group** drop-down menu, select the group *who is being granted permissions* to the entries of users in the user group.
6. In the attributes box, select the checkboxes by the attributes to which the member user group is being granted permission.
7. Click the **Add** button to save the new delegation ACI.

24.3.2. Delegating Access to User Groups in the Command Line

A new delegation access control rule is added using the **delegation-add** command. There are three required arguments:

- **--group**, the group *who is being granted permissions* to the entries of users in the user group.
- **--membergroup**, the group *whose entries can be edited* by members of the delegation group.
- **--attrs**, the attributes which users in the member group are allowed to edit.

For example:

```
$ ipa delegation-add "basic manager attrs" --
  attrs=manager,title,employee,employeenumber --group=engineering_managers --
  membergroup=engineering
-----
Added delegation "basic manager attrs"
-----
  Delegation name: basic manager attrs
  Permissions: write
  Attributes: manager, title, employee, employeenumber
  Member user group: engineering
  User group: engineering_managers
```

Delegation rules are edited using the **delegation-mod** command. The **--attrs** option overwrites whatever the previous list of supported attributes was, so always include the complete list of attributes along with any new attributes.

```
$ ipa delegation-mod "basic manager attrs" --
attrs=manager,title,employeetype,employeenumber,displayname
-----
Modified delegation "basic manager attrs"
-----
Delegation name: basic manager attrs
Permissions: write
Attributes: manager, title, employeetype, employeenumber, displayname
Member user group: engineering
User group: engineering_managers
```



IMPORTANT

Include all of the attributes when modifying a delegation rule, including existing ones.

24.4. Defining Role-Based Access Controls

Role-based access control grants a very different kind of authority to users compared to self-service and delegation access controls. Role-based access controls are fundamentally administrative, with the potential to add, delete, and significantly modify entries.

There are three parts to role-based access controls:

- The *permission*. The permission defines a specific operation or set of operations (write, add, or delete) and the target entries within the IdM LDAP directory to which those operations apply. Permissions are building blocks; they can be assigned to multiple privileges as needed.
- The *privileges* available to a role. A privilege is essentially a group of permissions. Permissions are not applied directly to a role. Permissions are added to a privilege so that the privilege creates a coherent and complete picture of a set of access control rules. For example, a permission can be created to add, edit, and delete automount locations. Then that permission can be combined with another permission relating to managing FTP services, and they can be used to create a single privilege that relates to managing filesystems.
- The *role*. This is the list of IdM users who are able to perform the actions defined in the privileges.

It is possible to create entirely new permissions, as well as to create new privileges based on existing permissions or new permissions. A list of the default privileges and their associated permissions are in [Table 24.1, “Privileges and Permissions in IdM”](#).



NOTE

Identity Management does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In IdM, read and search access are implicitly granted to any authenticated user.

Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

Table 24.1. Privileges and Permissions in IdM

Privilege	Associated Permissions
Automount Administrators	Add_Automount_maps Remove_Automount_maps Add_Automount_keys Remove_Automount_keys
Certificate Administrators	Retrieve_Certificates_from_the_CA Request_Certificate Request_Certificates_from_a_different_hos Get_Certificates_status_from_the_CA Revoke_Certificate Certificate_Remove_Hold
Delegation Administrator	Add_Roles Remove_Roles Modify_Roles Modify_Role_membership Modify_privilege_membership
DNS Administrators (for users)	add_dns_entries remove_dns_entries update_dns_entries
DNS Servers (for machines)	add_dns_entries remove_dns_entries update_dns_entries
Group Administrators	Add_Groups Remove_Groups Modify_Groups Modify_Group_membership
HBAC Administrator	Add_HBAC_rule

	Delete_HBAC_rule Modify_HBAC_rule Manage_HBAC_rule_membership Add_HBAC_services Delete_HBAC_services Add_HBAC_service_groups Delete_HBAC_service_groups Manage_HBAC_service_group_membership
Host Administrators	Add_Hosts Remove_Hosts Modify_Hosts Manage_host_keytab Enroll_a_host Add_krbPrincipalName_to_a_host
Host Enrollment	Manage_host_keytab Enroll_a_host Add_krbPrincipalName_to_a_host
Host Group Administrators	Add_Hostgroups Remove_Hostgroups Modify_Hostgroups Modify_Hostgroup_membership
Modify Users and Reset Passwords	Modify_Users
Netgroups Administrators	Add_netgroups Remove_netgroups Modify_netgroups Modify_netgroup_membership
Password Policy Administrator	Add_Group_Password_Policy_costemplate Delete_Group_Password_Policy_costemplate

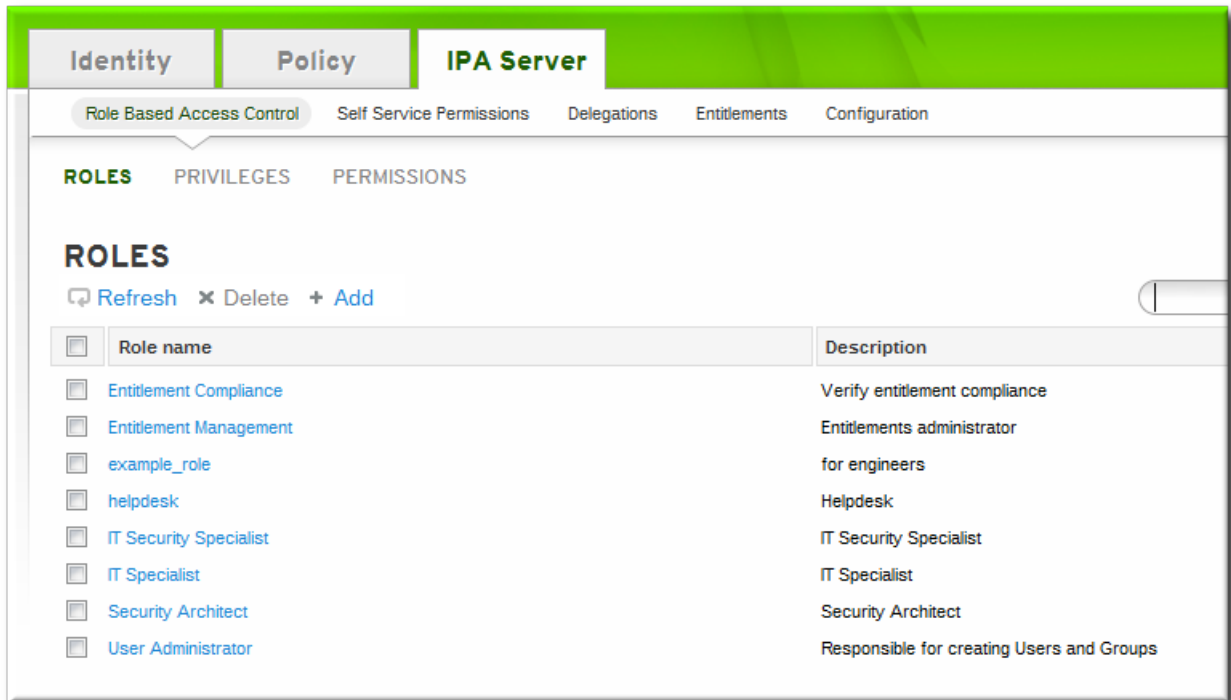
	Modify_Group_Password_Policy_costemplate Add_Group_Password_Policy Delete_Group_Password_Policy Modify_Group_Password_Policy
Replication Administrators ^[a]	Add_Replication_Agreements Remove_Replication_Agreements Modify_Replication_Agreements
Service Administrators	Add_Services Remove_Services Modify_Services Manage_service_keytab
Sudo Administrator	Add_Sudo_rule Delete_Sudo_rule Modify_Sudo_rule Add_Sudo_command Delete_Sudo_command Modify_Sudo_command Add_Sudo_command_group Delete_Sudo_command_group Manage_Sudo_command_group_membership
User Administrators	Change_a_user_password Add_user_to_default_group Unlock_user_accounts Remove_Users Modify_Users Add_Users
Write IPA Configuration	Write_IPA_Configuration

^[a] This permission can only be granted to servers, not to users.

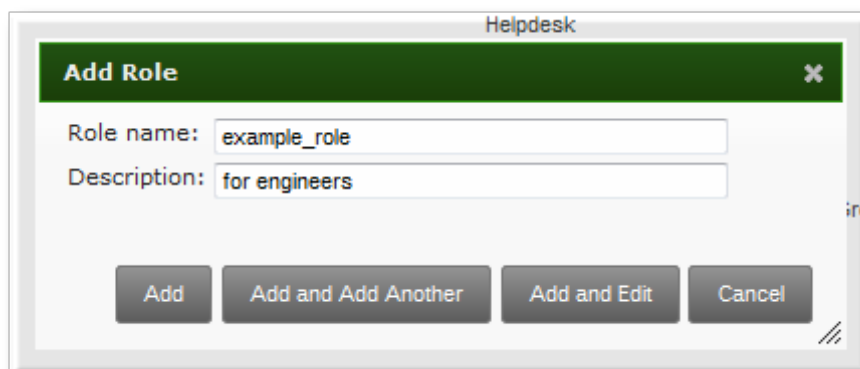
24.4.1. Creating Roles

24.4.1.1. Creating Roles in the Web UI

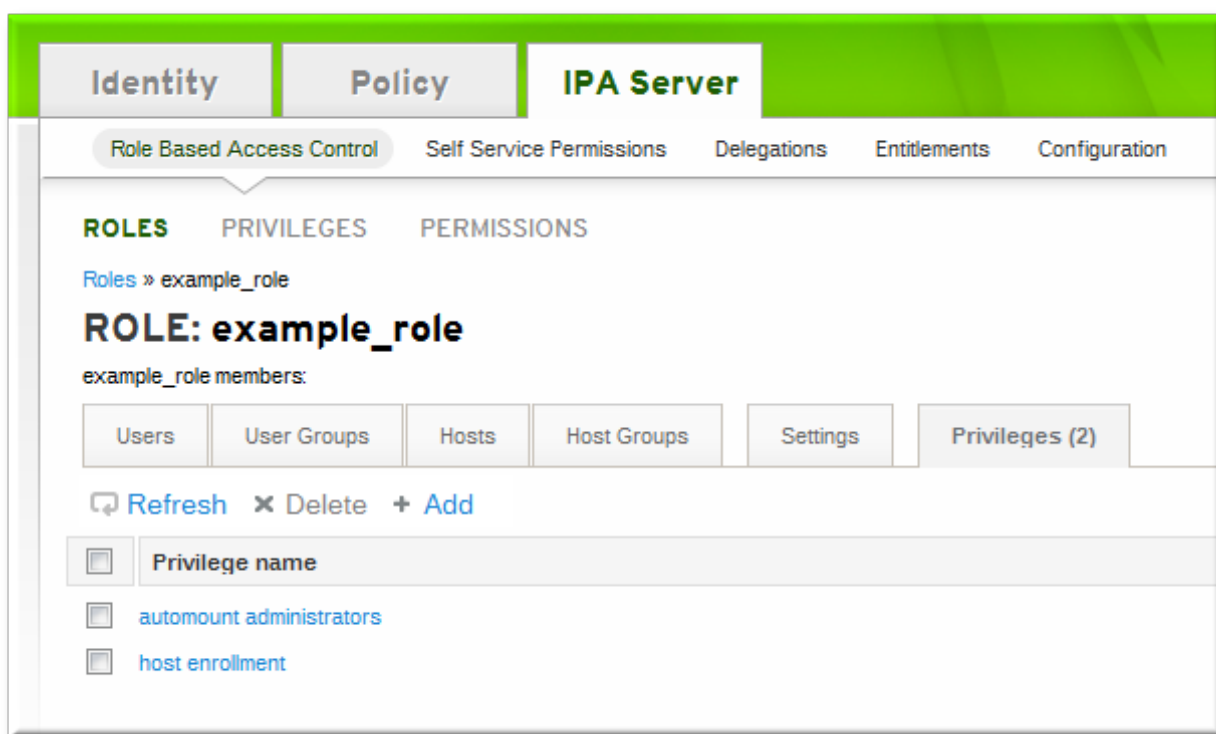
1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Click the **Add** link at the top of the list of role-based ACIs.



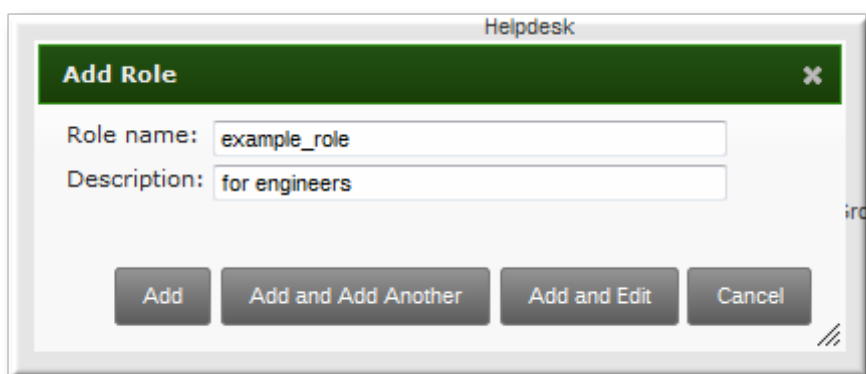
3. Enter the role name and a description.



4. Click the **Add and Edit** button to save the new role and go to the configuration page.
5. Open the **Privileges** tab in the role configuration page.
6. Click the **Add** link at the top of the list of privileges to add a new privilege.



7. Enter the role name and a description.



24.4.1.2. Creating Roles in the Command Line

1. Add the new role:

```
# ipa role-add --desc="User Administrator" useradmin
-----
Added role "useradmin"
-----
Role name: useradmin
Description: User Administrator
```

2. Add the required privileges to the role:

```
# ipa role-add-privilege --privileges="User Administrators" useradmin
Role name: useradmin
Description: User Administrator
Privileges: user administrators
-----
Number of privileges added 1
-----
```

3. Add the required groups to the role. In this case, we are adding only a single group, **useradmin**, which already exists.

```
# ipa role-add-member --groups=useradmins useradmin
Role name: useradmin
Description: User Administrator
Member groups: useradmins
Privileges: user administrators
-----
Number of members added 1
-----
```

24.4.2. Creating New Permissions



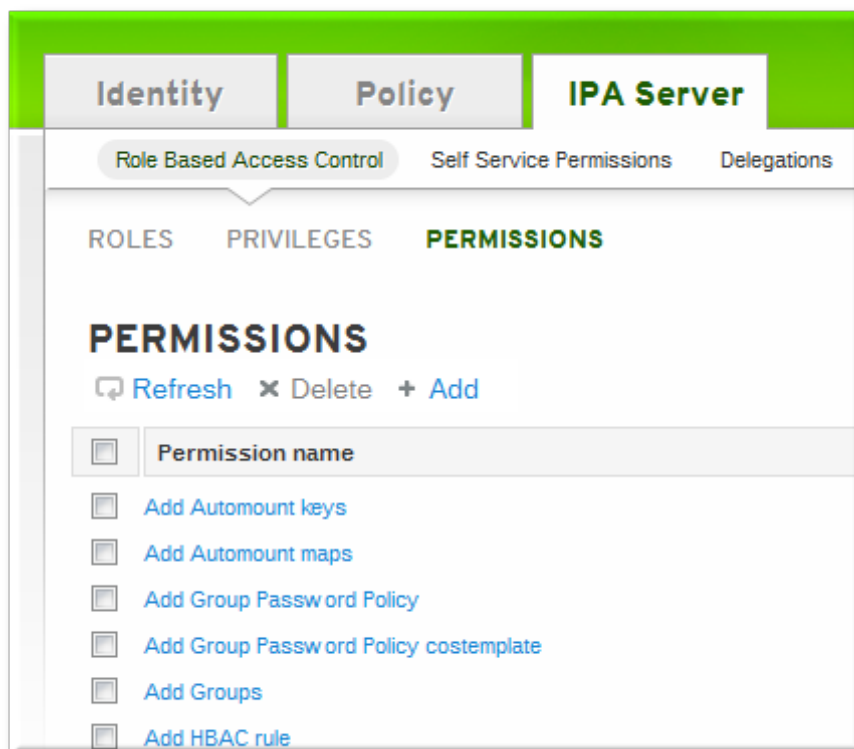
NOTE

Identity Management does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In IdM, read and search access are implicitly granted to any authenticated user.

