



# **Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide**

---

Managing Identity and Authorization Policies for Linux-Based Infrastructures

Tomáš Čapek

Aneta Petrová

Ella Deon Ballard



# Red Hat Enterprise Linux 7 Linux Domain Identity, Authentication, and Policy Guide

---

## Managing Identity and Authorization Policies for Linux-Based Infrastructures

Tomáš Čapek  
Red Hat Customer Content Services  
tcapek@redhat.com

Aneta Petrová  
Red Hat Customer Content Services  
apetrova@redhat.com

Ella Deon Ballard  
Red Hat Customer Content Services

## Legal Notice

Copyright © 2015 Red Hat.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-sa/3.0/). 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.

## Keywords

1. FreeIPA. 2. Identity Management. 3. IdM. 4. IPA.

## Abstract

Identity and policy management, for both users and machines, is a core function for most enterprise environments. Identity Management 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.



# Table of Contents

<b>Chapter 1. Introduction to Identity Management</b>	<b>5</b>
1.1. IdM v. LDAP: A More Focused Type of Service	5
1.2. Bringing Linux Services Together	8
1.3. Relationships Between Servers and Clients	11
1.4. Additional Resources	15
<b>Part I. Installing Identity Management Servers and Services</b>	<b>17</b>
<b>Chapter 2. Prerequisites for Installation</b>	<b>18</b>
2.1. Supported Server Platforms	18
2.2. Hardware Recommendations	18
2.3. Software Requirements	18
2.4. System Prerequisites	19
<b>Chapter 3. Installing an IdM Server</b>	<b>26</b>
3.1. The ipa-server-install utility	26
3.2. Installation Procedure Descriptions and Examples	29
<b>Chapter 4. Setting up IdM Replicas</b>	<b>43</b>
4.1. Planning the Server and Replica Topologies	43
4.2. Prerequisites for Installing a Replica Server	44
4.3. Creating the Replica	46
4.4. Adding Additional Replication Agreements	51
<b>Chapter 5. Setting up Systems as IdM Clients</b>	<b>52</b>
5.1. What Happens in Client Setup	52
5.2. Opening the IdM Required System Ports	53
5.3. Configuring a Linux System as an IdM Client	54
5.4. Manually Configuring a Linux Client	58
5.5. Setting up a Linux Client Through Kickstart	65
5.6. Re-enrolling a Host	66
5.7. Renaming Machines and Reconfiguring IdM Client Configuration	67
5.8. Performing a Two-Administrator Enrollment	68
5.9. Removing Clients from the Domain	69
5.10. Manually Unconfiguring Client Machines	69
<b>Chapter 6. Upgrading Identity Management</b>	<b>71</b>
6.1. Migrating the IdM Server to Red Hat Enterprise Linux 7	72
<b>Chapter 7. Uninstalling IdM Servers and Replicas</b>	<b>80</b>
<b>Chapter 8. The Basics of Managing the IdM Server and Services</b>	<b>81</b>
8.1. Starting and Stopping the IdM Domain	81
8.2. About the IdM Client Tools	81
8.3. Logging into IdM	86
8.4. Using the IdM Web UI	88
<b>Chapter 9. Backing Up and Restoring Identity Management</b>	<b>99</b>
9.1. Full-Server Backup and Data-Only Backup	100
9.2. Restoring a Backup	104
<b>Part II. Managing User Identities in a Linux Domain</b>	<b>106</b>
<b>Chapter 10. Managing Users and User Groups</b>	<b>107</b>
10.1. Setting up User Home Directories	107

10.2. Managing User Entries	109
10.3. Managing Public SSH Keys for Users	116
10.4. Changing Passwords	122
10.5. Enabling and Disabling User Accounts	124
10.6. Unlocking User Accounts After Password Failures	126
10.7. Managing User Private Groups	127
10.8. Managing Unique UID and GID Number Assignments	128
10.9. Managing User and Group Schema	132
10.10. Managing User Groups	143
10.11. Issuing User Certificates with the IdM CA	163
10.12. Managing User Certificates	167
<b>Part III. Managing System Identities in a Linux Domain</b>	<b>170</b>
<b>Chapter 11. Managing Hosts</b>	<b>171</b>
11.1. About Hosts, Services, and Machine Identity and Authentication	171
11.2. About Host Entry Configuration Properties	172
11.3. Disabling and Re-enabling Host Entries	173
11.4. Creating Certificates for Hosts	174
11.5. Managing Public SSH Keys for Hosts	184
11.6. Setting Ethers Information for a Host	191
11.7. Managing Host Groups	192
<b>Chapter 12. Managing Services</b>	<b>197</b>
12.1. Adding and Editing Service Entries and Keytabs	197
12.2. Creating Certificates for Services	200
12.3. Storing Certificates in NSS Databases	211
12.4. Configuring Clustered Services	211
12.5. Using the Same Service Principal for Multiple Services	212
12.6. Disabling and Re-enabling Service Entries	212
<b>Chapter 13. Delegating User Access to Hosts and Services</b>	<b>214</b>
13.1. Delegating Service Management	214
13.2. Delegating Host Management	215
13.3. Delegating Host or Service Management in the Web UI	215
13.4. Accessing Delegated Services	217
<b>Chapter 14. Integrating with NIS Domains and Netgroups</b>	<b>218</b>
14.1. About NIS and Identity Management	218
14.2. Setting the NIS Port for Identity Management	219
14.3. Creating Netgroups	220
14.4. Exposing Automount Maps to NIS Clients	225
14.5. Migrating from NIS to IdM	226
<b>Chapter 15. Managing DNS</b>	<b>233</b>
15.1. Installing DNS Services Into an Existing Server	233
15.2. BIND in Identity Management	233
15.3. Supported DNS Zone Types	234
15.4. DNS Configuration Priorities	235
15.5. Managing Master DNS Zones	235
15.6. Managing Dynamic DNS Updates	249
15.7. Managing DNS Forwarding	256
15.8. Managing Reverse DNS Zones	263
15.9. Defining DNS Query Policy	265

<b>Part IV. Defining Domain-wide System Policies</b>	<b>267</b>
<b>Chapter 16. Using Automount</b>	<b>268</b>
16.1. About Automount and IdM	268
16.2. Configuring Automount	269
16.3. Setting up a Kerberized NFS Server	274
16.4. Configuring Locations	277
16.5. Configuring Maps	279
<b>Chapter 17. Defining Password Policies</b>	<b>286</b>
17.1. About Password Policies and Policy Attributes	286
17.2. Viewing Password Policies	288
17.3. Creating and Editing Password Policies	294
17.4. Managing Password Expiration Limits	297
17.5. Changing the Priority of Group Password Policies	298
17.6. Setting Account Lockout Policies	298
17.7. Enabling a Password Change Dialog	301
<b>Chapter 18. Managing the Kerberos Domain</b>	<b>302</b>
18.1. About Kerberos	302
18.2. Setting Kerberos Ticket Policies	303
18.3. Refreshing Kerberos Tickets	305
18.4. Kerberos Flags for Services and Hosts	307
18.5. Caching Kerberos Passwords	309
18.6. Removing Keytabs	310
<b>Chapter 19. Using sudo</b>	<b>311</b>
19.1. About sudo and IPA	311
19.2. Setting up sudo Commands and Command Groups	312
19.3. Defining sudo Rules	317
19.4. Configuring Hosts to Use IdM sudo Policies	328
<b>Chapter 20. Configuring Host-Based Access Control</b>	<b>331</b>
20.1. About Host-Based Access Control	331
20.2. Creating Host-Based Access Control Entries for Services and Service Groups	332
20.3. Defining Host-Based Access Control Rules	336
20.4. Testing Host-Based Access Control Rules	344
<b>Chapter 21. Defining SELinux User Maps</b>	<b>349</b>
21.1. About Identity Management, SELinux, and Mapping Users	349
21.2. Configuring SELinux User Map Order and Defaults	351
21.3. Mapping SELinux Users and IdM Users	354
<b>Chapter 22. Defining Automatic Group Membership for Users and Hosts</b>	<b>360</b>
22.1. About Automembership	360
22.2. Defining Automembership Rules (Basic Procedure)	361
22.3. Examples of Using Automember Groups	364
<b>Chapter 23. Restricting Domains for PAM services</b>	<b>367</b>
<b>Part V. Configuring the Identity Management Server</b>	<b>369</b>
<b>Chapter 24. Defining Access Control for IdM Users</b>	<b>370</b>
24.1. Access Controls for IdM Entries	370
24.2. Defining Self-Service Settings	371
24.3. Delegating Permissions over Users	375
24.4. Defining Role-Based Access Controls	377

24.4. Defining Role-Based Access Controls	377
<b>Chapter 25. Identity Management Files and Logs</b>	<b>394</b>
25.1. A Reference of IdM Server Configuration Files and Directories	394
25.2. IdM Domain Services and Log Rotation	396
25.3. About default.conf and Context Configuration Files	397
25.4. Checking IdM Server Logs	398
<b>Chapter 26. Managing Certificates and Certificate Authorities</b>	<b>405</b>
26.1. Renewal Messages	405
26.2. Automatic CA Certificate Renewal	405
26.3. Manual CA Certificate Renewal	405
26.4. Changing Certificate Chaining	406
26.5. Starting IdM with Expired Certificates	407
26.6. Configuring Alternate Certificate Authorities	408
26.7. Promoting a Replica to a Master CA Server	408
26.8. Configuring OCSP Responders	411
26.9. Certificate Profiles	413
26.10. Certificate Authority ACL Rules	418
<b>Chapter 27. Disabling Anonymous Binds</b>	<b>425</b>
<b>Chapter 28. Changing Domain DNS Configuration</b>	<b>426</b>
28.1. Setting DNS Entries for Multi-Homed Servers	426
28.2. Setting up Additional Name Servers	426
28.3. Changing Load Balancing for IdM Servers and Replicas	426
<b>Chapter 29. Managing the Server-Replica Relationships</b>	<b>428</b>
29.1. Managing Replication Agreements Between IdM Servers	428
29.2. Removing a Replica	436
29.3. Renaming a Server or Replica Host System	436
<b>Chapter 30. Migrating from an LDAP Directory to IdM</b>	<b>438</b>
30.1. An Overview of LDAP to IdM Migration	438
30.2. Examples for Using migrate-ds	446
30.3. Scenario 1: Using SSSD as Part of Migration	448
30.4. Scenario 2: Migrating an LDAP Server Directly to Identity Management	450
30.5. Scenario 3: Migrating over SSL	451
<b>Appendix A. Troubleshooting Identity Management</b>	<b>454</b>
A.1. Installation Issues	454
A.2. UI Connection Problems	458
A.3. IdM Server Problems	459
A.4. Host Problems	460
A.5. Kerberos Errors	461
A.6. SELinux Login Problems	462
<b>Index</b>	<b>462</b>
<b>Appendix B. Revision History</b>	<b>466</b>

# Chapter 1. Introduction to Identity Management

Red Hat Identity Management 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, and authorization software is hardly new, Identity Management is one of the only options that support Linux and 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 that share centrally-managed services, like Kerberos and DNS. This introductory chapter explains:

- » what Identity Management is
- » how all the centrally-managed services work together within the IdM domain
- » how servers and clients interact 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 was previously unavailable to Linux and Unix environments, and it does it using native Linux applications and protocols.

### 1.1.1. Defining a True Linux Domain

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. With IdM, **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. IdM servers and IdM clients are Linux or Unix machines, and Identity Management is a management tool for Linux domains. IdM can also synchronize data with an Active Directory domain to allow integration with Windows servers, but it does not support Windows clients.
- ✧ 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.

IdM serves as a bridge between Linux and the IdM world. IdM, when used in concert with [Cross-Realm Kerberos Authentication](#), makes it possible for both IdM and Linux to cooperate in terms of identity, authentication and authorization. IdM and Kerberos are each able to use their own native clients.

IdM provides a very simple solution to a very common and very specific problem: identity management. In a sense, Identity Management isn't making administrators do something new; it is helping them do it better. The following examples of how IdM can be used in various company environments illustrates some of the capabilities of Red Hat Identity Management.

### **IdM in a *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.

### **IdM in a *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.

### **IdM in an *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 and Unix servers, using their native tools and applications, which is 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.

### 1.1.2. Contrasting Identity Management with a Standard LDAP Directory

The closest relative to Identity Management is a standard LDAP directory like Red Hat Directory Server. However, they have a different purpose. 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 back end, and it is not a general policy server.

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 host names), a generic directory service can store and retrieve any kind of information. LDAP directories like Red Hat 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 Directory Server are frequently used as back ends that store data for other applications. 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.

**Identity Management focuses on identities** (user and machine) **and policies** that relate to those identities and their interactions. While it uses an LDAP back end 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 and 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 Red Hat Directory Server**

	Red Hat 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

	Red Hat Directory Server	Identity Management
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 Red Hat Directory Server have flexibility and adaptability which makes them a perfect back end to any number of applications. Its primary purpose is to store and retrieve data efficiently.

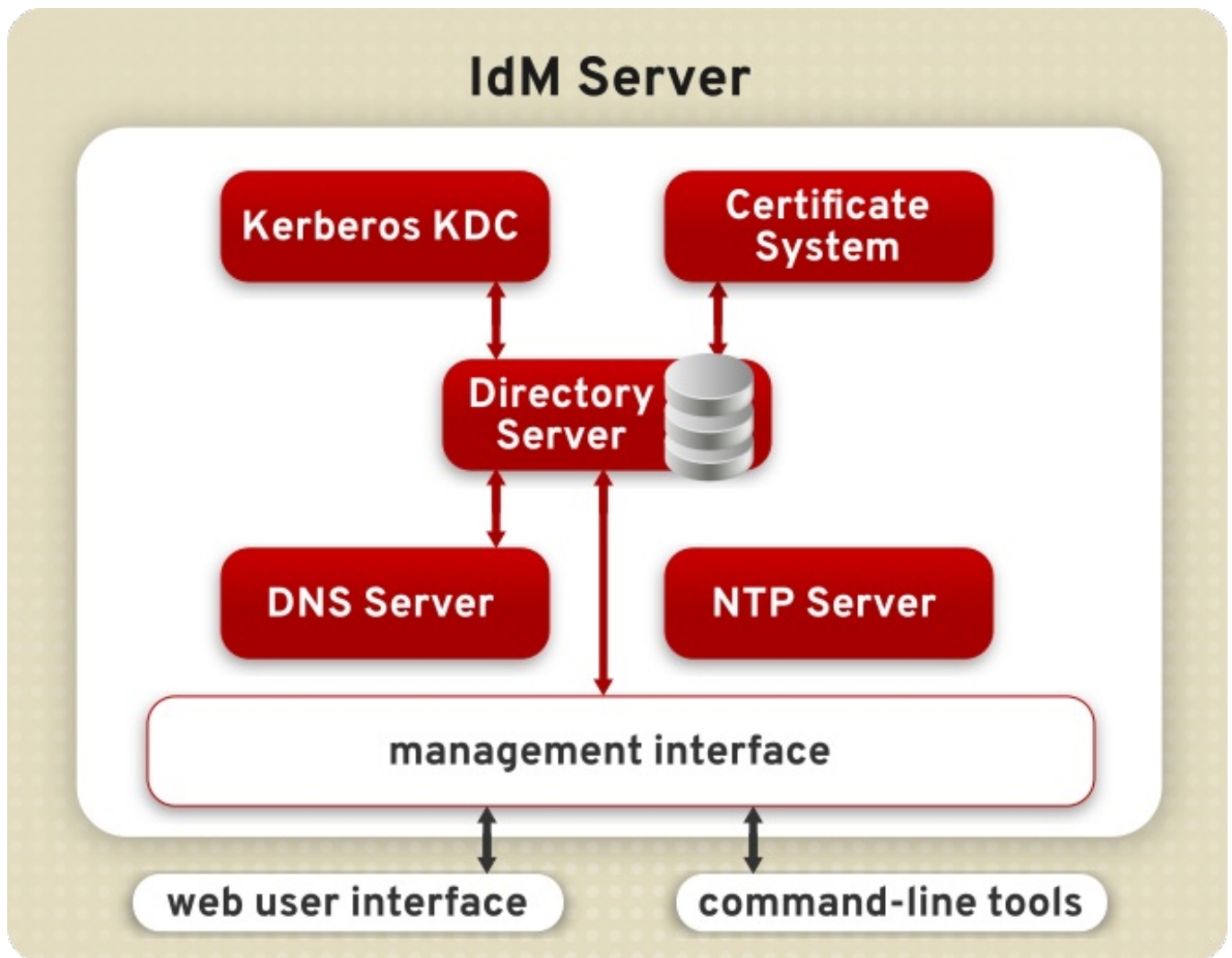
IdM fills a very different niche. It is optimized to perform a single task very effectively. It stores user information and authentication and authorization policies, as well as other information related to access, like host information. Its single purpose is to manage identities.