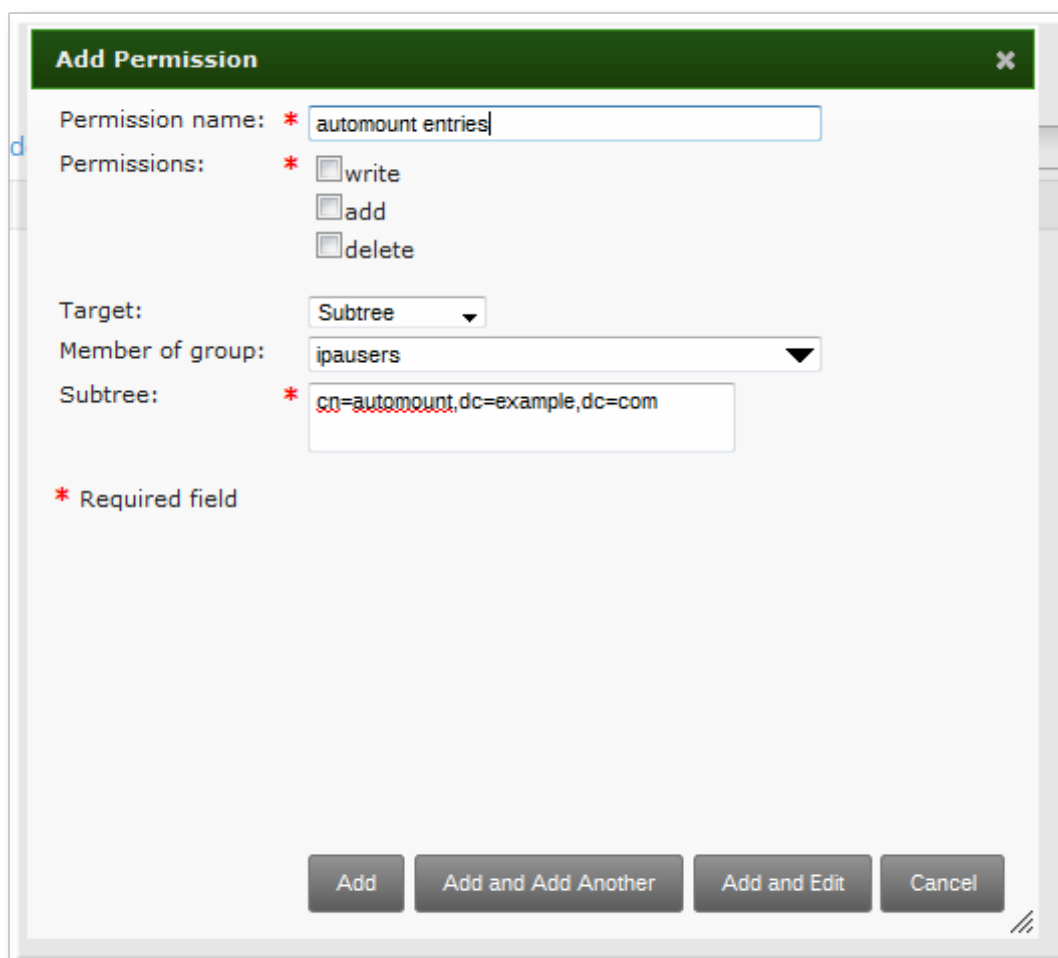
Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

24.4.2.1. Creating New Permissions from the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Select the **Permissions** task link.
3. Click the **Add** link at the top of the list of permissions.



4. Enter the name of the new permission.
5. Select the checkboxes next to the allowed operations for this permission.



6. Select the method to use to identify the target entries from the **Target** drop-down menu. There are four different methods:

- *Type* looks for an entry type like user, host, or service and then provides a list of all possible attributes for that entry type. The attributes which will be accessible through this ACI are selected from the list.
- *Filter* uses an LDAP filter to identify which entries the permission applies to.
- *Subtree* targets every entry beneath the specified subtree entry. All attributes within the matching entries can be modified.
- *Target group* specifies a user group, and all the user entries within that group are available through the ACI. All attributes within the matching entries can be modified.



NOTE

For *Filter*, *Subtree*, and *Target group* methods, no attributes are set in the UI. These must be added later using **ipa permission-mod --attrs**. If no attributes are set for the permission then, by default, all attributes are excluded.

7. Fill in the required information to identify the target entries, depending on the selected type.
8. Click the **Add** button to save the permission.
9. For *Filter*, *Subtree*, and *Target group* methods, set the attributes for the ACI to include. This must be done from the command line.

For example:

```
[jsmith@ipaserver ~]$ ipa permission-mod "manage Windows groups" --
attrs=description,member
```

24.4.2.2. Creating New Permissions from the Command Line

A new permission is added using the **permission-add** command. All permissions require a list of attributes over which permission is granted (**--attr**), a list of allowed actions (**--permissions**), and the target entries for the ACI. There are four methods to identify the target entries:

- **--type** looks for an entry type like user, host, or service and then provides a list of all possible attributes for that entry type.
- **--filter** uses an LDAP filter to identify which entries the permission applies to.
- **--subtree** targets every entry beneath the specified subtree entry.
- **--targetgroup** specifies a user group, and all the user entries within that group are available through the ACI.

Example 24.1. Adding a Permission with a Filter

A filter can be any valid LDAP filter.

```
$ ipa permission-add "manage Windows groups" --
filter="(!(objectclass=posixgroup))" --permissions=write --attrs=description
```



NOTE

The **permission-add** command does not validate the given LDAP filter. Verify that the filter returns the expected results before configuring the permission.

Example 24.2. Adding a Permission for a Subtree

All a subtree filter requires is a DN within the directory. Since IdM uses a simplified, flat directory tree structure, this can be used to target some types of entries, like automount locations, which are containers or parent entries for other configuration.

```
$ ipa permission-add "manage automount locations" --
  subtree="ldap://ldap.example.com:389/cn=automount,dc=example,dc=com" --
  permissions=write --attrs=automountmapname,automountkey,automountInformation
```

Example 24.3. Adding a Permission Based on Object Type

There seven object types that can be used to form a permission:

- ▶ user
- ▶ group
- ▶ host
- ▶ service
- ▶ hostgroup
- ▶ netgroup
- ▶ dnsrecord

Each type has its own set of allowed attributes, in a comma-separated list.

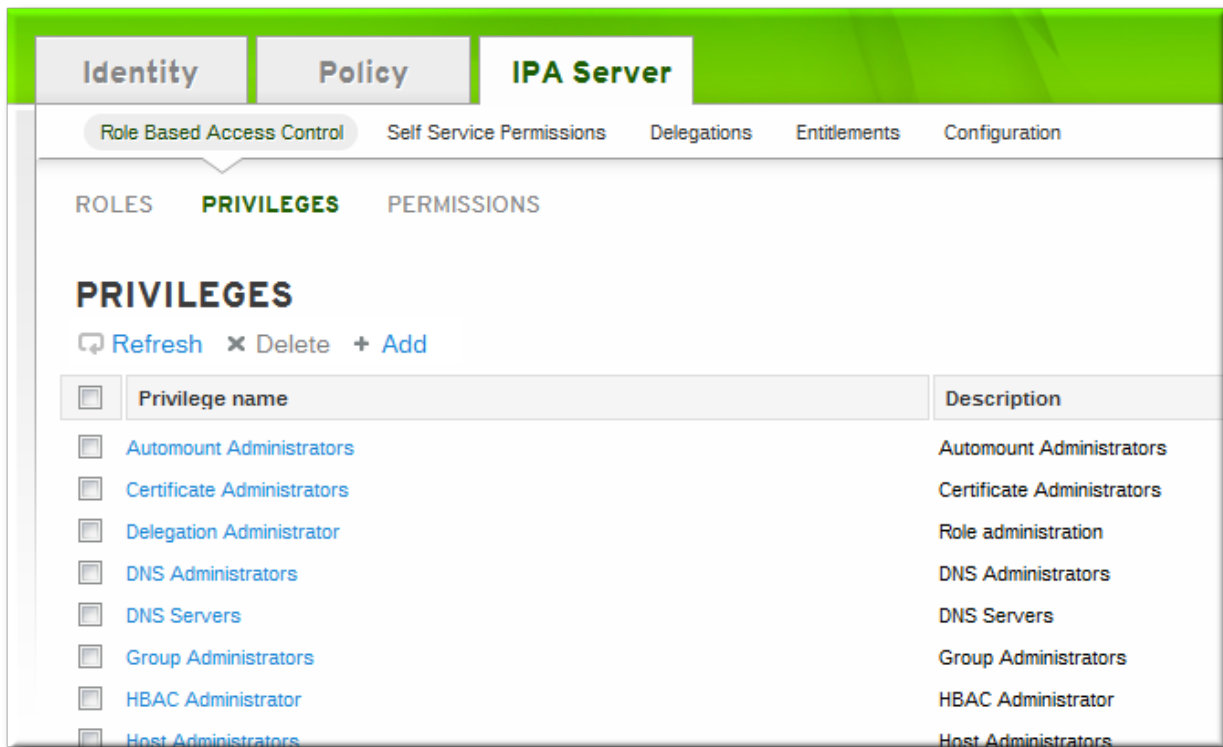
```
$ ipa permission-add "manage service" --permissions=all --type=service --
  attrs=krbprincipalkey,krbprincipalname,managedby
```

The attributes (**--attrs**) must exist and be allowed attributes for the given object type, or the permission operation fails with schema syntax errors.

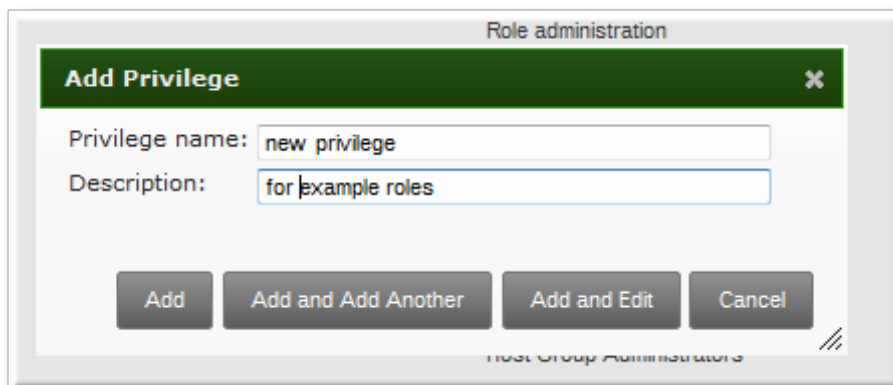
24.4.3. Creating New Privileges

24.4.3.1. Creating New Privileges from the Web UI

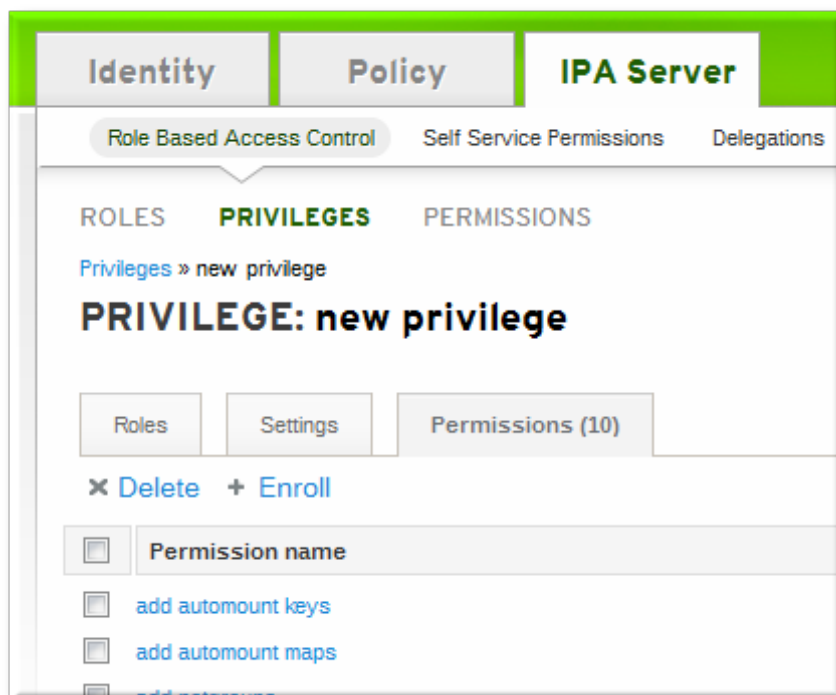
1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Select the **Privileges** task link.
3. Click the **Add** link at the top of the list of privileges.



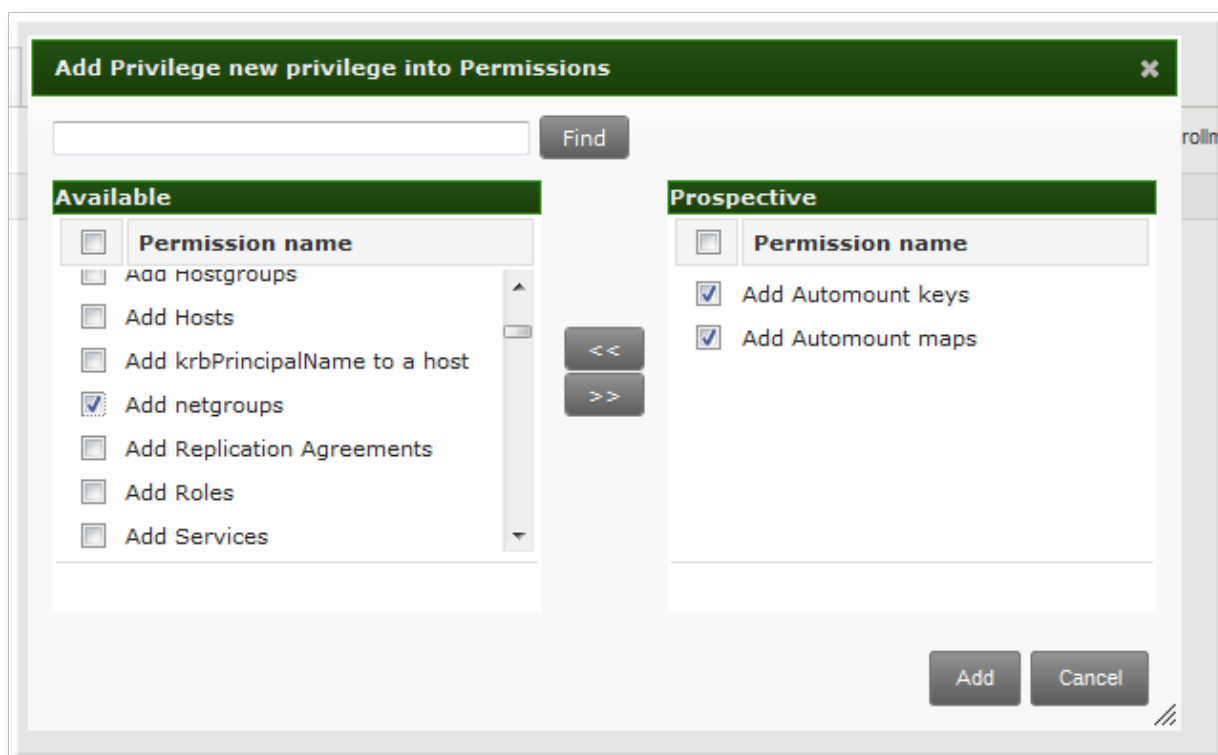
4. Enter the name and a description of the privilege.



5. Click the **Add and Edit** button to go to the privilege configuration page to add permissions.
6. Select the **Permissions** tab.
7. Click the **Add** link at the top of the list of permissions to add permission to the privilege.



- Click the checkbox by the names of the permissions to add, and click the right arrows button, >>, to move the permissions to the selection box.



- Click the **Add** button.

24.4.3.2. Creating New Privileges from the Command Line

Privilege entries are created using the **privilege-add** command, and then permissions are added to the privilege group using the **privilege-add-permission** command.

- Create the privilege entry.

```
$ ipa privilege-add "managing filesystems" --desc="for filesystems"
```

2. Assign the desired permissions. For example:

```
$ ipa privilege-add-permission "managing filesystems" --  
permissions="managing automount","managing ftp services"
```

Chapter 25. Configuration: Configuring the IdM Server

The IdM servers and backend services are configured with default settings that are applicable in most environments.

There are some configuration areas where the IdM server configuration can be tweaked to improve security or performance in certain situations.

This chapter covers information about the IdM configuration, including files and logs used by the IdM server, and procedures for updating the IdM server configuration itself.

25.1. Identity Management Files and Logs

Identity Management is a unifying framework that combines disparate Linux services into a single management context. However, the underlying technologies — such as Kerberos, DNS, 389 Directory Server, and Dogtag Certificate System — retain their own configuration files and log files. Identity Management directly manages each of these elements through their own configuration files and tools.

This section covers the directories, files, and logs used specifically by IdM. For more information about the configuration files or logs for a specific server used within IdM, see the product documentation.

25.1.1. A Reference of IdM Server Configuration Files and Directories

Table 25.1. IdM Server Configuration Files and Directories

Directory or File	Description
Server Configuration	
/etc/ipa	The main IdM configuration directory.
/etc/ipa/default.conf	The primary configuration file for IdM.
/etc/ipa/ca.crt	The CA certificate issued by the IdM server's CA.
~/ipa/	A user-specific IdM directory that is created on the local system in the system user's home directory the first time the user runs an IdM command.
IdM Logs	
~/ipa/log/cli.log	The log file for all XML-RPC calls and responses by the IdM command-line tools. This is created in the home directory for the <i>system user</i> who runs the tools, who may have a different name than the IdM user.
/var/log/ipaclient-install.log	The installation log for the client service.
/var/log/ipaserver-install.log	The installation log for the IdM server.
System Services	
/etc/rc.d/init.d/ipa	The IdM server init script.
Web UI	
/etc/ipa/html	A symlink directory in the main configuration directory for the HTML files used by the IdM web UI.
/etc/httpd/conf.d/ipa.conf	The configuration files used by the Apache host for the web UI application.
/etc/httpd/conf.d/ipa-rewrite.conf	
/etc/httpd/conf/ipa.keytab	The keytab file used by the web UI service.
/usr/share/ipa	The main directory for all of the HTML files, scripts, and stylesheets used by the web UI.
/usr/share/ipa/ipa-rewrite.conf	The configuration files used by the Apache host for the web UI application.
/usr/share/ipa/ipa.conf	
/usr/share/ipa/updates	Contains any updated files, schema, and other elements for Identity Management.
/usr/share/ipa/html	Contains the HTML files, JavaScript files, and stylesheets used by the web UI.
/usr/share/ipa/ipaclient	Contains the JavaScript files used to access Firefox's autoconfiguration feature and set up the Firefox browser to work in the IdM Kerberos realm.
/usr/share/ipa/migration	Contains HTML pages, stylesheets, and Python scripts used for running the IdM server in migration mode.
/usr/share/ipa/ui	Contains all of the scripts used by the UI to perform IdM operations.

/var/log/httpd	The log files for the Apache web server.
Kerberos	
/etc/krb5.conf	The Kerberos service configuration file.
SSSD	
/etc/sss/sssd.api.d/sss-ipa.conf	The configuration file used to identify the IdM server, IdM Directory Server, and other IdM services used by SSSD.
/var/log/sss	The log files for SSSD.
389 Directory Server	
/var/lib/dirsrv/slapd- <i>REALM_NAME</i>	All of the schema, configuration, and database files associated with the Directory Server instance used by the IdM server.
/var/log/dirsrv/slapd- <i>REALM_NAME</i>	Log files associated with the Directory Server instance used by the IdM server.
Dogtag Certificate System	
/etc/pki-ca	The main directory for the IdM CA instance.
/etc/pki-ca/conf/CS.cfg	The main configuration file for the IdM CA instance.
/var/lib/dirsrv/slapd-PKI-IPA/	All of the schema, configuration, and database files associated with the Directory Server instance used by the IdM CA.
/var/log/dirsrv/slapd-PKI-IPA/	Log files associated with the Directory Server instance used by the IdM CA.
Cache Files	
/var/cache/ipa	Cache files for the IdM server and the IdM Kerberos password daemon.
System Backups	
/var/lib/ipa/sysrestore	Contains backups of all of the system files and scripts that were reconfigured when the IdM server was installed. These include the original .conf files for NSS, Kerberos (both krb5.conf and kdc.conf), and NTP.
/var/lib/ipa-client/sysrestore	Contains backups of all of the system files and scripts that were reconfigured when the IdM client was installed. Commonly, this is the sss.conf file for SSSD authentication services.

25.1.2. About default.conf and Context Configuration Files

Certain global defaults — like the realm information, the LDAP configuration, and the CA settings — are stored in the **default.conf** file. This configuration file is referenced when the IdM client and servers start and every time the **ipa** command is run to supply information as operations are performed.

The parameters in the **default.conf** file are simple *attribute=value* pairs. The attributes are case-insensitive and order-insensitive.

```
[global]
basedn=dc=example,dc=com
realm=EXAMPLE.COM
domain=example.com
xmlrpc_uri=https://server.example.com/ipa/xml
ldap_uri=ldapi://%2fvar%2frun%2fslapd-EXAMPLE-COM.socket
enable_ra=True
ra_plugin=dogtag
mode=production
```

When adding more configuration attributes or overriding the global values, users can create additional *context* configuration files. A **server.conf** and **cli.conf** file can be created to create different options when the IdM server is started or when the **ipa** command is run, respectively. The IdM server checks the **server.conf** and **cli.conf** files first, and then checks the **default.conf** file.

Any configuration files in the **/etc/ipa** directory apply to all users for the system. Users can set individual overrides by creating **default.conf**, **server.conf**, or **cli.conf** files in their local IdM directory, **~/ipa/**. This optional file is merged with **default.conf** and used by the local IdM services.

25.1.3. Checking IdM Server Logs

Identity Management unifies several different Linux services, so it relies on those services' native logs for tracking and debugging those services.

The other services (Apache, 389 Directory Server, and Dogtag Certificate System) all have detailed logs and log levels. See the specific server documentation for more information on return codes, log formats, and log levels.

Table 25.2. IdM Log Files

Service	Log File	Description	Additional Information
IdM server	/var/log/ipaserver-install.log	Server installation log	
IdM server	~/ipa/log/cli.log	Command-line tool log	
IdM client	/var/log/ipaclient-install.log	Client installation log	
Apache server	/var/log/httpd/access /var/log/httpd/error	These are standard access and error logs for Apache servers. Both the web UI and the XML-RPC command-line interface use Apache, so some IdM-specific messages will be recorded in the error log along with the Apache messages.	Apache log chapter
Dogtag Certificate System	/var/log/pki-ca-install.log	The installation log for the IdM CA.	
Dogtag Certificate System	/var/log/pki-ca/debug /var/log/pki-ca/system /var/log/pki-ca/transactions /var/log/pki-ca/signedAudit	These logs mainly relate to certificate operations. In IdM, this is used for service principals, hosts, and other entities which use certificates.	Logging chapter
389 Directory Server	/var/log/dirsrv/slapd-RE-ALM/access /var/log/dirsrv/slapd-RE-ALM/audit /var/log/dirsrv/slapd-RE-ALM/errors	The access and error logs both contain detailed information about attempted access and operations for the domain Directory Server instance. The error log setting can be changed to provide very detailed output.	The access log is buffered, so the server only writes to the log every 30 seconds, by default. ► Monitoring servers and databases ► Log entries explained
389 Directory Server	/var/log/dirsrv/slapd-RE-ALM/access /var/log/dirsrv/slapd-RE-ALM/audit /var/log/dirsrv/slapd-RE-ALM/errors	This directory server instance is used by the IdM CA to store certificate information. Most operational data here will be related to server-replica interactions.	The access log is buffered, so the server only writes to the log every 30 seconds, by default. ► Monitoring servers and databases ► Log entries explained

Kerberos	<code>/var/log/krb5libs.log</code>	This is the primary log file for Kerberos connections.	This location is configured in the krb5.conf file, so it could be different on some systems.
Kerberos	<code>/var/log/krb5kdc.log</code>	This is the primary log file for the Kerberos KDC server.	This location is configured in the krb5.conf file, so it could be different on some systems.
Kerberos	<code>/var/log/kadmind.log</code>	This is the primary log file for the Kerberos administration server.	This location is configured in the krb5.conf file, so it could be different on some systems.
DNS	<code>/var/log/messages</code>	DNS error messages are included with other system messages.	DNS logging is <i>not</i> enabled by default. DNS logging is enabled by running the querylog command: <pre>/usr/sbin/rndc querylog</pre> This begins writing log messages to the system's /var/log/messages file. To turn off logging, run the querylog command again.

25.1.3.1. Enabling Server Debug Logging

Debug logging for the IdM server is set in the **server.conf** file.



NOTE

Editing the **defaults.conf** configuration file affects all IdM components, not only the IdM server.

1. Edit or create the **server.conf** file.

```
vim server.conf
```

2. Add the **debug** line and set its value to true.

```
[global]
debug=True
```


3. Restart the Apache daemon to load the changes.

```
service httpd restart
```

25.1.3.2. Debugging Command-Line Operations

Any command-line operation with the **ipa** command can return debug information by using the **-v** option. For example:

```
$ ipa -v user-show admin
ipa: INFO: trying https://ipaserver.example.com/ipa/xml
First name: John
Last name: Smythe
User login [jsmythe]:
ipa: INFO: Forwarding 'user_add' to server u'https://ipaserver.example.com/ipa/xml'
-----
Added user "jsmythe"
-----
  User login: jsmythe
  First name: John
  Last name: Smythe
  Full name: John Smythe
  Display name: John Smythe
  Initials: JS
  Home directory: /home/jsmythe
  GECOS field: John Smythe
  Login shell: /bin/sh
  Kerberos principal: jsmythe@EXAMPLE.COM
  UID: 1966800003
  GID: 1966800003
  Keytab: False
  Password: False
```

Using the option twice, **-vv**, displays the XML-RPC exchange:

```
$ ipa -vv user-add
```

```
ipa: INFO: trying https://ipaserver.example.com/ipa/xml
First name: Jane
Last name: Russell
User login [jruscell]:
ipa: INFO: Forwarding 'user_add' to server u'https://ipaserver.example.com/ipa/xml'
send: u'POST /ipa/xml HTTP/1.0\r\nHost: ipaserver.example.com\r\nAccept-Language:
en-us\r\nAuthorization: negotiate
YIIFgQYJKoZIhvcSAQICAQBuggVwMIIIFbKADAgEFoQMCAQ6iBwMFACAAAACjggGHYYIBGzCCAX+gAwIBBa
EZGxdSSFRTLKvORy5CT1MuUkVESEFULkNPTaI5MDegAwIBA6EwMC4bBEhUVFABJmRlbGwtcGUxODUwLTAX
LnJodHMwZ5nLmJvcy5yZWRoYXQuY29to4IBIDCCARYgAwIBEqEDAgECooIBDgSCAQpV2YEWv03X+SEND
UBf0hMFGc3Fvnd51nELV0rIB1tfGVjpNlkuXxKSfFKVD3vyAUqkii255T0mnXyTwayE93W1U4s0shetmG
50zeU4KDMhuupzQZSCb5xB0KPU4HMDvP1UnDFJUGCK9tcqDjiYE+1JrEcZ8H+Vxvvl+nP6yIdUQKqoEuN
hJaLWiiT8ieAzk8zvmDlDzpFYtInGwe9D5ko1Bb7Npu0SEpdVJB2gnB5vszCidLlzM4JUqX8p21AZV0UY
A6QZOWX90XhQhDElKcuHCN2s9FBRoFYK83gf1voS7xSFlzZaFsEGHNmdA0qXbzREKqUr8fmWmNvBGpD
iR2ILQep09lL56JqSCA8owggPGoAMCARKigg09BIIDuarbB67zjmBu9Ax2K+0klSD99pNv97h9yxol8c6
NGLB4CmE8Mo39rL4MMXHeOS00Cbn+TD97XVGLu+cgkfVcuIG4PMMBoajuSnPmIf7qDvfa8IYDFLDDnRB7I
//IXtCc/Z4rBbaxk0SMIRLrSkf5wha7awtN1JbizBMQw+J0Uln8JjsWxu0Ls75hBtIDbPf3fva3vWbF7kT
BChBsheewSAlck9qUglyNxAODgFVvRrXbXfw51Uo++9qHnhh+zFSWepfv7US7RYK0Kx0KFd+uauY1ES+x1
nMvK18ap2pcy0odBkKu1kwJDND0JXUDSY08MXK2zb/UGWrVEf6GIsBgu122UGiHp6C+0fEu+nRrvt0RY65
Bgi8E1vm55fbb/9dQWNCqheL9m6QJWPw0rgc+E5S0990N6x3Vv2+Zk17EmbZXinPd2tDe7fJ9cS9o/z7
Qjw8z8vvSzHL4GX7FKi2HJdBST3nEgOC8Pq046UnAJcA8pf1ZkWCk9xDWH+5PSph6WnvpRqugqf/6cq+3
jk3MEjCrX+JBj8QL6AgN3oEB4kvjZpAC+FFtkdX59VLDwfl/r0gmW3ZNk0nLLCLkiiYUMTEHZBzJw9kFbs
X3LmS8qQQA6rQ2L782DYisElywFZ/0Sax8JO/G62Zvy7BHy7SQSGIvcdA0afeNyfxawM1vTpvSh0Grnl1
Yfs3FhZAKnVcLYr1PTapR23uLgRMv+0c9wAbwuUDfKg00l5vAd1j55VuyapgDEzi/URsLdVdcF4zigT4Kr
TByCwU2/pI6FmEPqB2tsjM2A8JmqA+9Nt8bjaNdNwCOWE0dF50zeL9P8oodWGkbRZLk4DLIurpCW1d6IyT
BhPQ5qZqHJWeoGiFa5y94zBpp27goMPmE0BskXT0JQmveYf10eKEMSzyiWPL2mwi7KEMtfGcPwTIGP2LR
E/QxNvPGkwFF0+PDjZGVw+APKkMKqc1VXxhtJA/2NmBr01pZIIJ9R+41sR/QoACcXIUXJnhrTwwR1viKC
B5Tec87gN+e0Cf0g+fmZuXNRscwJfhYQJYwJqdYzGtZW+h8jDWqa2EPcDwIQwyFagXNQ/aMvh1yNTECPLE
grMhYmFAUdLQzI2BDnfbDftIs0rXjSC0oZn/Uaoqdr4F5sy0rYAXH47bS6MW8CxyylreH8n2tQXjenakL
FHcnjt4M1n0c/igzNSEz28qW9WSr4bCdKH+ra3BVPt/AF0WHWkxGF4vWr/iNHCjq8fLF+DsAEx0Zs696Rg
0fwZy079A\r\nUser-Agent: xmlrpc.lib.py/1.0.1 (by www.pythonware.com)\r\nContent-
Type: text/xml\r\nContent-Length: 1240\r\n\r\n'
send: "<?xml version='1.0' encoding='UTF-8'?
>\n<methodCall>\n<methodName>user_add</methodName>\n<params>\n<param>\n<value><arr
ay><data>\n<value><string>jruscell</string></value>\n</data></array></value>\n</par
am>\n<param>\n<value><struct>\n<member>\n<name>all</name>\n<value><boolean>0</boo
lean></value>\n</member>\n<member>\n<name>displayname</name>\n<value><string>Jane
Russell</string></value>\n</member>\n<member>\n<name>cn</name>\n<value><string>Jan
e
Russell</string></value>\n</member>\n<member>\n<name>noprivate</name>\n<value><bo
olean>0</boolean></value>\n</member>\n<member>\n<name>uidnumber</name>\n<value><i
nt>999</int></value>\n</member>\n<member>\n<name>raw</name>\n<value><boolean>0</b
oolean></value>\n</member>\n<member>\n<name>version</name>\n<value><string>2.11</
string></value>\n</member>\n<member>\n<name>gecos</name>\n<value><string>Jane
Russell</string></value>\n</member>\n<member>\n<name>sn</name>\n<value><string>Rus
sell</string></value>\n</member>\n<member>\n<name>krbprincipalname</name>\n<value
><string>jruscell@EXAMPLE.COM</string></value>\n</member>\n<member>\n<name>givenn
ame</name>\n<value><string>Jane</string></value>\n</member>\n<member>\n<name>init
ials</name>\n<value><string>JR</string></value>\n</member>\n</struct></value>\n</p
aram>\n</params>\n</methodCall>\n"
reply: 'HTTP/1.1 200 OK\r\n'
header: Date: Thu, 15 Sep 2011 00:50:39 GMT
header: Server: Apache/2.2.15 (Red Hat)
header: WWW-Authenticate: Negotiate
YIGZBgkqhkiG9xIBAgICAg+BiTCBhQADAgEFoQMCAQ+iejB4oAMCARKicQRvVl5x6Zt9PbWNzvPEWkdu+
3PTCq/ZVKjGHM+1zDBz81GL/f+/Pr75zTuveLYn9de0C3k27vz96fn2HQsy9qVH7sfqn0RWGQWz1+kDkuD
6bJ/Dp/mpJvicW5gSkCSH6/UCNuE4I0xqwabLIz8MM/5o
header: Connection: close
```

```

header: Content-Type: text/xml; charset=utf-8
body: "<?xml version='1.0' encoding='UTF-8'?
>\n<methodResponse>\n<params>\n<param>\n<value><struct>\n<member>\n<name>result</n
ame>\n<value><struct>\n<member>\n<name>dn</name>\n<value><string>uid=jrussell,cn=u
sers,cn=accounts,dc=example,dc=com</string></value>\n</member>\n<member>\n<name>h
as_keytab</name>\n<value><boolean>0</boolean></value>\n</member>\n<member>\n<name
>displayname</name>\n<value><array><data>\n<value><string>Jane
Russell</string></value>\n</data></array></value>\n</member>\n<member>\n<name>uid
</name>\n<value><array><data>\n<value><string>jrussell</string></value>\n</data></a
rray></value>\n</member>\n<member>\n<name>objectclass</name>\n<value><array><data
>\n<value><string>top</string></value>\n<value><string>person</string></value>\n<va
lue><string>organizationalperson</string></value>\n<value><string>inetorgperson</st
ring></value>\n<value><string>inetuser</string></value>\n<value><string>posixaccoun
t</string></value>\n<value><string>krbprincipalaux</string></value>\n<value><string
>krbticketpolicyaux</string></value>\n<"
body:
'value><string>ipaobject</string></value>\n</data></array></value>\n</member>\n<me
mber>\n<name>loginshell</name>\n<value><array><data>\n<value><string>/bin/sh</stri
ng></value>\n</data></array></value>\n</member>\n<member>\n<name>uidnumber</name>
\n<value><array><data>\n<value><string>1966800004</string></value>\n</data></array
></value>\n</member>\n<member>\n<name>initials</name>\n<value><array><data>\n<valu
e><string>JR</string></value>\n</data></array></value>\n</member>\n<member>\n<nam
e>gidnumber</name>\n<value><array><data>\n<value><string>1966800004</string></valu
e>\n</data></array></value>\n</member>\n<member>\n<name>gecos</name>\n<value><arr
ay><data>\n<value><string>Jane
Russell</string></value>\n</data></array></value>\n</member>\n<member>\n<name>sn</
name>\n<value><array><data>\n<value><string>Russell</string></value>\n</data></arra
y></value>\n</member>\n<member>\n<name>homedirectory</name>\n<value><array><data>
\n<value><string>/home/jrussell</string></value>\n</data></array></value>\n</membe
r>\n<member>\n<name>has_password</name>\n<value><boolean>0</'
body:
'boolean></value>\n</member>\n<member>\n<name>krbprincipalname</name>\n<value><ar
ray><data>\n<value><string>jrussell@EXAMPLE.COM</string></value>\n</data></array></
value>\n</member>\n<member>\n<name>givenname</name>\n<value><array><data>\n<value
><string>Jane</string></value>\n</data></array></value>\n</member>\n<member>\n<nam
e>cn</name>\n<value><array><data>\n<value><string>Jane
Russell</string></value>\n</data></array></value>\n</member>\n<member>\n<name>ipau
niqueid</name>\n<value><array><data>\n<value><string>bba27e6e-df34-11e0-a5f4-
001143d2c060</string></value>\n</data></array></value>\n</member>\n</struct></valu
e>\n</member>\n<member>\n<name>value</name>\n<value><string>jrussell</string></va
lue>\n</member>\n<member>\n<name>summary</name>\n<value><string>Added user
"jrussell"</string></value>\n</member>\n</struct></value>\n</param>\n</params>\n</
methodResponse>\n'
```

```

-----
Added user "jrussell"
-----
User login: jrussell
First name: Jane
Last name: Russell
Full name: Jane Russell
Display name: Jane Russell
Initials: JR
Home directory: /home/jrussell
GECOS field: Jane Russell
Login shell: /bin/sh
Kerberos principal: jrussell@EXAMPLE.COM
UID: 1966800004
GID: 1966800004
Keytab: False
Password: False

```

**IMPORTANT**

The **-v** and **-vv** options are *global* options and must be used before the subcommand when running **ipa**.