## 1.2. Bringing Linux Services Together

Identity Management unifies various related Linux services into a single management environment. It establishes a simple, easy way to bring host machines into the domain of those services.

At its core, an IdM server is an identity and authentication server. The primary IdM server is essentially a domain controller, and it uses a Kerberos server and KDC for authentication. An LDAP back end contains all 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 and 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 system 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 the Directory Server back end for IdM. 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: Red Hat Directory Server

Identity Management contains an internal Red Hat 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 Directory Server instance.

When multiple servers are configured, they can talk to each other because 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: Red Hat 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 Red Hat 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). In Red Hat Enterprise Linux 7, CA is optional. You can set up a CA-less IdM deployment by only providing the necessary signed certificates. For more information about the possible CA configurations, see [Section 3.2.3, “Installing with Different CA Configurations”](#).

## 1.2.4. Service Discovery: DNS

Identity Management uses the Domain Name System (DNS) for dynamic service discovery. The IdM client installation utility can use information from DNS to automatically configure the client machine. After the client is enrolled in the IdM domain, it uses DNS to locate IdM servers and services within the domain. For more information about service discovery, see the [System-Level Authentication Guide](#).

Having the IdM server also be a DNS server is optional but strongly recommended. When the IdM server also manages DNS, there is tight integration between the DNS and native IdM tools which automates some of DNS record management. Even if an IdM server is used as a master DNS server, other external DNS servers can still be used as slave servers.

### 1.2.5. Management: SSSD

The System Security Services Daemon (SSSD) is a platform application that caches credentials. Most system authentication is configured locally, which means that services must check with a local user store to determine users and credentials. SSSD allows a local service to check with a local cache in SSSD. The cache may be taken from any variety of remote identity providers, including Identity Management.

SSSD can cache user names and passwords, Kerberos principals and keytabs, automount maps, sudo rules that are defined on IPA servers, and SSH keys that are used by Identity Management domain users and systems. This allows two significant benefits to administrators: first, all identity configuration can be centralized in a single application (the IdM server); and second, external information can be cached on a local system to continue normal authentication operations in case the system or the IdM server becomes unavailable.

SSSD is automatically configured by IdM client installation and management scripts, so the system configuration never needs to be manually updated, even as domain configuration changes.

Consistently with Windows Active Directory, SSSD allows the user to log in with either the user name attribute or the User Principal Name (UPN) attribute.

SSSD supports the **true**, **false**, and **preserve** values for the **case\_sensitive** option. When the **preserve** value is enabled, the input matches regardless of the case, but the output is always the same case as on the server; SSSD preserves the case for the UID field as it is configured.

SSSD allows certain cached entries to be refreshed in the background, so the entries are returned instantly because the back end keeps them updated at all times. Currently, entries for users, groups, and netgroups are supported.

### 1.2.6. 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. The 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. 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, and they also act as central repositories of identity and policy information.



#### Note

Most of the supported services, for which an IdM server serves as a controller, are not required. For example, a server may have a CA, a DNS server, an NTP server, or it can be installed without those services.

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).

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*. There are some differences between IdM servers and IdM replicas:

- ✧ While a server is a new installation, which means that it defines the domain configuration, a replica is based on an existing server and an existing domain configuration. Once an instance is configured, servers and replicas are basically identical in functionality and behavior within the IdM domain.
- ✧ In versions of Red Hat Enterprise Linux prior to 7.1, only one server in the IdM domain generates the CRL and renews the PKI subsystem certificates.
- ✧ Starting with Red Hat Enterprise Linux 7.1, only one server in the IdM domain can renew DNSSEC keys.

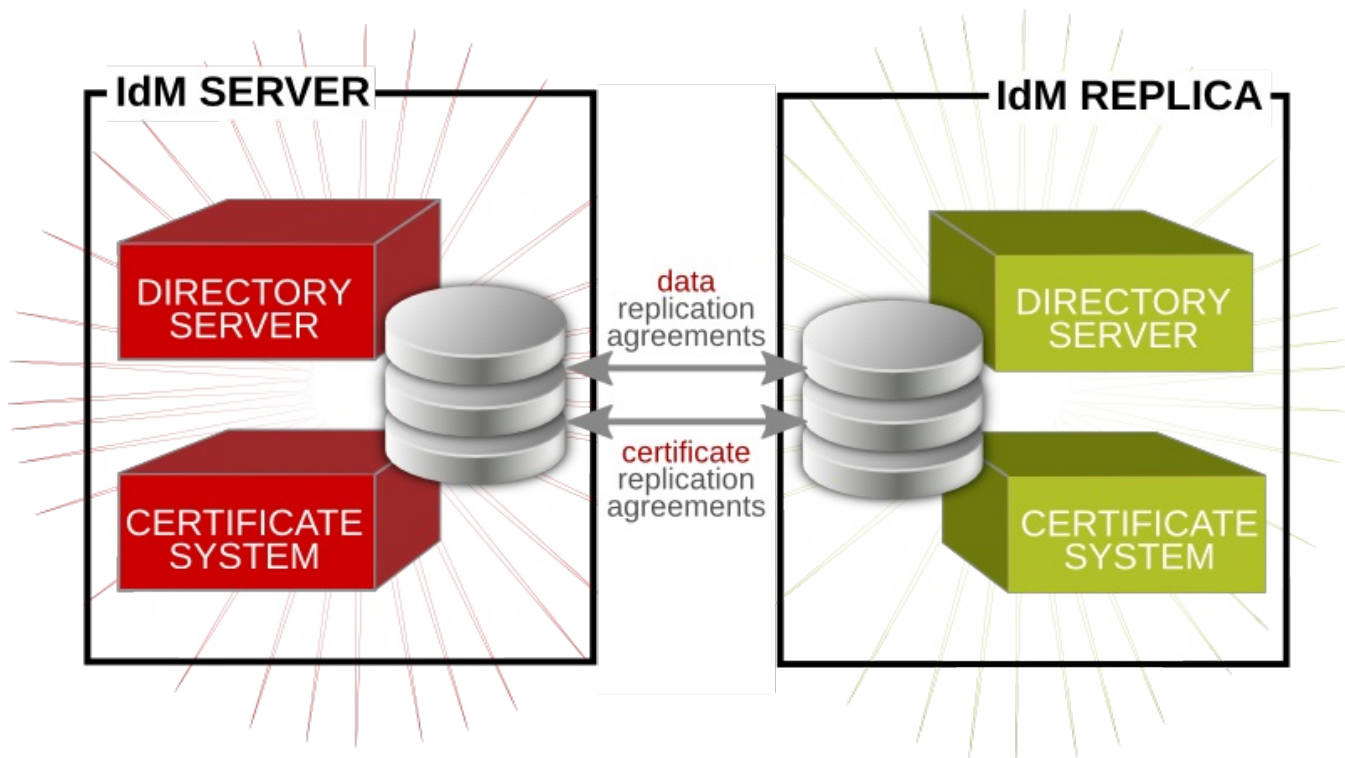


#### Note

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 the configuration of Server A 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*. Replication agreements are configured for all LDAP back ends as well as the LDAP subtrees used by Red Hat Certificate System. 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 Red Hat Directory Server domain. Planning the server and replica topology is described more in [Section 4.1, “Planning the Server and Replica Topologies”](#).