25.2. Disabling Anonymous Binds

Accessing domain resources and running client tools always require Kerberos authentication. However, the backend LDAP directory used by the IdM server allows anonymous binds by default. This potentially opens up all of the domain configuration to unauthorized users, including information about users, machines, groups, services, netgroups, and DNS configuration.

It is possible to disable anonymous binds on the 389 Directory Server instance by using LDAP tools to reset the **nsslapd-allow-anonymous-access** attribute.

1. Change the **nsslapd-allow-anonymous-access** attribute to **rootdse**.

```
ldapmodify -x -D "cn=Directory Manager" -w secret -h server.example.com -p 389
```

```
Enter LDAP Password:
dn: cn=config
changetype: modify
replace: nsslapd-allow-anonymous-access
nsslapd-allow-anonymous-access: rootdse
```

**IMPORTANT**

Anonymous access can be completely allowed (on) or completely blocked (off). However, completely blocking anonymous access also blocks external clients from checking the server configuration. LDAP and web clients are not necessarily domain clients, so they connect anonymously to read the root DSE file to get connection information.

The **rootdse** allows access to the root DSE and server configuration *without* any access to the directory data.

2. Restart the 389 Directory Server instance to load the new setting.

```
service dirsrv restart
```

25.3. Configuring Alternate Certificate Authorities

IdM creates a Dogtag Certificate System certificate authority (CA) during the server installation process. To use an external CA, it is possible to create the required server certificates and then import them into the 389 Directory Server and the HTTP server, which require IdM server certificates.

**TIP**

Save an ASCII copy of the CA certificate as `/usr/share/ipa/html/ca.crt`. This allows users to download the correct certificate when they configure their browsers.

1. Use the `ipa-server-certinstall` command to install the certificate.

```
# /usr/sbin/ipa-server-certinstall -d /path/to/pkcs12.p12
```

2. To keep using browser autoconfiguration in Firefox, regenerate the `/usr/share/ipa/html/configure.jar` file.
 - a. Create a directory, and then create the new security databases in that directory.

```
# mkdir /tmp/signdb
# certutil -N -d /tmp/signdb
```

- b. Import the PKCS #12 file for the signing certificate into that directory.

```
# pk12util -i /path/to/pkcs12.p12 -d /tmp/signdb
```

- c. Make a temporary signing directory, and copy the IdM JavaScript file to that directory.

```
# mkdir /tmp/sign
# cp /usr/share/ipa/html/preferences.html /tmp/sign
```

- d. Use the object signing certificate to sign the JavaScript file and to regenerate the `configure.jar` file.

```
# signtool -d /tmp/signdb -k Signing_cert_nickname -Z
/usr/share/ipa/html/configure.jar -e .html /tmp/sign
```

25.4. Configuring CRLs and OCSP Responders

A certificate is created with a validity period, meaning it has a point where it expires and is no longer valid. The expiration date is contained in the certificate itself, so a client always checks the validity period in the certificate to see if the certificate is still valid.

However, a certificate can also be revoked before its validity period is up, but this information is not contained in the certificate. A CA publishes a *certificate revocation list* (CRL), which contains a complete list of every certificate that was issued by that CA and subsequently revoked. A client can check the CRL to see if a certificate within its validity period has been revoked and is, therefore, invalid.

Validity checks are performed using the online certificate status protocol (OCSP), which sends a request to an *OCSP responder*. Each CA integrated with the IdM server uses an internal OCSP responder, and any client which runs a validity check can check the IdM CA's internal OCSP responder.

Every certificate issued by the IdM CA puts its OCSP responder service URL in the certificate. For example:

```
http://ipaserver.example.com:9180/ca/ocsp
```

**NOTE**

For the IdM OCSP responder to be available, port 9180 needs to be open in the firewall.

25.4.1. Using an OSCP Responder with SELinux

Clients can use the Identity Management OCSP responder to check certificate validity or to retrieve CRLs. A client can be a number of different services, but is most frequently an Apache server and the `mod_revocator` module (which handles CRL and OCSP operations).

The Identity Management CA has an OCSP responder listening over port 9180, which is also the port available for CRL retrieval. This port is protected by default SELinux policies to prevent unauthorized access. If an Apache server attempts to connect to the OCSP port, then it may be denied access by SELinux.

The Apache server, on the local machine, must be granted access to port 9180 for it to be able to connect to the Identity Management OCSP responder. There are two ways to work around this by changing the SELinux policies:

- Edit the SELinux policy to allow Apache servers using the `mod_revocator` module to connect to port 9180:

```
semodule -i revoker.pp
```

- Generate a new SELinux policy to allow access based on the SELinux error logs for the `mod_revocator` connection attempt.

```
audit2allow -a -M revoker
```

25.4.2. Changing the CRL Update Interval

The CRL file is automatically generated by the Dogtag Certificate System CA every four hours. This interval can be changed by editing the Dogtag Certificate System configuration.

1. Stop the CA server.

```
service pki-ca stop
```

2. Open the **CS.cfg** file.

```
vim /etc/pki-ca/CS.cfg
```

3. Change the **`ca.crl.MasterCRL.autoUpdateInterval`** to the new interval setting.
4. Restart the CA server.

```
service pki-ca start
```

25.4.3. Changing the OCSP Responder Location

Each IdM server generates its own CRL. Likewise, each IdM server uses its own OCSP responder, with its own OCSP responder URL in the certificates it issues.

A DNS CNAME can be used by IdM clients, and then from there be redirected to the appropriate IdM

server OCSP responder.

1. Open the certificate profile.

```
vim /var/lib/pki-ca/profiles/ca/caIPAserviceCert.cfg
```

2. Change the ***policyset.serverCertSet.9.default.params.crlDistPointsPointName_0*** parameter to the DNS CNAME hostname.
3. Restart the CA server.

```
service pki-ca restart
```

That change must be made on every IdM server, with the ***crlDistPointsPointName_0*** parameter set to the same hostname.

25.5. Changing Domain DNS Configuration

25.5.1. Setting DNS Entries for Multi-Homed Servers

Some server machines may support multiple network interface cards (NICs). Multi-homed machines typically have multiple IPs, all assigned to the same hostname. This works fine in IdM most of the time because it listens on all available interfaces, except localhost. For a server to be available through any NIC, edit the DNS zone file and add entries for each IP address. For example:

```
ipaserver IN A 192.168.1.100
ipaserver IN A 192.168.1.101
ipaserver IN A 192.168.1.102
```

25.5.2. Setting up Additional Name Servers

The list of configured nameservers in */etc/resolv.conf* only contains the IdM server itself when configuration is finished. If the local **named** service ever crashes, then the IdM server is unable to run and DNS services for the entire domain are no longer available.

Other DNS servers should be added manually to the IdM server's */etc/resolv.conf* file.

```
[root@server ~]# vim /etc/resolv.conf

search example.com

; the IdM server
nameserver 127.0.0.1

; backup DNS servers
nameserver 198.51.100.0
nameserver 192.0.2.0
```



NOTE

A default limit of three servers is set for the */etc/resolv.conf* file.

Other information about configuring the */etc/resolv.conf* file is given in the **resolv.conf**

manpage.

25.6. Changing Load Balancing for IdM Servers and Replicas

As [Section 1.3.1, “About IdM Servers and Replicas”](#) touches on, IdM servers and replicas in the domain automatically share the load among instances to maintain performance. The load balancing is defined first by the *priority* set for the server or replica in its SRV entry, and then by the *weight* of that instance for servers/replicas with the same priority. Clients contact servers/replicas with the highest priority and then work their way down.

Load balancing is done automatically by servers, replicas, and clients. The configuration used for load balancing can be altered by changing the priority and the weight given to a server or replica.

(All replicas are initially created with the same priority.)

For example, this gives server1 a higher priority than server 2, meaning it will be contacted first:

```
$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="0 100 389
server1.example.com."

$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="1 100 389
server2.example.com."
```

More information about SRV records is in [RFC 2782](#).

25.7. Managing Replication Agreements Between IdM Servers

Information is shared between the IdM servers and replicas using *multi-master replication*. What this means is that servers and replicas all receive updates and, therefore, are data masters. The domain information is copied between the servers and replicas using *replication*.

As replicas are added to the domain, mutual replication agreements are automatically created between the replica and the server it is based on. Additional replication agreements can be created between other replicas and servers or the configuration of the replication agreement can be changed using the **ipa-replica-manage** command.

When a replica is created, the replica install script creates two replication agreements: one going from the master server to the replica and one going from the replica to the master server.

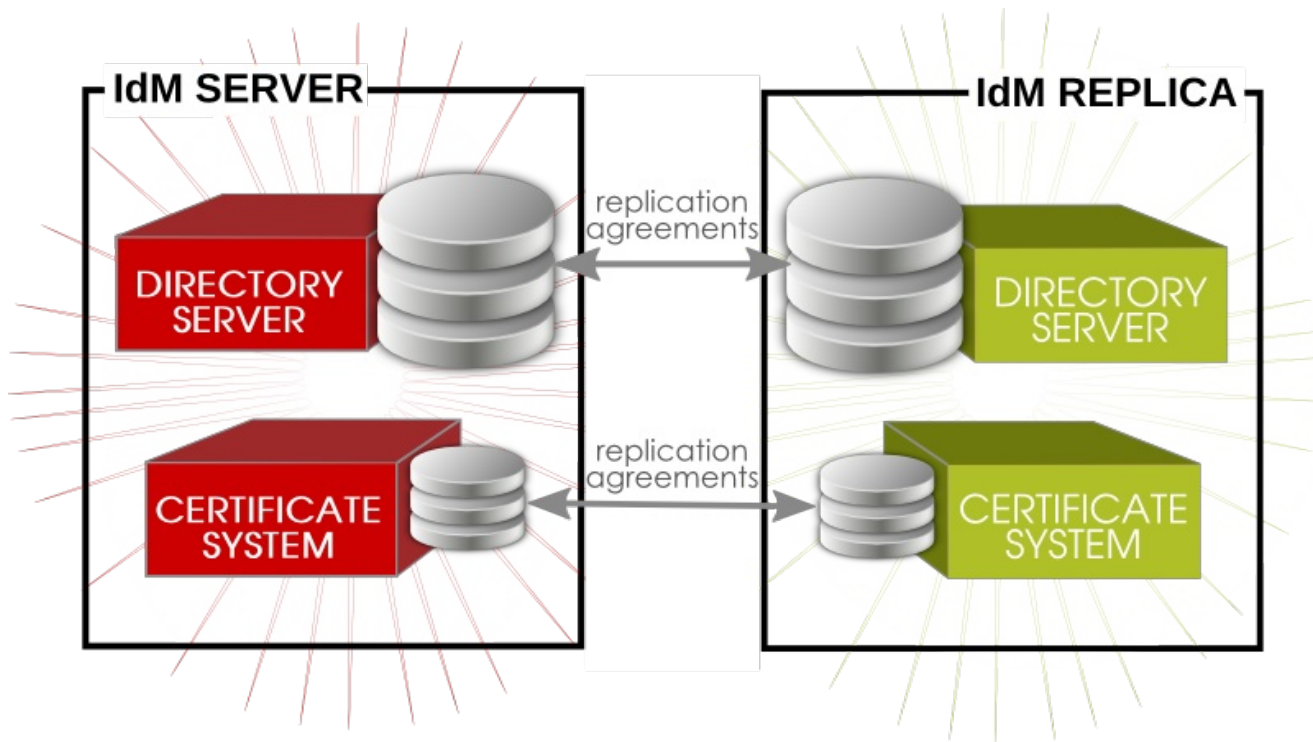


Figure 25.1. Server and Replica Agreements

As more replicas and servers are added to the domain, there can be replicas and servers that have replication agreements to other servers and replicas but not between each other. For example, the first IdM server is Server A. Then, the admin creates Replica B, and the install script creates a Server A => Replica B replication agreement and a Replica B => Server A replication agreement. Next, the admin creates Replica C based on Server A. The install script creates a Server A => Replica C replication agreement and a Replica C => Server A replication agreement. Replica B and Replica C both have replication agreements with Server A — but they do not have agreements with each other. For data availability, consistency, failover tolerance, and performance, it can be beneficial to create a pair of replication agreements between Replica B and Replica C, even though their data will eventually be replicated over to each other through replication with Server A.

25.7.1. Listing Replication Agreements

The **ipa-replica-manage** command can list all of the servers and replicas in the replication topology, using the **list** command:

```
# ipa-replica-manage list
srv1.example.com
srv2.example.com
srv3.example.com
srv4.example.com
```

After getting the server/replica list, then it is possible to list the replication agreements for the server. These are the other servers/replicas to which the specified server sends updates.

```
# ipa-replica-manage list srv1.example.com
srv2.example.com
srv3.example.com
```

25.7.2. Creating and Removing Replication Agreements

Replication agreements are created by *connecting* one server to another server.

```
ipa-replica-manage server1 server2
```

If only one server is given, the replication agreements are created between the local host and the specified server.

For example:

```
# ipa-replica-manage connect srv2.example.com srv4.example.com
```

Replication occurs over standard LDAP; to enable SSL, then include the CA certificate for the local host (or the specified *server1*). The CA certificate is then installed in the remote server's certificate database to enable TLS/SSL connections. For example:

```
# ipa-replica-manage connect --cacert=/etc/ipa/ca.crt srv2.example.com  
srv4.example.com
```

To remove a replication agreement between specific servers/replicas, use the **disconnect** command:

```
# ipa-replica-manage disconnect srv2.example.com srv4.example.com
```

Using the **disconnect** command removes that one replication agreement but leaves both the server/replica instances in the overall replication topology. To remove a server entirely from the IdM replication topology, with all its data, (and, functionally, removing it from the IdM domain as a server), use the **del** server:

```
# ipa-replica-manage del srv2.example.com
```

25.7.3. Forcing Replication

Replication between servers and replicas occurs on a schedule. Although replication is frequent, there can be times when it is necessary to initiate the replication operation manually. For example, if a server is being taken offline for maintenance, it is necessary to flush all of the queued replication changes out of its changelog before taking it down.

To initiate a replication update manually, use the **force-sync** command. The server which receives the update is the local server; the server which sends the updates is specified in the **--from** option.

```
# ipa-replica-manage force-sync --from srv1.example.com
```

25.7.4. Reinitializing IdM Servers

When a replica is first created, the database of the master server is copied, completely, over to the replica database. This process is called *initialization*. If a server/replica is offline for a long period of time or there is some kind of corruption in its database, then the server can be re-initialized, with a fresh and updated set of data.

This is done using the **re-initialize** command. The target server being initialized is the local host. The server or replica from which to pull the data to initialize the local database is specified in the **--from** option:

```
# ipa-replica-manage re-initialize --from srv1.example.com
```

25.7.5. Resolving Replication Conflicts

Changes — both for IdM domain data and for certificate and key data — are replicated between IdM servers and replicas (and, in similar paths, between IdM and Active Directory servers).

Even though replication operations are run continuously, there is a chance that changes can be made on one IdM server at the same time different changes are made to the same entry on a different IdM server. When replication begins to process those entries, the changes collide — this is a *replication conflict*.

Every single directory modify operation is assigned a server-specific *change state number* (CSN) to track how those modifications are propagated during replication. The CSN also contains a modify timestamp. When there is a replication conflict, the timestamp is checked and the last change wins.

Simply accepting the most recent change is effective for resolving conflicts with attribute values. That method is too blunt for some types of operations, however, which affect the directory tree. Some operations, like `modrdn`, DN changes, or adding or removing parent and child entries, require administrator review before the conflict is resolved.



NOTE

Replication conflicts are resolved by editing the entries directory in the LDAP database.

When there is a replication conflict, both entries are added to the directory and are assigned a ***nsds5ReplConflict*** attribute. This makes it easy to search for entries with a conflict:

```
ldapsearch -x -D "cn=directory manager" -w password -b "dc=example,dc=com"
"nsds5ReplConflict=*" \* nsds5ReplConflict
```

25.7.5.1. Solving Naming Conflicts

When two entries are added to the IdM domain with the same DN, both entries are added to the directory, but they are renamed to use the ***nsuniqueid*** attribute as a naming attribute. For example:

```
nsuniqueid=0a950601-435311e0-86a2f5bd-
3cd26022+uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
```

Those entries can be searched for and displayed in the IdM CLI, but they cannot be edited or deleted until the conflict is resolved and the DN is updated.

To resolve the conflict:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:

```
ldapmodify -x -D "cn=directory manager" -w secret -h ipaserver.example.com -
p 389
dn: nsuniqueid=66446001-
1dd211b2+uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
changetype: modrdn
newrdn: cn=TempValue
deleteoldrdn: 0
```

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -x -D "cn=directory manager" -w secret -h ipaserver.example.com -
p 389
dn: cn=TempValue,cn=users,cn=accounts,dc=example,dc=com
changetype: modify
delete: uid
dc: jsmith
-
delete: nsds5ReplConflict
-
```



NOTE

The unique identifier attribute **nsuniqueid** cannot be deleted.

3. Rename the entry with the intended attribute-value pair. For example:

```
ldapmodify -x -D "cn=directory manager" -w secret -h ipaserver.example.com -
p 389
dn: cn=TempValue,dc=example,dc=com
changetype: modrdn
newrdn: uid=jsmith
deleteoldrdn: 1
```

Setting the value of the **deleteoldrdn** attribute to **1** deletes the temporary attribute-value pair **cn=TempValue**. To keep this attribute, set the value of the **deleteoldrdn** attribute to **0**.

25.7.5.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a **glue** entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

Glue entries are temporary entries that include the object classes **glue** and **extensibleObject**. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the **glue** object class and the **nsds5ReplConflict** attribute.

In such cases, either modify the glue entry to remove the **glue** object class and the **nsds5ReplConflict** attribute to keep the entry as a normal entry or delete the glue entry and its child entries.

- The server creates a minimalistic entry with the **glue** and **extensibleObject** object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

25.8. Removing a Replica

Deleting or *demoting* a replica removes the IdM replica from the server/replica topology so that it no longer processes IdM requests and it also removes the host machine itself from the IdM domain.

1. On an IdM server, obtain a Kerberos ticket before running IdM tools.

```
[root@replica ~]# kinit admin
```

2. List all of the configured replication agreements for the IdM domain.

```
[root@replica ~]# ipa-replica-manage list
Directory Manager password:

ipaserver.example.com: master
ipaserver2.example.com: master
replica.example.com: master
replica2.example.com: master
```

3. Removing the replica from the topology involves deleting all the agreements between the replica and the other servers in the IdM domain and all of the data about the replica in the domain configuration.

```
[root@replica ~]# ipa-replica-manage del replica.example.com
```

4. If the replica was configured with its own CA, then also use the **ipa-csreplica-manage** command to remove all of the replication agreements between the certificate databases for the replica.

This is required if the replica itself was configured with a Dogtag Certificate System CA. It is not required if only the master server or other replicas were configured with a CA.

```
[root@replica ~]# ipa-csreplica-manage del replica.example.com
```

5. On the replica, uninstall the replica packages.

```
[root@replica ~]# ipa-server-install --uninstall -U
```

25.9. Starting IdM with Expired Certificates

If IdM administrative server certificates expire, then most IdM services will be inaccessible, including administrative services. The underlying Apache and 389 Directory Server services can be configured to allow SSL access to those services, even if the certificates are expired.



NOTE

Allowing limited access with expired certificates permits Apache, Kerberos, DNS, and 389 Directory Server services to continue working. With those services active, users are able to log into the domain.

Client services such as **sudo** that require SSL for access will still fail because of the expired server certificates.

1. Change the **mod_nss** configuration for the Apache server to not enforce valid certificates, in the **NSSEnforceValidCerts** parameter. If this parameter is not already in the file, then add it.

Set the value to **off**.

```
[root@ipaserver ~]# vim /etc/httpd/conf.d/nss.conf  
  
NSSEnforceValidCerts off
```

2. Restart Apache.

```
[root@ipaserver ~]# service httpd restart
```

3. Change the **nsslapd-validate-cert** attribute in the 389 Directory Server configuration to **warn** instead of **true** to disable validity checks.

```
[root@ipaserver ~]# ldapmodify -D "cn=directory manager" -w secret -p 389 -  
h ipaserver.example.com  
  
dn: cn=config  
changetype: modify  
replace: nsslapd-validate-cert  
nsslapd-validate-cert: warn
```

4. Restart 389 Directory Server.

```
[root@ipaserver ~]# service dirsrv restart
```

Chapter 26. Migrating from an LDAP Directory to IdM

When an infrastructure has previously deployed an LDAP server for authentication and identity lookups, it is possible to migrate the user data, including passwords, to a new Identity Management instance, without losing user or password data.

Identity Management has migration tools to help move directory data and only requires minimal updates to clients. However, the migration process assumes a simple deployment scenario (one LDAP namespace to one IdM namespace). For more complex environments, such as ones with multiple namespaces or custom schema, contact Red Hat support services for assistance.

26.1. An Overview of LDAP to IdM Migration

The actual migration part of moving from an LDAP server to Identity Management — the process of moving the data from one server to the other — is fairly straightforward. The process is simple: move data, move passwords, and move clients.

The crucial part of migration is not data migration; it is deciding how clients are going to be configured to use Identity Management. For each client in the infrastructure, you need to decide what services (such as Kerberos and SSSD) are being used and what services can be used in the final, IdM deployment.

A secondary, but significant, consideration is planning how to migrate passwords. Identity Management requires Kerberos hashes for every user account in addition to passwords. Some of the considerations and migration paths for passwords are covered in [Section 26.1.2, “Planning Password Migration”](#).

26.1.1. Planning the Client Configuration

Identity Management can support a number of different client configurations, with varying degrees of functionality, flexibility, and security. Decide which configuration is best *for each individual client* based on its operating system, functional area (such as development machines, production servers, or user laptops), and your IT maintenance priorities.



IMPORTANT

The different client configurations *are not mutually exclusive*. Most environments will have a mix of different ways that clients use to connect to the IdM domain. Administrators must decide which scenario is best for each individual client.

26.1.1.1. Initial Client Configuration (Pre-Migration)

Before deciding where you want to go with the client configuration in Identity Management, first establish where you are before the migration.

The initial state for almost all LDAP deployments that will be migrated is that there is an LDAP service providing identity and authentication services.

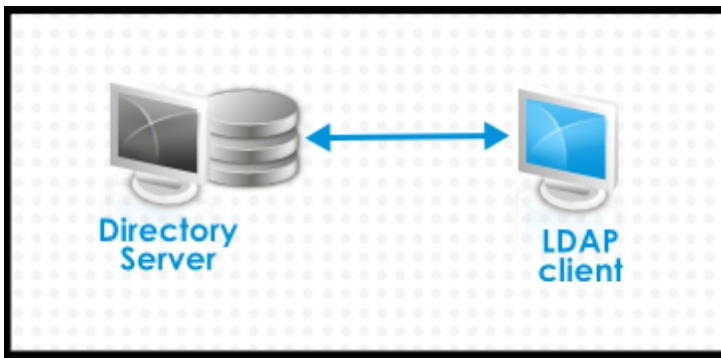


Figure 26.1. Basic LDAP Directory and Client Configuration

Linux and Unix clients use `PAM_LDAP` and `NSS_LDAP` libraries to connect directly to the LDAP services. These libraries allow clients to retrieve user information from the LDAP directory *as if* the data were stored in `/etc/passwd` or `/etc/shadow`. (In real life, the infrastructure may be more complex if a client uses LDAP for identity lookups and Kerberos for authentication or other configurations.)

There are structural differences between an LDAP directory and an IdM server, particularly in schema support and the structure of the directory tree. (For more background on those differences, see [Section 1.1, “IdM v. LDAP: A More Focused Type of Service”](#).) While those differences may impact data (especially with the directory tree, which affects entry names), they have little impact on the *client configuration*, so it really has little impact on migrating clients to Identity Management.

26.1.1.2. Recommended Configuration for Red Hat Enterprise Linux Clients

Red Hat Enterprise Linux has a service called the *System Security Services Daemon* (SSSD). SSSD uses special PAM and NSS libraries (`pam_sss` and `nss_sss`, respectively) which allow SSSD to be integrated very closely with Identity Management and leverage the full authentication and identity features in Identity Management. SSSD has a number of useful features, like caching identity information so that users can log in even if the connection is lost to the central server; these are described in the *Red Hat Enterprise Linux Deployment Guide*.

Unlike generic LDAP directory services (using `pam_ldap` and `nss_ldap`), SSSD establishes relationships between identity and authentication information by defining *domains*. A domain in SSSD defines four backend functions: authentication, identity lookups, access, and password changes. The SSSD domain is then configured to use a *provider* to supply the information for any one (or all) of those four functions. An identity provider is always required in the domain configuration. The other three providers are optional; if an authentication, access, or password provider is not defined, then the identity provider is used for that function.

SSSD can use Identity Management for all of its backend functions. This is the ideal configuration because it provides the full range of Identity Management functionality, unlike generic LDAP identity providers or Kerberos authentication. For example, during daily operation, SSSD enforces host-based access control rules and security features in Identity Management.



TIP

During the migration process from an LDAP directory to Identity Management, SSSD can seamlessly migrate user passwords without additional user interaction.

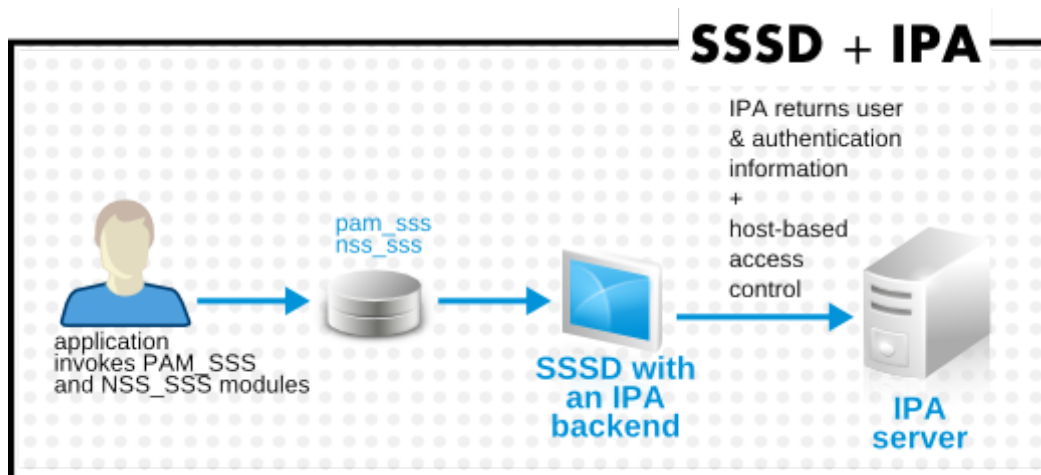


Figure 26.2. Clients and SSSD with an IdM Backend

The **ipa-client-install** script automatically configured SSSD to use IdM for all four of its backend services, so Red Hat Enterprise Linux clients are set up with the recommended configuration by default.



NOTE

This client configuration is only supported for Red Hat Enterprise Linux 6.1 and later and Red Hat Enterprise Linux 5.7 later, which support the latest versions of SSSD and **ipa-client**. Older versions of Red Hat Enterprise Linux can be configured as described in [Section 26.1.1.3, “Alternative Supported Configuration”](#).

26.1.1.3. Alternative Supported Configuration

Unix and Linux systems such as Mac, Solaris, HP-UX, AIX, and Scientific Linux support all of the services that IdM manages but do not use SSSD. Likewise, older Red Hat Enterprise Linux versions (6.1 and 5.6) support SSSD but have an older version, which does not support IdM as an identity provider.

When it is not possible to use a modern version of SSSD on a system, then clients can be configured to connect to the IdM server as if it were an LDAP directory service for identity lookups (using **nss_ldap**) and to IdM as if it were a regular Kerberos KDC (using **pam_krb5**).

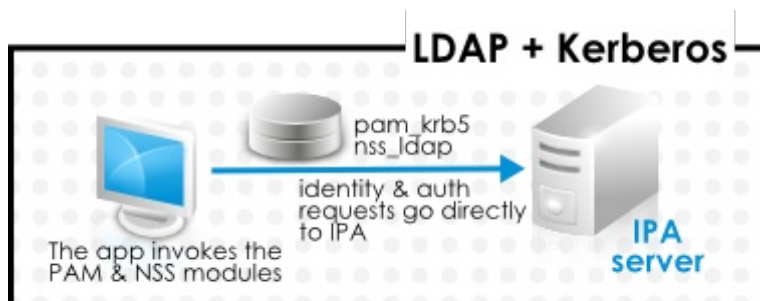


Figure 26.3. Clients and IdM with LDAP and Kerberos

If a Red Hat Enterprise Linux client is using an older version of SSSD, SSSD can still be configured to use the IdM server as its identity provider and its Kerberos authentication domain; this is described in

the SSSD configuration section of the *Red Hat Enterprise Linux Deployment Guide*.

Any IdM domain client can be configured to use **nss_ldap** and **pam_krb5** to connect to the IdM server. For some maintenance situations and IT structures, a scenario that fits the lowest common denominator may be required, using LDAP for both identity and authentication (**nss_ldap** and **pam_ldap**). However, it is generally best practice to use the most secure configuration possible for a client (meaning SSSD and Kerberos or LDAP and Kerberos).

26.1.2. Planning Password Migration

Probably the most visible issue that can impact LDAP-to-Identity Management migration is migrating user passwords.

Identity Management (by default) uses Kerberos for authentication and requires that each user have Kerberos hashes stored in the Identity Management Directory Server in addition to the standard user passwords. To generate these hashes, the user password needs to be available to the IdM server in cleartext. This is the case when the user is created in Identity Management. However, when the user is migrated from an LDAP directory, the associated user password is already hashed, so the corresponding Kerberos key cannot be generated.



IMPORTANT

Users cannot authenticate to the IdM domain or access IdM resources until they have Kerberos hashes.

If a user does not have a Kerberos hash [8], that user cannot log into the IdM domain even if he has a user account. There are three options for migrating passwords: forcing a password change, using a web page, and using SSSD.

Migrating users from an existing system provides a smoother transition but also requires parallel management of LDAP directory and IdM during the migration and transition process. If you do not preserve passwords, the migration can be performed more quickly but it requires more manual work by administrators and users.

26.1.2.1. Method 1: Using Temporary Passwords and Requiring a Change

When passwords are changed in Identity Management, they will be created with the appropriate Kerberos hashes. So one alternative for administrators is to force users to change their passwords by resetting all user passwords when user accounts are migrated. (This can also be done simply by re-creating the LDAP directory accounts in IdM, which automatically creates accounts with the appropriate keys.) The new users are assigned a temporary password which they change at the first login. No passwords are migrated.

26.1.2.2. Method 2: Using the Migration Web Page

When it is running in migration mode, Identity Management has a special web page in its web UI that will capture a cleartext password and create the appropriate Kerberos hash.

<https://ipaserver.example.com/ipa/migration>

Administrators could tell users to authenticate once to this web page, which would properly update their user accounts with their password and corresponding Kerberos hash, without requiring password changes.

26.1.2.3. Method 3: Using SSSD (Recommended)

SSSD can work with IdM to mitigate the user impact on migrating by generating the required user keys. For deployments with a lot of users or where users shouldn't be burdened with password changes, this is the best scenario.

1. A user tries to log into a machine with SSSD.
2. SSSD attempts to perform Kerberos authentication against the IdM server.
3. Even though the user exists in the system, the authentication will fail with the error *key type is not supported* because the Kerberos hashes do not yet exist.
4. SSSD then performs a plaintext LDAP bind over a secure connection.
5. IdM intercepts this bind request. If the user has a Kerberos principal but no Kerberos hashes, then the IdM identity provider generates the hashes and stores them in the user entry.
6. If authentication is successful, SSSD disconnects from IdM and tries Kerberos authentication again. This time, the request succeeds because the hash exists in the entry.

That entire process is entirely transparent to the user; as far as users know, they simply log into a client service and it works as normal.

26.1.2.4. Migrating Cleartext LDAP Passwords

Although in most deployments LDAP passwords are stored encrypted, there may be some users or some environments that use cleartext passwords for user entries.

When users are migrated from the LDAP server to the IdM server, their cleartext passwords are not migrated over. Identity Management does not allow cleartext passwords. Instead, a Kerberos principle is created for the user, the keytab is set to true, and the password is set as expired. This means that Identity Management requires the user to reset the password at the next login.



NOTE

If passwords are hashed, the password is successfully migrated through SSSD and the migration web page, as in [Section 26.1.2.3, “Method 3: Using SSSD \(Recommended\)”](#).

26.1.2.5. Automatically Resetting Passwords That Do Not Meet Requirements

If user passwords in the original directory do not meet the password policies defined in Identity Management, then the passwords must be reset after migration.

Password resets are done automatically the first time the users attempts to **kinit** into the IdM domain.