**Figure 1.2. Server and Replica Interactions**

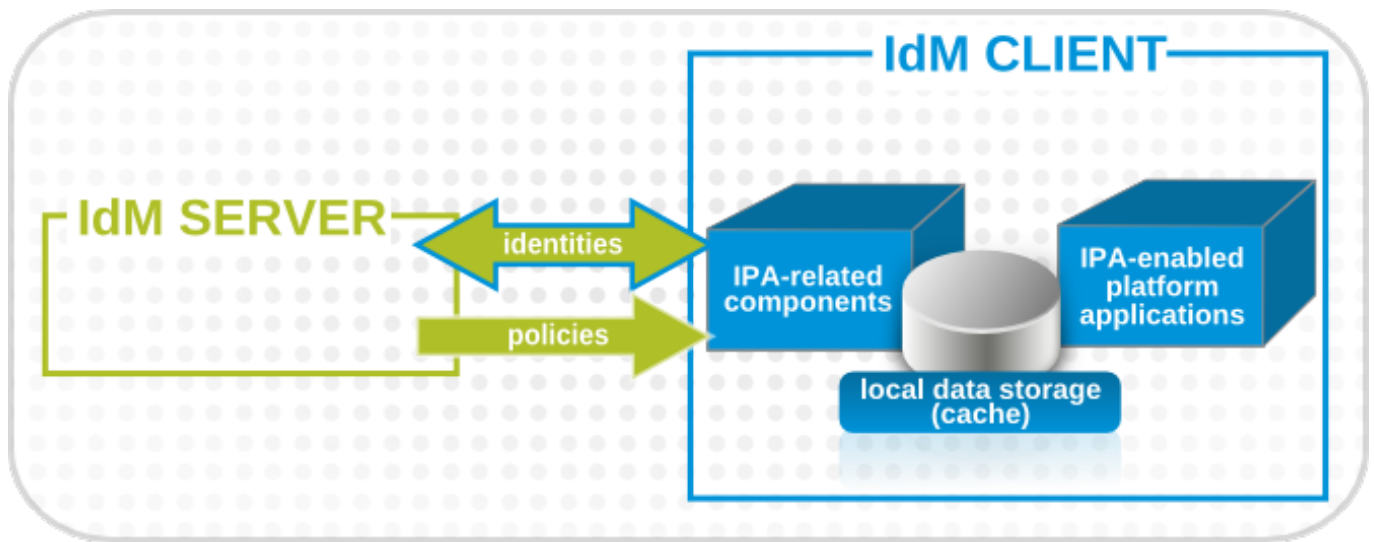
### Note

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 15.5, “Adding an SRV Record”](#) and [Section 28.3, “Changing Load Balancing for IdM Servers and Replicas”](#).

## 1.3.2. 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 is 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.

IdM clients use a number of IdM-enabled platform applications, as well as tools provided by IdM itself. For Red Hat Enterprise Linux systems, the platform tools available for IdM to use include for example the System Security Services Daemon (SSSD). IdM itself provides other tools, such as certain PAM and NSS modules and IdM command-line utilities. These are IdM components, rather than platform components used by IdM.



**Figure 1.3. Server and Client Interactions**

IdM uses the local storage (cache) on a client to improve performance by:

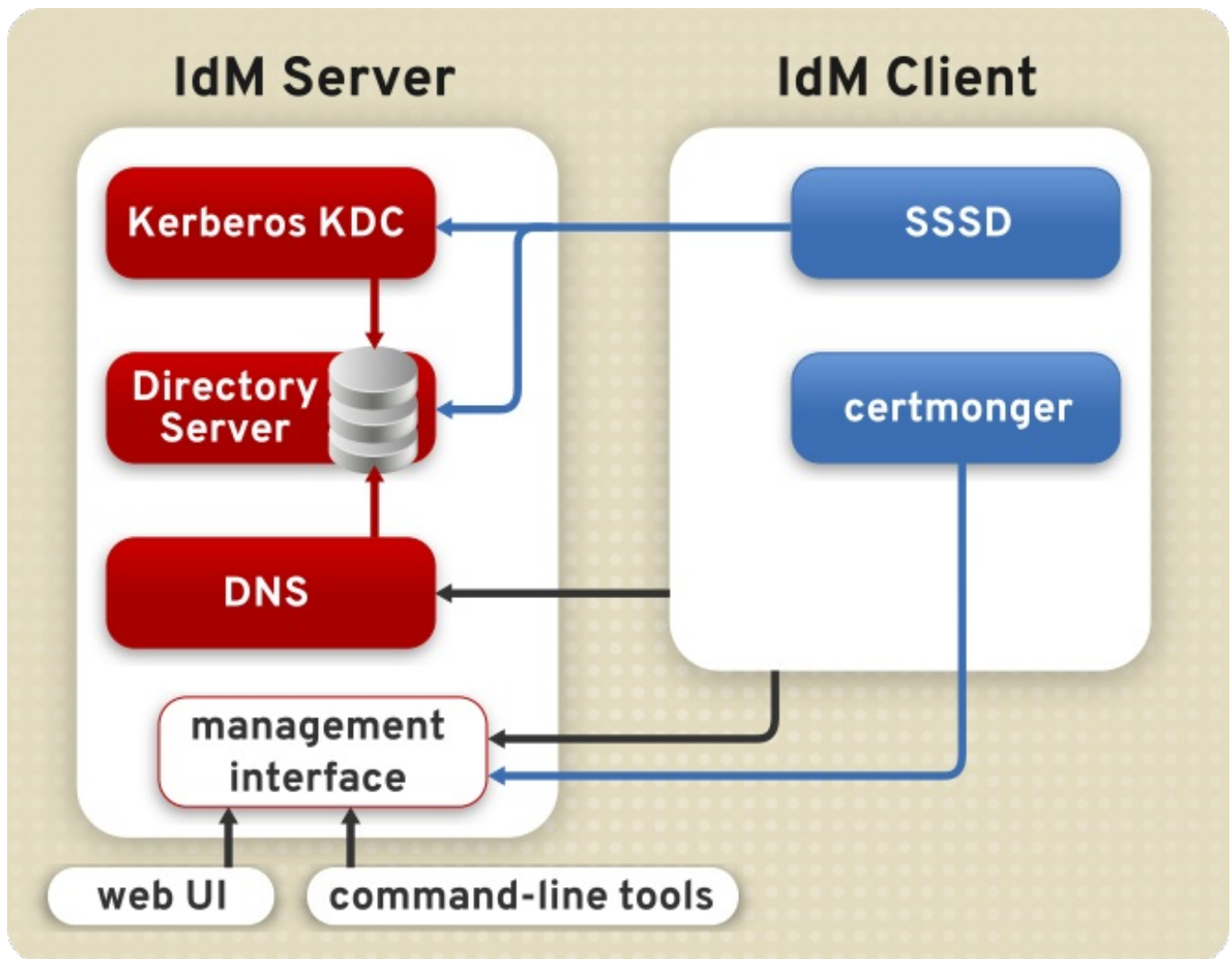
- » storing IdM information when the machine is offline
- » keeping information active beyond its normal timeout period if the client cannot access the central server; the cache is persistent even after rebooting the machine
- » reducing the round-trip time of requests by checking information locally before looking at the server

Information is stored either in an LDB database (similar to LDAP) or the local file system (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 Red Hat 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.
- The **certmonger** service 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, but it may be used by other services, such as a web server.

## 1.4. Additional Resources

In addition to this guide, you can find documentation on other features and services related to Red Hat Enterprise Linux Identity Management in the following guides:

### [System-Level Authentication Guide](#)

The *System-Level Authentication Guide* documents different applications and services available to configure authentication on local systems, including the

**authconfig** utility and the one-time password (OTP) authentication, the SSSD service, the Pluggable Authentication Module (PAM) framework, Kerberos, the **certmonger** utility, and single-sign on (SSO) for applications.

### [Windows Integration Guide](#)

The *Windows Integration Guide* documents how to integrate Linux domains with Microsoft Windows Active Directory (AD) using Identity Management. Among other topics, the guide covers various aspects of direct and indirect AD integration, the ID Views feature, using SSSD to access a Common Internet File System (CIFS), and the **realmd** system.



## **Part I. Installing Identity Management Servers and Services**

## Chapter 2. Prerequisites for Installation

The Identity Management installation and configuration process requires the environment to be suitably configured. You are also required to provide certain information during the installation and configuration procedures, such as realm names and certain user names and passwords. The following section describes these requirements.

### 2.1. Supported Server Platforms

IdM 4.1 is supported on the Red Hat Enterprise Linux 7 x86\_64 platform.

### 2.2. Hardware Recommendations

A basic user entry is approximately 1 KB in size. A simple host entry with a certificate is also approximately 1 KB in size.

RAM is the most important hardware feature to size properly. While all deployments are different, depending on the number of users and groups and the type of data stored, you can use the following recommendations as guidelines for determining how much RAM your IdM deployment might require:

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



#### Note

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 chapter about optimizing system performace in the Directory Server documentation](#).

### 2.3. Software Requirements

Most of the packages that an IdM server depends on are installed automatically as dependencies when the *ipa-server* package is installed. The dependencies installed together with *ipa-server* include packages such as *389-ds-base* for the LDAP service or *krb5-server* for the Kerberos service, as well as various IdM tools.