```
[jsmith@server ~]$ kinit
Password for jsmith@EXAMPLE.COM:
Password expired. You must change it now.
Enter new password:
Enter it again:
```

26.1.3. Migration Considerations and Requirements

As you are planning migrating from an LDAP server to Identity Management, make sure that your LDAP environment is able to work with the Identity Management migration script.

26.1.3.1. LDAP Servers Supported for Migration

The migration process from an LDAP server to Identity Management uses a special script, **ipa**

ipa migrate-ds, to perform the migration. This script has certain expectations about the structure of the LDAP directory and LDAP entries in order to work. Migration is supported only for LDAPv3-compliant directory services, which include several common directories:

- SunONE Directory Server
- Apache Directory Server
- OpenLDAP

Migration from an LDAP server to Identity Management has been tested with Red Hat Directory Server.



NOTE

Migration using the migration script is *not* supported for Microsoft Active Directory because it is not an LDAPv3-compliant directory. For assistance with migrating from Active Directory, contact Red Hat Professional Services.

26.1.3.2. Migration Environment Requirements

There are many different possible configuration scenarios for both Red Hat Directory Server and Identity Management, and any of those scenarios may affect the migration process. For the example migration procedures in this chapter, these are the assumptions about the environment:

- A single LDAP directory domain is being migrated to one IdM realm. No consolidation is involved.
- User passwords are stored as a hash in the LDAP directory that the IdM Directory Server can support.
- The LDAP directory instance is both the identity store and the authentication method. Client machines are configured to use **pam_ldap** or **nss_ldap** to connect to the LDAP server.
- Entries use only standard LDAP schema. Custom attributes will not be migrated to Identity Management.

26.1.3.3. Migration Tools

Identity Management uses a specific command, **ipa migrate-ds**, to drive the migration process so that LDAP directory data are properly formatted and imported cleanly into the IdM server.

The Identity Management server must be configured to run in migration mode, and then the migration script can be used.

26.1.3.4. Migration Sequence

There are four major steps when migrating to Identity Management, but the order varies slightly depending on whether you want to migrate the server first or the clients first.

With a client-based migration, SSSD is used to change the client configuration while an IdM server is configured:

1. Deploy SSSD.
2. Reconfigure clients to connect to the current LDAP server and then fail over to IdM.
3. Install the IdM server.
4. Migrate the user data using the IdM **ipa-migrate-ds** script. This exports the data from the LDAP directory, formats for the IdM schema, and then imports it into IdM.
5. Take the LDAP server offline and allow clients to fail over to Identity Management transparently.

With a server migration, the LDAP to Identity Management migration comes first:

1. Install the IdM server.
2. Migrate the user data using the IdM **ipa-migrate-ds** script. This exports the data from the LDAP directory, formats it for the IdM schema, and then imports it into IdM.
3. *Optional.* Deploy SSSD.
4. Reconfigure clients to connect to IdM. It is not possible to simply replace the LDAP server. The IdM directory tree — and therefore user entry DNs — is different than the previous directory tree. While it is required that clients be reconfigured, clients do not need to be reconfigured immediately. Updated clients can point to the IdM server while other clients point to the old LDAP directory, allowing a reasonable testing and transition phase after the data are migrated.



NOTE

Do not run both an LDAP directory service and the IdM server for very long in parallel. This introduces the risk of user data being inconsistent between the two services.

Both processes provide a general migration procedure, but it may not work in every environment. Set up a test LDAP environment and test the migration process before attempting to migrate the real LDAP environment.

26.2. Examples for Using **migrate-ds**

The data migration is performed with the **ipa migrate-ds** command. At its simplest, the command takes the LDAP URL of the directory to migrate and exports the data based on common default settings.

```
ipa migrate-ds ldap://ldap.example.com:389
```

It is possible to customize how the **migrate-ds** commands identifies and exports data. This is useful if the original directory tree has a unique structure or if some entries or attributes within entries should be excluded from migration.

26.2.1. Migrating Specific Subtrees

The default directory structure places person entries in the **ou=People** subtree and group entries in the **ou=Groups** subtree. These subtrees are container entries for those different types of directory data. If no options are passed with the **migrate-ds** command, then the utility assumes that the given LDAP directory uses the **ou=People** and **ou=Groups** structure.

Many deployments may have an entirely different directory structure (or may only want to export certain parts of the directory tree). There are two options which allow administrators to give the RDN of a different user or group subtree:

- **--user-container**
- **--group-container**

**NOTE**

In both cases, the subtree must be the RDN only and must be relative to the base DN. For example, the **ou=Employees,dc=example,dc=com** subtree can be migrated using **--user-container=ou=Employees**, but **ou=Employees,ou=People,dc=example,dc=com** cannot be migrated with that option because **ou=Employees** is not a direct child of the base DN.

For example:

```
[root@ipaserver ~]# ipa migrate-ds --user-container=ou=employees --group-
container="ou=employee groups" ldap://ldap.example.com:389
```

There is a third option that allows administrators to set a base DN for migration: **--base-dn**. With this option, it is possible to change the target for container subtrees. For example:

```
[root@ipaserver ~]# ipa migrate-ds --user-container=ou=employees --base-
dn="ou=people,dc=example,dc=com" ldap://ldap.example.com:389
```

Now, the **ou=Employees** user subtree can be migrated from within the larger **ou=People** subtree without migrating every people-related subtree.

26.2.2. Specifically Including or Excluding Entries

By default, the **migrate-ds** script exports every user entry with the **person** object class and every group entry within the given user and group subtrees.

In some migration paths, only specific types of users and groups may need to be exported, or, conversely, specific users and groups may need to be excluded.

One option is to set positively which *types* of users and groups to include. This is done by setting which object classes to search for when looking for user or group entries.

This is a really useful option when there are custom object classes used in an environment for different user types. For example, this migrates only users with the custom **fullTimeEmployee** object class:

```
[root@ipaserver ~]# ipa migrate-ds --user-objectclass=fullTimeEmployee
ldap://ldap.example.com:389
```

Because of the different types of groups, this is also very useful for migrating only certain types of groups (such as user groups) while excluding other types of groups, like certificate groups. For example:

```
[root@ipaserver ~]# ipa migrate-ds --group-
objectclass=groupOfNames,groupOfUniqueNames ldap://ldap.example.com:389
```

Positively specifying user and groups to migrate based on object class implicitly excludes all other users and groups from migration.

Alternatively, it can be useful to migrate all user and group entries except for just a small handful of entries. Specific user or group accounts can be excluded while all others of that type are migrated. For example, this excludes a hobbies group and two users:

```
[root@ipaserver ~]# ipa migrate-ds --exclude-groups="Golfers Group" --exclude-users=jsmith,bjensen ldap://ldap.example.com:389
```

Specifying an object class to migrate can be used together with excluding specific entries. For example, this specifically includes users with the **fullTimeEmployee** object class, yet excludes three managers:

```
[root@ipaserver ~]# ipa migrate-ds --user-objectclass=fullTimeEmployee --exclude-users=jsmith,bjensen,mreynolds ldap://ldap.example.com:389
```

26.2.3. Excluding Entry Attributes

By default, every attribute and object class for a user or group entry is migrated. There are some cases where that may not be realistic, either because of bandwidth and network constraints or because the attribute data are no longer relevant. For example, if users are going to be assigned new user certificates as they join the IdM domain, then there is no reason to migrate the **userCertificate** attribute.

Specific object classes and attributes can be ignored by the **migrate-ds** by using any of several different options:

- ▶ **--user-ignore-objectclass**
- ▶ **--user-ignore-attribute**
- ▶ **--group-ignore-objectclass**
- ▶ **--group-ignore-attribute**

For example, to exclude the **userCertificate** attribute and **strongAuthenticationUser** object class for users and the **groupOfCertificates** object class for groups:

```
[root@ipaserver ~]# ipa migrate-ds --user-ignore-attribute=userCertificate --user-ignore-objectclass=strongAuthenticationUser --group-ignore-objectclass=groupOfCertificates ldap://ldap.example.com:389
```



NOTE

Make sure not to ignore any required attributes. Also, when excluding object classes, make sure to exclude any attributes which are only supported by that object class.

26.2.4. Setting the Schema to Use

By default, Identity Management uses [RFC2307bis](#) schema to define user, host, hostgroup, and other network identities. This schema option can be reset to use [RFC2307](#) schema instead:

```
[root@ipaserver ~]# ipa migrate-ds --schema=RFC2307 ldap://ldap.example.com:389
```

26.3. Scenario 1: Using SSSD as Part of Migration

**IMPORTANT**

This is a general migration procedure, but it may not work in every environment. It is strongly recommended that you set up a test LDAP environment and test the migration process before attempting to migrate the real LDAP environment.

1. Set up SSSD. Using SSSD allows the required Kerberos keys and server certificates to be delivered to the clients.

- a. Install SSSD on every client machine:

```
# yum install sssd
```

- b. Configure an LDAP identity provider in SSSD to use the existing Directory Server for all functions (authentication, identity lookups, access, and password changes). This ensures every client works properly with the existing directory service.

2. Install Identity Management, including any custom LDAP directory schema ^[9], on a different machine from the existing LDAP directory.
3. Enable the IdM server to allow migration:

```
# ipa config-mod --enable-migration=TRUE
```

4. Disable the compat plug-in.

```
# ipa-compat-manage disable
```

5. Restart the IdM Directory Server instance.

```
# service dirsrv restart
```

6. Run the IdM migration script, **ipa migrate-ds**. At its most basic, this requires only the LDAP URL of the LDAP directory instance to migrate:

```
# ipa migrate-ds ldap://ldap.example.com:389
```

Simply passing the LDAP URL migrates all of the directory data using common default settings. The user and group data can be selectively migrated by specifying other options, as covered in [Section 26.2, “Examples for Using migrate-ds”](#).

Once the information is exported, the script adds all required IdM object classes and attributes and converts DN's in attributes to match the IdM directory tree.

7. Re-enable the compat plug-in.

```
# ipa-compat-manage enable
```

8. Restart the IdM Directory Server instance.

```
# service dirsrv restart
```

9. Move clients that have SSSD installed from the LDAP backend to the Identity Management backend and enroll them as client with IdM. This downloads the required keys and certificates. On Red Hat Enterprise Linux clients, this can be done using the **ipa-client-install**

command. For example:

```
# ipa-client-install --enable-dns-updates
```

10. Have users log into a machine with SSSD and Identity Management backend. This generates the required Kerberos keys for the user.

To monitor the user migration process, query the existing LDAP directory to see which user accounts have a password but do not yet have a Kerberos principal key.

```
$ ldapsearch -LL -x -D 'cn=Directory Manager' -w secret -b  
'ou=people,dc=example,dc=com' '(&(!krbprincipalkey=*)(userpassword=*))' uid
```



NOTE

Include the quotes around the filter so that it is not interpreted by the shell.

11. Once users have been migrated over, configure non-SSSD clients to use the IdM domain, as required.
12. When the migration of all clients and users is complete, decommission the LDAP directory.

26.4. Scenario 2: Migrating an LDAP Server Directly to Identity Management



IMPORTANT

This is a general migration procedure, but it may not work in every environment. It is strongly recommended that you set up a test LDAP environment and test the migration process before attempting to migrate the real LDAP environment.

1. Install the IdM server, including any custom LDAP directory schema ^[10], on a different machine from the existing LDAP directory.
2. Disable the compat plug-in.

```
# ipa-compat-manage disable
```

3. Restart the IdM Directory Server instance.

```
# service dirsrv restart
```

4. Enable the IdM server to allow migration:

```
# ipa config-mod --enable-migration=TRUE
```

5. Run the IdM migration script, **ipa migrate-ds**. At its most basic, this requires only the LDAP URL of the LDAP directory instance to migrate:

```
# ipa migrate-ds ldap://ldap.example.com:389
```

Simply passing the LDAP URL migrates all of the directory data using common default settings.

The user and group data can be selectively migrated by specifying other options, as covered in [Section 26.2, “Examples for Using migrate-ds”](#).

Once the information is exported, the script adds all required IdM object classes and attributes and converts DN's in attributes to match the IdM directory tree.

6. Re-enable the compat plug-in.

```
# ipa-compat-manage enable
```

7. Restart the IdM Directory Server instance.

```
# service dirsrv restart
```

8. Update the client configuration to use PAM_LDAP and NSS_LDAP to connect to IdM instead of connecting to an LDAP directory, NIS, or local files.
9. *Optional.* Set up SSSD. Using SSSD migrates user passwords without additional user interaction, as described in [Section 26.1.2, “Planning Password Migration”](#).
 - a. Install SSSD on every client machine:

```
# yum install sssd
```

- b. Run the **ipa-client-install** to configure SSSD and related services to use the IdM server for identity and Kerberos authentication.

10. Instruct users to log into IdM using either SSSD client or the migration web page if SSSD is not available on the client. Both methods automatically migrate the user password into Identity Management.

```
https://ipaserver.example.com/ipa/migration
```

11. *Optional.* Reconfigure non-SSSD clients to use Kerberos authentication (**pam_krb5**) instead of LDAP authentication (**pam_ldap**).



NOTE

Use PAM_LDAP modules until all of the users have been migrated; then it is possible to use PAM_KRB5.

12. When the migration of all clients and users is complete, decommission the LDAP directory.

[8] It is possible to use LDAP authentication in Identity Management instead of Kerberos authentication, which means that Kerberos hashes are not required for users. However, this limits the capabilities of Identity Management and is not recommended.

[9] There is limited support for custom user and group schema in Identity Management.

[10] There is limited support for custom user and group schema in Identity Management.

Troubleshooting Identity Management

A.1. Installation Issues

A.1.1. Server Installation

The server installation log is located in `/var/log/ipaserver-install.log`. The IdM logs, both for the server and for IdM-associated services, are covered in [Section 25.1.3, “Checking IdM Server Logs”](#).

A.1.1.1. GSS Failures When Running IPA Commands

Immediately after installation, there can be Kerberos problems when trying to run an `ipa-*` command. For example:

```
ipa: ERROR: Kerberos error: ('Unspecified GSS failure. Minor code may provide more information', 851968)/('Decrypt integrity check failed', -1765328353)
```

There are two potential causes for this:

- DNS is not properly configured.
- Active Directory is in the same domain as the IdM server.

A.1.1.2. named Daemon Fails to Start

If an IdM server is configured to manage DNS and is set up successfully, but the **named** service fails to start, this can indicate that there is a package conflict. Check the `/var/log/messages` file for error messages related to the **named** service and the **ldap.so** library:

```
ipaserver named[6886]: failed to dynamically load driver 'ldap.so': libldap-2.4.so.2: cannot open shared object file: No such file or directory
```

This usually means that the *bind-chroot* package is installed and is preventing the **named** service from starting. To resolve this issue, remove the *bind-chroot* package and then restart the IdM server.

```
[root@server ~]# yum remove bind-chroot
# ipactl restart
```

A.1.2. Replica Installation

A.1.2.1. Certificate System setup failed.

If the replica installation fails on step 3 ([3/11]: **configuring certificate server instance**), that usually means that the required port is not available. This can be verified by checking the debug logs for the CA, `/var/log/pki-ca/debug`, which may show error messages about being unable to find certain entries. For example:

```
[04/Feb/2013:22:29:03][http-9445-Processor25]: DatabasePanel
compareAndWaitEntries ou=people,o=ipaca not found, let's wait
```

The only resolution is to uninstall the replica:

```
[root@ipareplica ~]# ipa-server-install --uninstall
```

After uninstalling the replica, ensure that port 7389 on the replica is available, and retry the replica installation.

A.1.2.2. There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.

When the replica starts, there can be a series of SASL bind errors recorded in the 389 Directory Server logs stating that the GSS-API connection failed because it could not find a credentials cache:

```
slapd_ldap_sasl_interactive_bind - Error: could not perform interactive bind for
id [] mech [GSSAPI]: error -2 (Local error) (SASL(-1): generic failure: GSSAPI
Error: Unspecified GSS failure. Minor code may provide more information
(Credentials cache file '/tmp/krb5cc_496' not found)) ...
```

The replica is looking for a credentials cache in **/tmp/krb5cc_496** (where 496 is the 389 Directory Server user ID) and cannot find it.

There may also be messages that the server could not obtain Kerberos credentials for the host principal:

```
set_krb5_creds - Could not get initial credentials for principal [ldap/
replica1.example.com] in keytab [WRFIL:/etc/dirsrv/ds.keytab]: -1765328324
(Generic error)
```

These errors are both related to how and when the 389 Directory Server instance loads its Kerberos credentials cache.

While 389 Directory Server itself supports multiple different authentication mechanisms, Identity Management only uses GSS-API for Kerberos connections. The 389 Directory Server instance for Identity Management keeps its Kerberos credentials cache in memory. When the 389 Directory Server process ends — like when the IdM replica is stopped — the credentials cache is destroyed.

Also, the 389 Directory Server is used as the backend storage for the principal information for the KDC.

When the replica then restarts, the 389 Directory Server instance starts first, since it supplies information for the KDC, and then the KDC server starts. This start order is what causes the GSS-API and Kerberos connection errors.

The 389 Directory Server attempts to open a GSS-API connection, but since there is no credentials cache yet and the KDC is not started, the GSS connection fails. Likewise, any attempt to obtain the host credentials also fails.

These errors are transient. The 389 Directory Server re-attempts the GSS-API connection after the KDC starts and it has a credentials cache. The 389 Directory Server logs then record a **bind resumed** message.

These startup GSS-API connection failures can be ignored as long as that connection is successfully established.

A.1.2.3. The DNS forward record does not match the reverse address

When configuring a new replica, installation can fail with a series of certificate errors and, ultimately an error that the DNS forward and reverse records do not match.

```

ipa: DEBUG: approved_usage = SSLServer intended_usage = SSLServer
ipa: DEBUG: cert valid True for "CN=ipa-server2.example.com,O=EXAMPLE.COM"
ipa: DEBUG: handshake complete, peer = 192.168.17.37:9444
Certificate operation cannot be completed: Unable to communicate with CMS (Not
Found)

...

ipa: DEBUG: Created connection context.ldap2_21534032
ipa: DEBUG: Destroyed connection context.ldap2_21534032
The DNS forward record ipa-server2.example.com. does not match the reverse address
ipa-server2.example.org

```

The hostname for every server and replica in the IdM domain must be fully resolvable for both DNS forward (A) and reverse (PTR) records. Both forward and reverse records are checked during authentication and certificate-related operations. If the hostnames in the records do not match, then both certificate errors and DNS errors are returned.

This problem can occur if multiple hostnames are used for a single PTR record. This is allowed in the DNS standard, but it creates problems during IdM replica creation when it attempts to configure services.

Ensure the primary hostname for the replica host is the only one returned for PTR lookups and remove any duplicate or additional hostnames.

Verifying the DNS A and PTR records is covered in [Section 2.5.1, “DNS Records”](#).

A.1.3. Client Installations

For clients configured using **ipa-client-install**, the client installation log is located in **/var/log/ipaclient-install.log**. The IdM logs, both for the server and client and for IdM-associated services, are covered in [Section 25.1.3, “Checking IdM Server Logs”](#).

These are some issues and workarounds for client installation problems.

A.1.3.1. The client can't resolve reverse hostnames when using an external DNS.

While IdM can host its own DNS server as part of the domain services, it can also use external DNS name server. However, because of some of the limitations of reverse DNS, there can be problems with resolving reverse lookups if the external DNS is listed in the client's **/etc/resolv.conf** file or if there are other resources on the network with SRV records, like Active Directory.

The problem is that the external DNS name server returns the wrong hostname for the IdM server.

One way this exhibits is errors with finding the IdM server in the Kerberos database:

```

Jun 30 11:11:48 server1 krb5kdc[1279](info): AS_REQ (4 etypes {18 17 16 23})
192.168.60.135: NEEDED_PREAUTH: admin EXAMPLE COM for krbtgt/EXAMPLE COM EXAMPLE
COM, Additional pre-authentication required
Jun 30 11:11:48 server1 krb5kdc[1279](info): AS_REQ (4 etypes {18 17 16 23})
192.168.60.135: ISSUE: authtime 1309425108, etypes {rep=18 tkt=18 ses=18}, admin
EXAMPLE COM for krbtgt/EXAMPLE COM EXAMPLE COM
Jun 30 11:11:49 server1 krb5kdc[1279](info): TGS_REQ (4 etypes {18 17 16 23})
192.168.60.135: UNKNOWN_SERVER: authtime 0, admin EXAMPLE COM for
HTTP/server1.wrong.example.com@EXAMPLE.COM, Server not found in Kerberos database

```

There are several ways to work around this issue:

- Edit the **/etc/resolv.conf** file to remove the external DNS name server references.

- Add reverse lookup records for each IdM server.
- Give the IdM client or domain a subnet and forward all requests for that subnet.

A.1.3.2. The client is not added to the DNS zone.

If a client is in a subnet not controlled by an IdM DNS server, then the **nsupdate** command may fail to add the client to the DNS zone when **ipa-client-install** runs.

If IdM is managing the DNS domain, then add a zone entry for the client manually, as described in [Section 16.8, “Managing DNS Record Entries”](#). For example:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa dnsrecord-add ipaclient.example.com www --a-rec 1.2.3.4
```

If the DNS domain is managed outside of IdM, the resource record can be added manually to the zone configuration. For information on DNS in Red Hat Enterprise Linux, see [the DNS chapter in the Deployment Guide](#).

A.1.4. Uninstalling an IdM Client

For Red Hat Enterprise Linux clients, the **ipa-client-install** utility can be used to uninstall the client and remove it from the IdM domain. To remove the client, use the **--uninstall** option.

```
# ipa-client-install --uninstall
```



NOTE

There is an uninstall option with the **ipa-join** command. This is called by **ipa-client-install --uninstall** as part of the uninstallation process. However, while the **ipa-join** option removes the client from the domain, it does not actually uninstall the client or properly remove all of the IdM-related configuration. Do not run **ipa-join -u** to attempt to uninstall the IdM client. The only way to uninstall a client completely is to use **ipa-client-install --uninstall**.

A.2. UI Connection Problems

If negotiate authentication is not working, turn on verbose logging for the authentication process to help diagnose the issue:

1. Close all browser windows.
2. In a terminal, set the new log levels for Firefox:

```
export NSPR_LOG_MODULES=authenticate:5
export NSPR_LOG_FILE=/tmp/moz.log
```

This enables verbose logging and logs all information to **/tmp/moz.log**.

3. Restart the browser from the same terminal window and attempt t .

Some of the common error messages and workarounds are in [Table A.1, “UI Error Log Messages”](#).

Table A.1. UI Error Log Messages

Error Log Message	Description and Fix
<pre>-1208550944[90039d0]: entering nsNegotiateAuth::GetNextToken() -1208550944[90039d0]: gss_init_sec_context() failed: Miscellaneous failure No credentials cache found</pre>	There are no Kerberos tickets. Run kinit.
<pre>-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken() -1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure Server not found in Kerberos database</pre>	This can occur when you have successfully obtained Kerberos tickets but are still unable to authenticate to the UI. This indicates that there is a problem with the Kerberos configuration. The first place to check is the [domain_realm] section in the /etc/krb5.conf file. Make sure that the IdM Kerberos domain entry is correct and matches the configuration in the Firefox negotiation parameters. For example: <pre>.example.com = EXAMPLE.COM example.com = EXAMPLE.COM</pre>
Nothing is in the log file.	It is possible that you are behind a proxy which is removing the HTTP headers required for negotiate authentication. Try to connect to the server using HTTPS instead, which allows the request to pass through unmodified. Then check the log file again.

A.3. IdM Server Problems

A.3.1. There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.

When the replica starts, there can be a series of SASL bind errors recorded in the 389 Directory Server logs stating that the GSS-API connection failed because it could not find a credentials cache:

```
slapd_ldap_sasl_interactive_bind - Error: could not perform interactive bind for
id [] mech [GSSAPI]: error -2 (Local error) (SASL(-1): generic failure: GSSAPI
Error: Unspecified GSS failure. Minor code may provide more information
(Credentials cache file '/tmp/krb5cc_496' not found)) ...
```

The replica is looking for a credentials cache in **/tmp/krb5cc_496** (where 496 is the 389 Directory Server user ID) and cannot find it.

There may also be messages that the server could not obtain Kerberos credentials for the host principal:

```
set_krb5_creds - Could not get initial credentials for principal [ldap/
replica1.example.com] in keytab [WRFIL:/etc/dirsrv/ds.keytab]: -1765328324
(Generic error)
```

These errors are both related to how and when the 389 Directory Server instance loads its Kerberos credentials cache.

While 389 Directory Server itself supports multiple different authentication mechanisms, Identity Management only uses GSS-API for Kerberos connections. The 389 Directory Server instance for Identity Management keeps its Kerberos credentials cache in memory. When the 389 Directory Server process ends — like when the IdM replica is stopped — the credentials cache is destroyed.

Also, the 389 Directory Server is used as the backend storage for the principal information for the KDC.

When the replica then restarts, the 389 Directory Server instance starts first, since it supplies information for the KDC, and then the KDC server starts. This start order is what causes the GSS-API and Kerberos connection errors.

The 389 Directory Server attempts to open a GSS-API connection, but since there is no credentials cache yet and the KDC is not started, the GSS connection fails. Likewise, any attempt to obtain the host credentials also fails.

These errors are transient. The 389 Directory Server re-attempts the GSS-API connection after the KDC starts and it has a credentials cache. The 389 Directory Server logs then record a **bind resumed** message.

These startup GSS-API connection failures can be ignored as long as that connection is successfully established.

A.4. Host Problems

A.4.1. Certificate Not Found/Serial Number Not Found Errors

The IdM information is stored in a separate LDAP directory than the certificate information, and these two LDAP databases are replicated separately. It is possible for a replication agreement to be broken for one directory and working for another, which can cause problems with managing clients.

Specifically, if the replication agreement between the two CA databases is broken, then a server may not be able to find certificate information about a valid IdM client, causing certificate errors:

```
Certificate operation cannot be completed: EXCEPTION (Certificate serial number
0x2d not found)
```

For example, an IdM server and replica have a function replication agreement between their IdM databases, but the replication agreement between their CA databases is broken. If a host is created on the server, the host entry is replicated over to the replica — but the certificate for that host is not replicated. The replica is aware of the client, but any management operations for that client will fail because the replica doesn't have a copy of its certificate.

A.4.2. Debugging Client Connection Problems

Client connection problems are apparent immediately. This can mean that users cannot log into a machine or attempts to access user and group information fails (for example, **getent passwd admin**).

Authentication in IdM is managed with the SSSD daemon, which is described in the *Red Hat Enterprise*

Linux Deployment Guide. If there are problems with client authentication, then check the SSSD information.

First, check the SSSD logs in `/var/log/sss/`. There is a specific log file for the DNS domain, such as `sss_example.com.log`. If there is not enough information in the logs at the default logging level, then increase the log level.

To increase the log level:

1. Open the `sss.conf` file.

```
vim /etc/sss/sss.conf
```

2. In the `[domain/example.com]` section, set `debug_level`.

```
debug_level = 9
```

3. Restart the `sss` daemon.

```
service sss restart
```

4. Check the `/var/log/sss/sss_example.com.log` file for the debug messages.

A.5. Kerberos Errors

Kerberos errors frequently become apparent when trying to connect to the realm using `kinit` or a similar client. For information related to Kerberos, first check the Kerberos manpages, help files, and other resources.



IMPORTANT

Identity Management has its own command-line tools to use to manage Kerberos policies. **Do not** use `kadmin` or `kadmin.local` to manage IdM Kerberos settings.

There are several places to look for Kerberos error log information:

- For `kinit` problems or other Kerberos server problems, look at the KDC log in `/var/log/krb5kdc.log`.
- For IdM-specific errors, look in `/var/log/httpd/error_log`.

The IdM logs, both for the server and for IdM-associated services, are covered in [Section 25.1.3, “Checking IdM Server Logs”](#).

A.5.1. Problems making connections with SSH when using GSS-API

If there are bad reverse DNS entries in the DNS configuration, then it may not be possible to log into IdM resources using SSH. When SSH attempts to connect to a resource using GSS-API as its security method, GSS-API first checks the DNS records. The bad records prevent SSH from locating the resource.

It is possible to disable reverse DNS lookups in the SSH configuration. Rather than using reverse DNS records, SSH passes the given username directly to GSS-API.

To disable reverse DNS lookups with SSH, add or edit the **GSSAPITrustDNS** directive and set the value to **no**.

```
# vim /etc/ssh/ssh_config  
  
GSSAPITrustDNS no
```

A.5.2. There are problems connecting to an NFS server after changing a keytab

Clients attempting to mount NFS exports rely on the existence of a valid principal and secret key on both the NFS server and the client host. Clients themselves should not have access to the NFS keytab. The ticket for the NFS connection will be given to clients from the KDC.

Failure to export an updated keytab can cause problems that are difficult to isolate. For example, existing service connections may continue to function, but no new connections may be possible.

A.6. SELinux Login Problems

SELinux maps only work for remote users, not for users with a local account.

When a remote user logs in, authenticating against the IdM server, then the PAM SELinux modules create a file for that user in **/etc/selinux/policy_name/logins/login**.

If that file does not exist, then it means that SSSD is not properly configured to use the IdM server as one of its identity providers. This is required for SELinux mapping to work. Configuring SSSD is covered in the [Red Hat 6 Deployment Guide](#).

If the file exists but the remote user was given the wrong SELinux context, then the **pam_selinux** module may not be properly configured in the PAM stack. This is the module that reads the SELinux information and sets the user context. If the module is missing, then nothing processes the SELinux map and the user is defined a default context on the system.

Working with certmonger

Part of managing machine authentication is managing machine certificates. On clients, IdM manages the certificate lifecycle with the *certmonger* service, which works together with the certificate authority (CA) provided by IdM.

The **certmonger** daemon and its command-line clients simplify the process of generating public/private key pairs, creating certificate requests, and submitting requests to the CA for signing. As part of managing certificates, the **certmonger** daemon monitors certificates for expiration and can renew certificates that are about to expire. The certificates that **certmonger** monitors are tracked in files stored in a configurable directory. The default location is `/var/lib/certmonger/requests`.

certmonger uses the IdM **getcert** command to manage all certificates. As covered in [Section 3.4, “Examples: Installing with Different CA Configurations”](#), an IdM server can be configured to use different types of certificate authorities. The most common (and recommended) configuration is to use a full CA server, but it is also possible to use a much more limited, self-signed CA. The exact **getcert** command used by **certmonger** to communicate with the IdM backend depends on which type of CA is used. The **ipa-getcert** command is used with a full CA, while the **selfsign-getcert** command is used with a self-signed CA.



NOTE

Because of general security issues, self-signed certificates are not typically used in production, but can be used for development and testing.

B.1. Requesting a Certificate with certmonger

With the IdM CA, **certmonger** uses the **ipa-getcert** command.

Certificates and keys are stored locally in plaintext files (**.pem**) or in an NSS database, identified by the certificate nickname. When requesting a certificate, then, the request should identify the location where the certificate will be stored and the nickname of the certificate. For example:

```
# ipa-getcert request -d /etc/pki/nssdb -n Server-Cert
```

The `/etc/pki/nssdb` file is the global NSS database, and **Server-Cert** is the nickname of this certificate. The certificate nickname must be unique within this database.

When requesting a certificate to be used with an IdM service, the **-K** option is required to specify the service principal. Otherwise, **certmonger** assumes the certificate is for a host. The **-N** option must specify the certificate subject DN, and the subject base DN must match the base DN for the IdM server, or the request is rejected.

```
$ ipa-getcert request -d /etc/httpd/alias -n Server-Cert -K
HTTP/client1.example.com -N 'CN=client1.example.com,O=EXAMPLE.COM'
```

Example B.1. Using certmonger for a Service

```
$ ipa-getcert request -r -f /etc/httpd/conf/ssl.crt/server.crt -k
/etc/httpd/conf/ssl.key/server.key -N CN=`hostname` --fqdn` -D `hostname` -U id-
kp-serverAuth
```

The options vary depending on whether you are using a self-signed certificate (**selfsign-getcert**) and the desired configuration for the final certificate, as well as other settings. In [Example B.1, “Using certmonger for a Service”](#), these are common options:

- ▶ The **-r** option will automatically renew the certificate if the key pair already exists. This is used by default.
- ▶ The **-f** option stores the certificate in the given file.
- ▶ The **-k** option either stores the key in the given file or, if the key file already exists, uses the key in the file.
- ▶ The **-N** option gives the subject name.
- ▶ The **-D** option gives the DNS domain name.
- ▶ The **-U** option sets the extended key usage flag.

B.2. Storing Certificates in NSS Databases

By default, **certmonger** uses plaintext files to store the key and the certificate, but these keys and certificates can also be stored in NSS databases. This is done using the **-d** option to set the security database location and **-n** to give the certificate nickname which is used for the certificate in the database. These options are used instead of the PEM files given in the **-f** and **-k** options.