If you want to have the IdM server set up as a DNS server, which is strongly recommended, install the *ipa-server-dns* package before installing the IdM server.

For more information on DNS and why it is recommended to run a DNS server on the IdM server, see [Section 1.2.4, “Service Discovery: DNS”](#).



## Important

Due to [CVE-2014-3566](#), the Secure Socket Layer version 3 (SSLv3) protocol needs to be disabled in the `mod_nss` module. You can ensure that by following these steps:

1. Edit the `/etc/httpd/conf.d/nss.conf` file and set the `NSSProtocol` parameter to `TLSv1.0` (for backward compatibility) and `TLSv1.1`.

```
NSSProtocol TLSv1.0,TLSv1.1
```

2. Restart the `httpd` service.

```
# systemctl restart httpd.service
```

Note that Identity Management in Red Hat Enterprise Linux 7 automatically performs the above steps when the `yum update ipa-*` command is launched to upgrade the main packages.

## 2.4. System Prerequisites

The IdM server is set up using a configuration script that makes certain assumptions about the host system. If the host system does not meet these prerequisites, server configuration can fail.

### 2.4.1. System Files

The system, on which IdM is installed, is recommended to be clean. No custom configuration for services like DNS or Kerberos should be present on the system before installing and configuring the IdM server.

The IdM server script overwrites system files to set up the IdM domain. System files are backed up to `/var/lib/ipa/sysrestore/` during the installation of servers and replicas.

### 2.4.2. Host Name and DNS Configuration

Proper DNS configuration and host name settings are required for IdM servers and replicas of these servers to function correctly. The server host must have DNS properly configured regardless of whether the DNS server is integrated within IdM or hosted externally.

**Identity Management requires one separate DNS domain to be used for service records. To avoid conflicts on DNS level, the *primary DNS domain* used for IdM cannot be shared with any other system.** Follow recommended DNS naming practices, as described in the [Red Hat Enterprise Linux Security Guide](#).

Note that host names of IdM clients are not required to be part of the primary DNS domain.



## Warning

The primary DNS domain and the Kerberos realm cannot be changed after the initial installation. Red Hat strongly recommends that the Kerberos realm name is the same as the primary DNS domain name, with all letters uppercase. For example, if primary DNS domain is **ipa.example.com**, the **IPA.EXAMPLE.COM** Kerberos realm name is recommended. Different naming practices will prevent you from using Active Directory trusts and can have other negative consequences.

IdM can be configured to use a separate domain hosted on a standard, non-integrated DNS server. In such cases, the new domain must be manually created on the DNS server and manually filled with records from the zone file that will be generated by the IdM installer. Also, these records must be manually updated after installing or removing a replica, as well as after any changes in the service configuration, such as after an Active Directory trust is configured.

To reduce the maintenance burden, Red Hat recommends to install IdM with an integrated DNS server, as described in [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#). This automates most of the DNS record maintenance. If an IdM-integrated DNS server is used for DNS domain, set up correct delegation from the parent domain to the IdM servers. For example, if the IdM domain name is **ipa.example.com**, it must be properly delegated from the **example.com** domain.



## Note

You can verify the delegation using the **dig @IP address +norecurse +short ipa.example.com. NS** command, where *IP address* is the IP address of the server that manages the **example.com** DNS domain. If the delegation is correct, the command lists the IdM servers that have a DNS server installed.

## Verifying the Server Host Name

Use the **hostname** utility to display the host name.

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

The host name must be a fully-qualified domain name, such as **server.example.com** in the above example.



## Important

The fully-qualified domain name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the host name cause DNS failures. Additionally, the host name must be all lower-case; no capital letters are allowed. For other recommended naming practices, see the [Red Hat Enterprise Linux Security Guide](#).

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** utility

cannot be **localhost** or **localhost6**.

## Verifying the Forward and Reverse DNS Configuration

1. Obtain the IP address of the server. The **ip addr show** command displays both the IPv4 and IPv6 addresses:

- » The IPv4 address is displayed on the line starting with **inet**. In the following example, the configured IPv4 address is **192.0.2.1**.
- » The IPv6 address is displayed on the line starting with **inet6**. Only IPv6 addresses with **scope global** are relevant for this procedure. In the following example, the returned IPv6 address is **2001:DB8::1111**.

```
[root@server ~]# ip addr show
...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP group default qlen 1000
    link/ether 00:1a:4a:10:4e:33 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global dynamic eth0
        valid_lft 106694sec preferred_lft 106694sec
    inet6 2001:DB8::1111/32 scope global dynamic
        valid_lft 2591521sec preferred_lft 604321sec
    inet6 fe80::56ee:75ff:fe2b:def6/64 scope link
        valid_lft forever preferred_lft forever
```

2. Verify the forward DNS configuration by using the **dig** utility and adding the host name.
  - a. Run the **dig +short server.example.com A** command. The returned IPv4 address must match the IP address returned by **ip addr show**:

```
[root@server ~]# dig +short server.example.com A
192.0.2.1
```

- b. Run the **dig +short server.example.com AAAA** command. If the command returns an address, it must match the IPv6 address returned by **ip addr show**:

```
[root@server ~]# dig +short server.example.com AAAA
2001:DB8::1111
```



### Note

If no output is returned for the AAAA record, it does not indicate incorrect configuration; no output only means that no IPv6 address is configured in DNS for the server machine. If you do not intend to use the IPv6 protocol in your network, you can proceed with the installation in this situation.

3. Verify the reverse DNS configuration (PTR records) by using the **dig** utility and adding the IP address.

- a. Run the **dig +short -x IPv4 address** command. The server host name must be displayed in the command output. For example:

```
[root@server ~]# dig +short -x 192.0.2.1
server.example.com
```

- b. Use **dig** to query the IPv6 address as well if the **dig +short -x server.example.com AAAA** command in the previous step returned an IPv6 address. Again, the server host name must be displayed in the command output. For example:

```
[root@server ~]# dig +short -x 2001:DB8::1111
server.example.com
```



### Note

If **dig +short server.example.com AAAA** in the previous step did not display any IPv6 address, querying the AAAA record does not output anything. In this case, this is normal behavior and does not indicate incorrect configuration.

If a different host name or no host name is displayed, even though **dig +short server.example.com** in the previous step returned an IP address, it indicates that the reverse DNS configuration is incorrect.

## Verifying the Standards-compliance of DNS Forwarders

When configuring IdM with integrated DNS, verify that all DNS forwarders you want to use with the IdM DNS server comply with the [Extension Mechanisms for DNS](#) (EDNS0) and [DNS Security Extensions](#) (DNSSEC) standards. To do this, inspect the output of the following command for each forwarder separately:

```
$ dig +dnssec @IP_address_of_the_DNS_forwarder . SOA
```

The expected output displayed by the command contains the following information:

- ✧ status: **NOERROR**
- ✧ flags: **ra**
- ✧ EDNS flags: **do**
- ✧ The **RRSIG** record must be present in the **ANSWER** section

If any of these items is missing from the output, inspect the documentation of your DNS forwarder and verify that EDNS0 and DNSSEC are supported and enabled. In latest versions of the BIND server, the **dnssec-enabled yes;** option must be set in the **/etc/named.conf** file.

For example, the expected output can look like this:

```
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 48655
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096

;; ANSWER SECTION:
. 31679 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2015100701
1800 900 604800 86400
. 31679 IN RRSIG SOA 8 0 86400 20151017170000 20151007160000 62530 .
GNVz7SQs [...]
```

## The /etc/hosts File



### Important

Do not modify the **/etc/hosts** file manually. If **/etc/hosts** has been modified, make sure its contents conform to the following rules.

The following is an example of a correctly configured **/etc/hosts** file. It properly lists the IPv4 and IPv6 localhost entries for the host, followed by the IdM server IP address and host name as the first entry. Note that the IdM server host name cannot be part of the **localhost** entry.

```
127.0.0.1 localhost.localdomain localhost
::1 localhost6.localdomain6 localhost6
192.0.2.1 server.example.com
2001:DB8::1111 server.example.com
```

## 2.4.3. Red Hat Directory Server

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

## 2.4.4. 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.

**Table 2.1. IdM Ports**

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



## Note

Do not be concerned that IdM uses port 389. Using it is safe because all communication with IdM is encrypted with GSSAPI.

In addition, IdM can listen on port 8080 and in some installations also on ports 8443 and 749. However, these three ports are only used internally: even though IdM keeps them open, they are not required to be accessible from outside. It is recommended that you do not open ports 8080, 8443, and 749 and instead leave them blocked by a firewall.