For example:

```
# ipa-getcert request -d /export/alias -n ServerCert ...
```

B.3. Tracking Certificates with certmonger

certmonger can manage the entire certificate lifecycle. Along with generating requests, **certmonger** can track a certificate and automatically renew it when it expires at the end of its validity period.

This is done using the **start-tracking** command with the **getcert** command. The **-I** option creates the tracking entry, along with pointers to the key and certificate files, either in an NSS database (**-d** and **-n**) or in the PEM file (**-f** and **-k**). The **-r** option tells **certmonger** to renew the certificate.

```
# ipa-getcert start-tracking -I cert1-tracker -d /export/alias -n ServerCert -r
```



TIP

The **-r** option can be passed with the **request** command, in [Example B.1, “Using certmonger for a Service”](#). In that case, the requested certificate is automatically tracked and renewed by **certmonger**. Then, it is not necessary to configure tracking manually.

A certificate can be *untracked* by **certmonger** by using the **stop-tracking** command.

Glossary

A

access control instruction

See [ACI](#).

access control list

See [ACL](#).

access rights

In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

account inactivation

Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

ACI

An instruction that grants or denies permissions to entries in the directory.

See Also [access control instruction](#).

ACL

The mechanism for controlling access to your directory.

See Also [access control list](#).

All IDs Threshold

Replaced with the ID list scan limit in Directory Server version 7.1. A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

See Also [ID list scan limit](#).

All IDs token

A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access

When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index

Allows for efficient approximate or "sounds-like" searches.

attribute

Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list

A list of required and optional attributes for a given entry type or object class.

authenticating directory server

In pass-through authentication (PTA), the authenticating Directory Server is the Directory Server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.

authentication

(1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a [client](#) to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

authentication certificate

Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

B**base distinguished name**

See [base DN](#).

base DN

Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.

bind distinguished name

See [bind DN](#).

bind DN

Distinguished name used to authenticate to Directory Server when performing an operation.

bind rule

In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

branch entry

An entry that represents the top of a subtree in the directory.

browser

Software, such as Mozilla Firefox, used to request and view World Wide Web material stored as HTML files. The browser uses the HTTP protocol to communicate with the host server.

browsing index

Speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branch point in the directory tree to improve display performance.

See Also [virtual list view index](#) .

C

CA

See [Certificate Authority](#).

cascading replication

In a cascading replication scenario, one server, often called the hub supplier, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. It receives updates from the supplier server that holds the master copy of the data and in turn supplies those updates to the consumer.

certificate

A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in the directory as user object attributes.

Certificate Authority

Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a [CA](#).

CGI

Common Gateway Interface. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts and can be written

in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.

chaining

A method for relaying requests to another server. Results for the request are collected, compiled, and then returned to the client.

changelog

A changelog is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on replica servers or on other masters, in the case of multi-master replication.

character type

Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext

Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

class definition

Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.

class of service

See [CoS](#).

classic CoS

A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.

client

See [LDAP client](#).

code page

An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.

collation order

Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

consumer

Server containing replicated directory trees or subtrees from a supplier server.

consumer server

In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.

CoS

A method for sharing attributes between entries in a way that is invisible to applications.

CoS definition entry

Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry

Contains a list of the shared attribute values.

See Also [template entry](#).

D**daemon**

A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.

DAP

Directory Access Protocol. The ISO X.500 standard protocol that provides client access to the directory.

data master

The server that is the master source of a particular piece of data.

database link

An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.

default index

One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.

definition entry

See [CoS definition entry](#).

Directory Access Protocol

See [DAP](#).

Directory Manager

The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the Directory Manager.

directory service

A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

directory tree

The logical representation of the information stored in the directory. It mirrors the tree model used by most filesystems, with the tree's root point appearing at the top of the hierarchy. Also known as [DIT](#).

distinguished name

String representation of an entry's name and location in an LDAP directory.

DIT

See [directory tree](#).

DM

See [Directory Manager](#).

DN

See [distinguished name](#).

DNS

Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as **www.example.com**). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.

DNS alias

A DNS alias is a hostname that the DNS server knows points to a different host specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as **www.yourdomain.domain** might point to a real machine

called **realthing.yourdomain.domain** where the server currently exists.

E

entry

A group of lines in the LDIF file that contains information about an object.

entry distribution

Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list

Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index

Allows you to search efficiently for entries containing a specific attribute value.

F

file extension

The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename **index.html** the file extension is **html**.

file type

The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter

A constraint applied to a directory query that restricts the information returned.

filtered role

Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

G

general access

When granted, indicates that all authenticated users can access directory information.

GSS-API

Generic Security Services. The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

H**hostname**

A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, **www.example.com** is the machine **www** in the subdomain **example** and **com** domain.

HTML

Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Mozilla Firefox how to display text, position graphics, and form items and to display links to other pages.

HTTP

Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.

HTTPD

An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.

HTTPS

A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.

hub

In the context of replication, a server that holds a replica that is copied from a different server, and, in turn, replicates it to a third server.

See Also [cascading replication](#).

I**ID list scan limit**

A size limit which is globally applied to any indexed search operation. When the size of an individual ID list reaches this limit, the server replaces that ID list with an all IDs token.

index key

Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS

An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index

Speeds up searches for information in international directories.

International Standards Organization

See [ISO](#).

IP address

Also Internet Protocol address. A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10). Directory Server supports both IPv4 and IPv6 IP addresses.

ISO

International Standards Organization.

K**knowledge reference**

Pointers to directory information stored in different databases.

L**LDAP**

Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAP client

Software used to request and view LDAP entries from an LDAP Directory Server.

See Also [browser](#).

LDAP Data Interchange Format

See [LDAP Data Interchange Format](#).

LDAP URL

Provides the means of locating Directory Servers using DNS and then completing the query via LDAP. A sample LDAP URL is **ldap://ldap.example.com**.

LDAPv3

Version 3 of the LDAP protocol, upon which Directory Server bases its schema format.

LDBM database

A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.

LDIF

LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.

leaf entry

An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol

See [LDAP](#).

locale

Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

M

managed object

A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role

Allows creation of an explicit enumerated list of members.

management information base

See [MIB](#).

mapping tree

A data structure that associates the names of suffixes (subtrees) with databases.

master

See [supplier](#).

master agent

See [SNMP master agent](#).

matching rule

Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5

A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data that is unique with high probability and is mathematically extremely hard to produce; a piece of data that will produce the same message digest.

MD5 signature

A message digest produced by the MD5 algorithm.

MIB

Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree-like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace

Management Information Base namespace. The means for directory data to be named and referenced. Also called the [directory tree](#).

monetary format

Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.

multi-master replication

An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a changelog for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor

The server containing the database link that communicates with the remote server.

N

n + 1 directory problem

The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions

Multiple entries with the same distinguished name.

nested role

Allows the creation of roles that contain other roles.

network management application

Network Management Station component that graphically displays information about SNMP managed devices, such as which device is up or down and which and how many error messages were received.

network management station

See [NMS](#).

NIS

Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, filesystems, and network parameters throughout a network of computers.

NMS

Powerful workstation with one or more network management applications installed. Also [network management station](#).

ns-slapd

Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server.

See Also [slapd](#).

O

object class

Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier

A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

See Also [OID](#).

OID

See [object identifier](#).

operational attribute

Contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

P**parent access**

When granted, indicates that users have access to entries below their own in the directory tree if the bind DN is the parent of the targeted entry.

pass-through authentication

See [PTA](#).

pass-through subtree

In pass-through authentication, the [PTA directory server](#) will pass through bind requests to the [authenticating directory server](#) from all clients whose DN is contained in this subtree.

password file

A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as `/etc/passwd` because of where it is kept.

password policy

A set of rules that governs how passwords are used in a given directory.

PDU

Encoded messages which form the basis of data exchanges between SNMP devices. Also [protocol data unit](#).

permission

In the context of access control, permission states whether access to the directory information is granted or denied and the level of access that is granted or denied.

See Also [access rights](#).

pointer CoS

A pointer CoS identifies the template entry using the template DN only.

presence index

Allows searches for entries that contain a specific indexed attribute.

protocol

A set of rules that describes how devices on a network exchange information.

protocol data unit

See [PDU](#).

proxy authentication

A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.

proxy DN

Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.

PTA

Mechanism by which one Directory Server consults another to check bind credentials. Also [pass-through authentication](#).

PTA directory server

In pass-through authentication ([PTA](#)), the PTA Directory Server is the server that sends (passes through) bind requests it receives to the [authenticating directory server](#).

PTA LDAP URL

In pass-through authentication, the URL that defines the [authenticating directory server](#), pass-through subtree(s), and optional parameters.

R

RAM

Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.

rc.local

A file on Unix machines that describes programs that are run when the machine starts. It is also called `/etc/rc.local` because of its location.

RDN

The name of the actual entry itself, before the entry's ancestors have been appended to the

string to form the full distinguished name. Also [relative distinguished name](#).

read-only replica

A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.

read-write replica

A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.

referential integrity

Mechanism that ensures that relationships between related entries are maintained within the directory.

referral

(1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP server that can process the request.

(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.

relative distinguished name

See [RDN](#).

replica

A database that participates in replication.

replica-initiated replication

Replication configuration where replica servers, either hub or consumer servers, pull directory data from supplier servers. This method is available only for legacy replication.

replication

Act of copying directory trees or subtrees from supplier servers to replica servers.

replication agreement

Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the replica servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

RFC

Request for Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

role

An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes

Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root

The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

root suffix

The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

S**SASL**

An authentication framework for clients as they attempt to bind to a directory. Also [Simple Authentication and Security Layer](#).

schema

Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

schema checking

Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default, and users will receive an error if they try to save an entry that does not conform to the schema.

Secure Sockets Layer

See [SSL](#).

self access

When granted, indicates that users have access to their own entries if the bind DN matches the targeted entry.

Server Console

Java-based application that allows you to perform administrative management of your Directory Server from a GUI.

server daemon

The server daemon is a process that, once running, listens for and accepts requests from clients.

Server Selector

Interface that allows you select and configure servers using a browser.

server service

A process on Windows that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.

service

A background process on a Windows machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE

Server Instance Entry. The ID assigned to an instance of Directory Server during installation.

Simple Authentication and Security Layer

See [SASL](#).

Simple Network Management Protocol

See [SNMP](#).

single-master replication

The most basic replication scenario in which multiple servers, up to four, each hold a copy of the same read-write replicas to replica servers. In a single-master replication scenario, the supplier server maintains a changelog.

SIR

See [supplier-initiated replication](#).

slapd

LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication.

See Also [ns-slapd](#).

SNMP

Used to monitor and manage application processes running on the servers by exchanging data about network activity. Also [Simple Network Management Protocol](#).

SNMP master agent

Software that exchanges information between the various subagents and the NMS.

SNMP subagent

Software that gathers information about the managed device and passes the information to the master agent. Also called a [subagent](#).

SOA

See [start of authority \(SOA\)](#).

SSL

A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP. Also called [Secure Sockets Layer](#).

standard index

An index maintained by default.

start of authority (SOA)

A record which contains the core information about a DNS zone.

sub suffix

A branch underneath a root suffix.

subagent

See [SNMP subagent](#).

substring index

Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.

suffix

The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

superuser

The most privileged user available on Unix machines. The superuser has complete access privileges to all files on the machine. Also called [root](#).

supplier

Server containing the master copy of directory trees or subtrees that are replicated to replica servers.

supplier server

In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication

Replication configuration where [supplier](#) servers replicate directory data to any replica servers.

symmetric encryption

Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index

Cannot be deleted or modified as it is essential to Directory Server operations.

T

target

In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry

The entries within the scope of a CoS.

TCP/IP

Transmission Control Protocol/Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.

template entry

See [CoS template entry](#).

time/date format

Indicates the customary formatting for times and dates in a specific region.

TLS

The new standard for secure socket layers; a public key based protocol. Also [Transport Layer Security](#).

topology

The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security

See [TLS](#).

U**uid**

A unique number associated with each user on a Unix system.

URL

Uniform Resource Locator. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is *protocol://machine:port/document*. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

V**virtual list view index**

Speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branch point in the directory tree to improve display performance.

See Also [browsing index](#).

X**X.500 standard**

The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation.

Index**A****access control**

- read permission, [A Brief Look at Access Control Concepts](#), [Defining Role-Based Access Controls](#), [Creating New Permissions](#)

Active Directory

- schema differences between Identity Management, [User Schema Differences between Identity Management and Active Directory](#)

attributes

- setting multi-valued, [From the Command Line](#)

B**bind**

- DNS and LDAP, [About DNS in IdM](#)

C**chkconfig, [Starting and Stopping the IdM Domain](#)****client**

- troubleshooting
 - installation, [Client Installations](#)
- uninstalling, [Uninstalling an IdM Client](#)

D**DHCP, [Adding Host Entries from the Command Line](#)****DNS**

- adding zone records, [Adding Records to DNS Zones](#)
- adding zones, [Adding Forward DNS Zones](#)
- bind-dyndb-ldap and Directory Server, [About DNS in IdM](#)
- disabling zones, [Enabling and Disabling Zones](#)
- dynamic updates, [Enabling Dynamic DNS Updates](#)
- hosts with DHCP, [Adding Host Entries from the Command Line](#)
- PTR synchronization
 - requirements, [Synchronizing Forward and Reverse Zone Entries](#)

DNS zone records, [Adding Records to DNS Zones](#)

- deleting, [Deleting Records from DNS Zones](#)
- format for adding, [About the Commands to Add DNS Records](#)
- IPv4 example, [Examples of Adding DNS Resource Records](#)
- IPv6 example, [Examples of Adding DNS Resource Records](#)
- PTR example, [Examples of Adding DNS Resource Records](#)
- SRV example, [Examples of Adding DNS Resource Records](#)
- types of records, [Adding Records to DNS Zones](#)

glue entries, [Solving Orphan Entry Conflicts](#)

H

hosts

- creating
 - with DHCP, [Adding Host Entries from the Command Line](#)
- disabling, [Disabling and Re-enabling Host Entries](#)

I

installing clients

- disabling OpenSSH, [About ipa-client-install and OpenSSH](#)

K

Kerberos, [About Kerberos](#)

- separate credentials cache, [Caching User Kerberos Tickets](#)
- SSSD password cache, [Caching Kerberos Passwords](#)
- ticket policies, [Setting Kerberos Ticket Policies](#)
 - global, [Setting Global Ticket Policies](#)
 - user-level, [Setting User-Level Ticket Policies](#)

L

logging in

- SELinux problems, [SELinux Login Problems](#)
- separate credentials cache, [Caching User Kerberos Tickets](#)

N

naming conflicts

- in replication, [Solving Naming Conflicts](#)

P

password expiration, [Managing Password Expiration Limits](#)

password policies

- expiration, [Managing Password Expiration Limits](#)

port forwarding

- for the UI, [Using the UI with Proxy Servers](#)

proxy servers

- for the UI, [Using the UI with Proxy Servers](#)

PTR synchronization

- requirements, [Synchronizing Forward and Reverse Zone Entries](#)

R**reboot, [Starting and Stopping the IdM Domain](#)****replicas**

- number in replication, [About IdM Servers and Replicas](#)

replication

- size limits, [About IdM Servers and Replicas](#)

S**schema**

- differences between Identity Management and Active Directory, [User Schema Differences between Identity Management and Active Directory](#)
 - cn, [Values for cn Attributes](#)
 - initials, [Constraints on the initials Attribute](#)
 - sn, [Requiring the surname \(sn\) Attribute](#)
 - street and streetAddress, [Values for street and streetAddress](#)

SELinux

- login problems, [SELinux Login Problems](#)

servers

- number in replication, [About IdM Servers and Replicas](#)

services

- disabling, [Disabling and Re-enabling Service Entries](#)

SSH

- disabling at client install, [About ipa-client-install and OpenSSH](#)

SSSD

- and Kerberos passwords, [Caching Kerberos Passwords](#)
 - disabling cache, [Caching Kerberos Passwords](#)

starting with chkconfig, [Starting and Stopping the IdM Domain](#)

T

ticket policies, [Setting Kerberos Ticket Policies](#)

troubleshooting

- client installation, [Client Installations](#)
- Kerberos, unknown server error, [The client can't resolve reverse hostnames when using an external DNS.](#)
- resolving hostnames on client, [The client can't resolve reverse hostnames when using an external DNS.](#)
- SELinux, [SELinux Login Problems](#)

U

uninstalling

- clients, [Uninstalling an IdM Client](#)

users

- multi-valued attributes, [From the Command Line](#)
- password expiration, [Managing Password Expiration Limits](#)
- separate credentials cache, [Caching User Kerberos Tickets](#)

W

web UI

- port forwarding, [Using the UI with Proxy Servers](#)
- proxy servers, [Using the UI with Proxy Servers](#)

Z

zone records, [Adding Records to DNS Zones](#)

- deleting, [Deleting Records from DNS Zones](#)
- format for adding, [About the Commands to Add DNS Records](#)
- IPv4 example, [Examples of Adding DNS Resource Records](#)
- IPv6 example, [Examples of Adding DNS Resource Records](#)
- PTR example, [Examples of Adding DNS Resource Records](#)
- SRV example, [Examples of Adding DNS Resource Records](#)
- types, [Adding Records to DNS Zones](#)