## Opening the Required Ports

Opening ports requires the **firewalld** service to be running. To start **firewalld** as well as to configure it to start automatically when the system boots:

```
[root@server ~]# systemctl start firewalld.service
[root@server ~]# systemctl enable firewalld.service
```



## Note

You can determine whether **firewalld** is currently running using the **systemctl status firewalld.service** command.

For example, to open one of the required ports in the default zone and make the change both permanent and runtime:

1. Run the **firewall-cmd** command with the **--permanent** option specified.

```
[root@server ~]# firewall-cmd --permanent --add-port=389/tcp
```

2. Changes made with **firewall-cmd --permanent** are not effective immediately. To ensure that the changes take place immediately, run **firewall-cmd** again, this time without **--permanent**.

```
[root@server ~]# firewall-cmd --add-port=389/tcp
```

If you added multiple ports, it is simpler to make the changes take place immediately by running the **firewall-cmd --reload** command, which makes the current permanent configuration become new runtime configuration.

```
[root@server ~]# firewall-cmd --reload
```

To open all the IdM required ports in the default zone and make the change both permanent and runtime:

1. Run the **firewall-cmd** command with the **--permanent** option specified.

```
[root@server ~]# firewall-cmd --permanent --add-port={80/tcp,443/tcp,389/tcp,636/tcp,88/tcp,464/tcp,53/tcp,88/udp,464/udp,53/udp,123/udp}
```



2. Reload the **firewall-cmd** configuration to ensure that the change takes place immediately.

```
[root@server ~]# firewall-cmd --reload
```

For more information on **firewalld** and on opening and closing ports on a system, see the [Red Hat Security Guide](#) or the `firewall-cmd(1)` man page.

### 2.4.5. 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.

An NTP server must be running in order for some domain tasks to function properly. These domain tasks include data replication between servers and replicas in the topology. Kerberos authentication does not work without precise timekeeping, either for server-to-server authentication or for the initiation of replication. **The IdM server does not have to host the NTP server, but it is strongly recommended. This is the default configuration.**

Running an NTP server on an IdM server installed on a virtual machine (VM) can lead to inaccurate time synchronization in some environments. To avoid potential problems, it is recommended that IdM servers being installed on a VM do not run an NTP server. To disable NTP for IdM, add the **--no-ntp** option to the **ipa-server-install** command when installing the IdM server on a VM to prevent an NTP server from being installed. For more information about the reliability of an NTP server run on a VM, see [the related Knowledgebase solution](#).

### 2.4.6. NSCD

It is recommended that NSCD is disabled in IdM deployments. Alternatively, if disabling NSCD is not possible, only enable NSCD for maps that SSSD does not cache. Both NSCD and the SSSD service perform caching, and problems can occur when systems use both services simultaneously. See [the System-Level Authentication Guide](#) for information on how to avoid clashes between NSCD and SSSD.

## Chapter 3. Installing an IdM Server

An *IdM server* is a domain controller; it defines and manages the IdM domain. Setting up an IdM server follows these basic steps:

1. Installing the necessary packages on the machine
2. Configuring the server through setup scripts

Multiple domain controllers can be set up within one domain for load-balancing and failover tolerance. These additional servers are *replicas* of the master IdM server. This chapter describes installing an IdM server. For information on installing replicas, see [Chapter 4, Setting up IdM Replicas](#).

### 3.1. The `ipa-server-install` utility

The *ipa-server* package is the only package required to install an IdM server. If you want to have the IdM server set up as a DNS server, which is strongly recommended, install the *ipa-server-dns* package before installing the IdM server. Use the **yum** utility to install the required packages, for example:

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

For information about the dependencies installed together with *ipa-server*, see [Section 2.3, “Software Requirements”](#). For information about DNS and why it is recommended to run a DNS server on the IdM server, see [Section 1.2.4, “Service Discovery: DNS”](#).



#### 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, and that the service is configured as described in [Section 2.4.2, “Host Name and DNS Configuration”](#).

Note that the primary DNS domain and Kerberos realm cannot be changed after installation.

After installing the packages, the server instance is created using the **ipa-server-install** utility, which starts the IdM server setup script.

If you run **ipa-server-install** without any options, the interactive setup prompts for all the basic required information. The setup script also offers default values for most of the settings. For an example of this procedure, see [Section 3.2.1, “Basic Interactive Installation”](#) for installing without integrated DNS services and [Section 3.2.4.1, “Installing with an Integrated DNS Service Interactively”](#) for installing an IdM-integrated DNS server.

Alternatively, you can add command-line arguments to **ipa-server-install**, which passes the settings directly to the setup script. Some advanced settings, such as choosing other than default CA configuration, can only be passed to **ipa-server-install** using arguments, because the setup script does not prompt for the information during the

basic interactive installation process. For examples of running **ipa-server-install** with various arguments, see [Section 3.2.2, “Basic Silent Non-Interactive Installation”](#), [Section 3.2.3, “Installing with Different CA Configurations”](#), or [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#).





## Note

The port numbers and directory locations used by IdM are all defined automatically, as described in [Section 2.4.4, “System Ports”](#) and [Chapter 25, Identity Management Files and Logs](#). You cannot change or customize these ports and directories.

The **ipa-server-install** options are versatile enough to be customized to the specific deployment environment to install and configure different services as needed, and they also allow the configuration process to be easily scripted. [Table 3.1, “ipa-server-install Options”](#) lists some of the common arguments used with **ipa-server-install**. For the full list, see the `ipa-server-install(1)` man page.

**Table 3.1. ipa-server-install Options**

Argument	Description
<b>- -hostname</b> = <i>host name</i>	The fully-qualified domain name of the IdM server machine.
<div>  <b>Important</b> </div> <p>The fully-qualified domain name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the host name cause DNS failures. Additionally, the host name must be all lower-case; no capital letters are allowed. For other recommended naming practices, see the <a href="#">Red Hat Enterprise Linux Security Guide</a>.</p>	
<b>-r</b> <i>realm_name</i>	The name of the Kerberos realm to create for the IdM domain.

Argument	Description
-n <i>domain_name</i>	The name of the primary DNS domain for this IdM installation.
	 <b>Warning</b> When defining the domain name, make sure to follow the requirements described in <a href="#">Section 2.4.2, “Host Name and DNS Configuration”</a> .
--subject= <i>subject_DN</i>	Sets the base element for the subject DN of the issued certificates. This defaults to <b>0=realm</b> .
-a <i>ipa_admin_password</i>	The password for the IdM administrator. This is used for the admin user to authenticate to the Kerberos realm.
-p <i>directory_manager_password</i>	The password for the superuser, <b>cn=Directory Manager</b> , for the LDAP service.
-P <i>kerberos_master_password</i>	The password for the KDC administrator. This is randomly generated if no value is given.
--idmax= <i>number</i>	Sets the range for IDs which can be assigned by the IdM server. See <a href="#">Section 10.8.2, “ID Range Assignments During Installation”</a> for more details.
--idstart= <i>number</i>	
--ip-address	Specifies the IP address of the server. When added to <b>ipa-server-install</b> , this option only accepts IP addresses associated with the local interface.
--setup-dns	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.
--forwarder= <i>forwarder</i>	Gives a DNS forwarder to use with the DNS service. To specify more than one forwarder, use this option multiple times.
--no-forwarders	Uses root servers with the DNS service instead of forwarders.
--no-reverse	Does <i>not</i> create a reverse DNS zone when the DNS domain is set up. Use this option if reverse DNS zones already exist on another DNS server.
	If you do not use this option, the installation script automatically configures reverse DNS.

## 3.2. Installation Procedure Descriptions and Examples

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. The following installation procedure descriptions and examples illustrate how to use some common options during server installation.

These procedures and examples are not mutually exclusive; it is possible to use CA options, DNS options, and IdM configuration options in the same server invocation. Examples in the following sections are called out separately simply to make it more clear what each configuration area requires.

### 3.2.1. Basic Interactive Installation

If you run the **ipa-server-install** command without any arguments, the setup script automatically prompts you for the basic required information.

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

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

The installation process suggests default values for most of the configuration settings. The default values are displayed in brackets ([ ]), and you can choose them by pressing the **Enter** key.

2. The script prompts to configure an integrated DNS service. In this example, the default **no** option is chosen, meaning the installed IdM server will not run a DNS server.

```
Do you want to configure integrated DNS (BIND)? [no]:
```



#### Note

For an example that describes installing the DNS services, see [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#).

3. Enter the host name. The default value is determined automatically using reverse DNS.

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

4. Enter the domain name. The default value is determined automatically based on the host name.

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

5. Enter the new Kerberos realm name. The default value is based on the domain name.

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

6. Enter the password for the Directory Server superuser, **cn=Directory Manager**. No default value is available for this setting.

Directory Manager password:

7. Enter the password for the IdM system user account, **admin**, which will be created on the machine. No default value is available for this setting.

IPA admin password:

8. The script reprints the host name, IP address, and domain name. Confirm that the displayed information is correct by entering **yes**.

```
The IPA Master Server will be configured with
Hostname:      ipaserver.example.com
IP address:    192.168.1.1
Domain name:   example.com
Realm name:    EXAMPLE.COM
Continue to configure the system with these values? [no]: yes
```

9. The script now configures all of the associated services for IdM. Wait for the operation to complete.

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

```
Configuring NTP daemon (ntpd)
  [1/4]: stopping ntpd
  [2/4]: writing configuration
  [3/4]: configuring ntpd to start on boot
  [4/4]: starting ntpd
Done configuring NTP daemon (ntpd).
Configuring directory server (dirsrv): Estimated time 1 minute
  [1/38]: creating directory server user
```

...

```
Restarting the directory server
Restarting the KDC
Restarting the certificate server
Sample zone file for bind has been created in
/tmp/sample.zone.2yv_RI.db
Restarting the web server
```

```
=====
=====
```

Setup complete

10. The install script produces a DNS zone file with records: the **/tmp/sample.zone.2yv\_RI.db** file in the example output in the previous step. Add these records to the existing DNS servers.

Note that the server installation is not complete until the DNS records are added to the existing DNS servers.

11. The script recommends you to back up the CA certificate and to make sure the required network ports are open. For information about IdM port requirements and instructions on how to open these ports, see [Section 2.4.4, “System Ports”](#).
12. Authenticate to the Kerberos realm using the admin credentials to ensure that the user is properly configured and the Kerberos realm is accessible.

```
[root@server ~]# kinit admin
```

13. 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
UID: 939000000
GID: 939000000
Account disabled: False
Password: True
Kerberos keys available: True
-----
Number of entries returned 1
-----
```

### 3.2.2. Basic Silent Non-Interactive Installation

To allow automated and unattended configuration, pass the following basic required settings directly with the **ipa-server-install** utility:

- ✱ the **-r** option gives the Kerberos realm name
- ✱ the **-p** option gives the Directory Manager (DM) password; DM is the Directory Server super user
- ✱ the **-a** option gives the password for the IdM administrator

The **-U** option forces the installation to run unattended without requiring user interaction.

After you run **ipa-server-install** with these options, the setup script chooses default values for other settings, for example for the fully-qualified DNS name of the server or for the DNS domain name. You can supply custom values for the other settings by adding additional options to **ipa-server-install**. For more information about available arguments, see [Table 3.1, “ipa-server-install Options”](#) or the `ipa-server-install(1)` man page.



#### Note

If you pass these settings with **ipa-server-install**, the installed IdM server will not run a DNS server. For an example that describes installing the DNS services, see [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#).

**Example 3.1. Basic Installation without Interaction**

1. Run the **ipa-server-install** utility, providing the required settings.

```
[root@server ~]# ipa-server-install -r EXAMPLE.COM -p
DM_password -a admin_password -U
```

2. The setup script now configures all of the associated services for IdM. Wait for the operation to complete.

```
The log file for this installation can be found in
/var/log/ipaserver-install.log
=====
=====
This program will set up the IPA Server.

This includes:
  * Configure a stand-alone CA (dogtag) for certificate
management
  * Configure the Network Time Daemon (ntpd)
  * Create and configure an instance of Directory Server
  * Create and configure a Kerberos Key Distribution Center
(KDC)
  * Configure Apache (httpd)

The IPA Master Server will be configured with:
Hostname:      ipaserver.example.com
IP address(es): 2620:52:0:222f:21a:4aff:fe22:2114
Domain name:   example.com
Realm name:    EXAMPLE.COM

Configuring NTP daemon (ntpd)
[1/4]: stopping ntpd
[2/4]: writing configuration

...

Done.
Restarting the directory server
Restarting the KDC
Restarting the certificate server
Sample zone file for bind has been created in
/tmp/sample.zone._mS240.db
Restarting the web server
=====
=====
Setup complete

...
```

3. Complete the setup process and verify that everything is working as expected, as described in [Section 3.2.1, “Basic Interactive Installation”](#).



### 3.2.3. 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 IdM web UI, require server certificates to establish secure connections with each other.

In most deployments, a Red Hat Certificate System CA is installed with the IdM server. Certificate System uses a *CA signing certificate* to create and sign all of the server and user certificates created within the IdM domain. The CA signing certificate is itself required to be signed by the CA that issued it. The Certificate System CA signing certificate can be signed in two different ways:

#### The Certificate System is a *root CA*

The root CA is the highest CA in the CA hierarchy, and it is self-signed. If the Certificate System is a root CA, it can sign its own certificate. The root CA can also set its own certificate policies.

This is the default IdM configuration.

#### The Certificate System CA is signed by an externally-hosted CA

The Certificate System can be subordinate to an external CA in the CA hierarchy. The external CA can be a corporate CA or a third-party CA like Verisign or Thawte. In such deployments, the external CA is the root CA. The certificates issued within the IdM domain are potentially subject to restrictions set by the external root CA for attributes like the validity period.

Even when the root CA is an external CA, a Red Hat Certificate System instance is still used to issue all of the IdM domain certificates, that is, all of the IdM client and replica certificates. The only difference is that the initial CA certificate is not issued by the Certificate System CA but by a different CA.

Another configuration option is to install IdM without a CA.

#### A CA-less IdM installation

In very rare cases, it may not be possible to install certificate services with the IdM server. In such situations, you can install IdM without an integrated Red Hat Certificate System instance, as long as all required certificates are created and installed independently.

Installing without a CA requires that all certificates used within the IdM domain be created, uploaded, and renewed manually. The additional maintenance burden might be sustainable in some environments because of other restrictions within the infrastructure. However, most deployments use an integrated Certificate System instance together with the **certmonger** utility to manage IdM domain certificates.

#### 3.2.3.1. Installing with an Internal Root CA

Having the Red Hat Certificate System as a root CA is the default configuration and no additional parameters or configuration steps are required when **ipa-server-install** is run. No additional arguments are required to be added to the **ipa-server-install** utility to install a Certificate System instance as the root CA, because the **ipa-server-install** setup script automatically configures the Certificate System CA.

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

The log file for this installation can be found in `/var/log/ipaserver-install.log`

```
=====
=====
```

This program will set up the IPA Server.

This includes:

- \* **Configure a stand-alone CA (dogtag) for certificate management**
- \* Configure the Network Time Daemon (ntpd)
- \* Create and configure an instance of Directory Server
- \* Create and configure a Kerberos Key Distribution Center (KDC)
- \* Configure Apache (httpd)

...

Configuring certificate server (pki-tomcatd): Estimated time 3 minutes 30 seconds

[1/27]: creating certificate server user

[2/27]: configuring certificate server instance

...



## Note

For detailed descriptions for installing an IdM server with an internal root CA, see [Section 3.2.1, “Basic Interactive Installation”](#) and [Section 3.2.2, “Basic Silent Non-Interactive Installation”](#).

### 3.2.3.2. Installing Using an External CA

To install an IdM server that uses an external CA, add the **--external-ca** option to the **ipa-server-install** utility. You are then required to submit the generated certificate request to the external CA and to load the CA certificate and the issued server certificate to complete the setup.



## Note

The following procedure shows using the **--external-ca** option in the interactive installation process, otherwise described in [Section 3.2.1, “Basic Interactive Installation”](#). You can use **--external-ca** also with the non-interactive installation that is described in [Section 3.2.2, “Basic Silent Non-Interactive Installation”](#).

1. Add the **--external-ca** to the **ipa-server-install** command.

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

The script configures the associated services for IdM, such as NTP and Directory Server, as usual. Wait for the operation to complete.

**Note**

The **ipa-server-install** utility can sometimes fail with the following error:

```
ipa          : CRITICAL failed to configure ca instance
Command '/usr/sbin/pkispawn -s CA -f /tmp/configuration_file'
returned non-zero exit status 1
Configuration of CA failed
```

This failure occurs when the **\*\_proxy** environmental variables are set. For a solution on how to fix this problem, see [Section 3.2.3.5, “Unsetting the \\*\\_proxy Environmental Variables”](#)

2. After the script completes the setup, it returns the location of the certificate signing request (CSR) in the **/root/ipa.csr** file and prints instructions for how to configure the IdM server to use an external CA.

```
...

Configuring certificate server (pki-tomcatd): Estimated time 3
minutes 30 seconds
[1/8]: creating certificate server user
[2/8]: configuring certificate server instance
The next step is to get /root/ipa.csr signed by your CA and re-run
/sbin/ipa-server-install as: /sbin/ipa-server-install --external-
cert-file=/path/to/signed_certificate --external-cert-
file=/path/to/external_ca_certificate
```

The CSR is a request for a CA signing certificate for the IdM server so that the server can issue certificates within the IdM domain.

3. Submit the CSR located in **/root/ipa.csr** to the external CA. The process differs depending on the service to be used as the external CA.

**Important**

It might 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.

4. Retrieve the issued certificate and the CA certificate chain for the issuing CA in a base 64-encoded blob (either a PEM file or a Base\_64 certificate from a Windows CA). Again, the process differs for every certificate service. Usually, a download link on a web page or in the notification email allows the administrator to download all the required certificates.

**Important**

Be sure to get the full certificate chain for the CA, not just the CA certificate.

5. Run **ipa-server-install** again, this time specifying the locations and names of the newly-issued CA certificate and the CA chain files. For example:

```
[root@server ~]# ipa-server-install --external-cert-
file=/tmp/servercert20110601.pem --external-cert-
file=/tmp/cacert.pem
```

6. Complete the setup process and verify that everything is working as expected, as described in [Section 3.2.1, “Basic Interactive Installation”](#).

### 3.2.3.3. Installing without a CA

A CA-less installation requires you to provide:

- » An LDAP server certificate and a private key
- » An Apache server certificate and a private key
- » Full CA certificate chain of the CA that issued the LDAP and Apache server certificates

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 Red Hat Certificate System instance:

- » **certmonger** is not used to track certificates, so there is no expiration warning.
- » It is not possible 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.

Four or five options are required with the **ipa-server-install** or **ipa-replica-prepare** commands when installing without a CA to pass the necessary certificates directly to the setup process:

- » LDAP server certificate and a private key
  - **--dirsrv-cert-file** gives the certificate and private key files for the LDAP server certificate
  - **--dirsrv-pin** gives the password to access the private key in the files specified in **--dirsrv-cert-file**
- » Apache server certificate and private key

- **--http-cert-file** gives the certificate and private key files for the Apache server certificate
- **--http-pin** gives the password to access the private key in the files specified in **--http-cert-file**
- ✧ Full CA certificate chain of the CA that issued the server LDAP and Apache certificates
  - **--dirsrv-cert-file** and **--http-cert-file** can give certificate files with the full CA certificate chain or a part of it
  - **--ca-cert-file** gives certificate files to complete the full CA certificate chain, if needed



### Note

These five options are incompatible with the **--external-ca** option.

The **--dirsrv-cert-file** and **--http-cert-file** options can be specified multiple times. They accept:

- ✧ PEM-encoded and DER-encoded X.509 certificate files
- ✧ PKCS#1 and PKCS#8 private key files
- ✧ PKCS#7 certificate chain files
- ✧ PKCS#12 files

The **--ca-cert-file** option can also be specified multiple times. It accepts:

- ✧ PEM-encoded and DER-encoded X.509 certificate files
- ✧ PKCS#7 certificate chain files

The files provided using **--dirsrv-cert-file** and **--http-cert-file** must contain exactly one server certificate and exactly one private key. The files provided using **--dirsrv-cert-file** and **--http-cert-file** combined with the files provided using **--ca-cert-file** must contain the full CA certificate chain of the CA that issued the LDAP and Apache server certificates.



### Note

The content of the files provided using **--dirsrv-cert-file** and **--http-cert-file** is often identical.

## Example 3.2. Installing Identity Management Without a CA

Run **ipa-server-install** and pass the required certificates by specifying the **--http-cert-file**, **--http-pin**, **--dirsrv-cert-file**, **--dirsrv-pin** options, and if needed also **--ca-cert-file**. For example:

```
[root@server ~]# ipa-server-install --http-cert-file /tmp/server.crt -
--http-cert-file /tmp/server.key --http-pin secret --dirsrv-cert-file
/tmp/server.crt --dirsrv-cert-file /tmp/server.key --dirsrv-pin secret
--ca-cert-file ca.crt
```



## Note

Earlier versions of Identity Management required you to supply the **--root-ca-file** option, specifying the PEM file of the root CA certificate, during a CA-less installation. This is no longer necessary because the trusted CA is always the issuer of the DS and HTTP server certificates. IdM now automatically recognizes the root CA certificate from the certificates specified by **--dirsrv-cert-file**, **--http-cert-file**, and **--ca-cert-file**.

Both the **--root-ca-file** option and the other options used for a CA-less installation in earlier versions of IdM still work to ensure backward compatibility.

### 3.2.3.4. Installing a CA Certificate Manually

The **ipa-cacert-manage** utility allows you to install a new certificate to IdM. It enables you to change the current certificate, for example when the certificate is nearing its validity expiration date.

To manually install a CA certificate:

1. Run the **ipa-cacert-manage install** command and specify the path to the file containing the certificate. The command accepts PEM certificate files. For example:

```
[root@server ~]# ipa-cacert-manage install /etc/group/cert.pem
```

The certificate is now present in the LDAP certificate store.

2. Run the **ipa-certupdate** utility, which updates client servers with the information about the new certificate from LDAP. You have to run **ipa-certupdate** on every client separately.



## Important

If you do not run **ipa-certupdate** after installing a certificate manually, the certificate will not be distributed to clients.

The **ipa-cacert-manage install** command can take the following options:

**-n**

gives the nickname of the certificate; the default value is the subject name of the certificate

**-t**

specifies the trust flags for the certificate in the **certutil** format; the default value is **C,,**. For information about the format in which to specify the trust flags, see the `ipa-cacert-manage(1)` man page.

### 3.2.3.5. Unsetting the \*\_proxy Environmental Variables

The **\*\_proxy** environmental variables can cause the **ipa-server-install --external-ca** command to fail with the following error:

```
ipa          : CRITICAL failed to configure ca instance Command
'/usr/sbin/pkispawn -s CA -f /tmp/configuration_file' returned non-zero
exit status 1
Configuration of CA failed
```

If you experience this error, determine whether the variables are causing it by using the **env** utility:

```
env|grep proxy
http_proxy=http://example.com:8080
ftp_proxy=http://example.com:8080
https_proxy=http://example.com:8080
```

If running **env|grep proxy** returns variables such as the above, follow these steps to solve the problem:

1. Use the following shell script to unset the **\*\_proxy** environmental variables:

```
# for i in ftp http https; do unset ${i}_proxy; done
```

2. Run the **pkidestroy** utility to remove the unsuccessful CA subsystem installation:

```
# pkidestroy -s CA -i pki-tomcat; rm -rf /var/log/pki/pki-tomcat
/etc/sysconfig/pki-tomcat /etc/sysconfig/pki/tomcat/pki-tomcat
/var/lib/pki/pki-tomcat /etc/pki/pki-tomcat /root/ipa.csr
```

3. Remove the failed IdM server installation:

```
# ipa-server-install --uninstall
```

After this, run **ipa-server-install --external-ca** again.

### 3.2.4. Configuring DNS Services within the IdM Domain

IdM can be installed with its integrated DNS server. Installing an IdM-integrated DNS server is recommended for convenience. When installing IdM with an integrated DNS server, make sure to follow the prerequisites described in [Section 2.4.2, “Host Name and DNS Configuration”](#).

**Note**

You can also install DNS services into an existing IdM server using the **ipa-dns-install** utility. Therefore, if you install an IdM server without integrated DNS, you can add DNS services later. For more information, see [Section 15.1, “Installing DNS Services Into an Existing Server”](#).

**3.2.4.1. Installing with an Integrated DNS Service Interactively**

To instruct the **ipa-server-install** setup script to configure the IdM server as a DNS server interactively, add the **--setup-dns** option to **ipa-server-install**. During the interactive installation, the setup script prompts you for the required DNS information.

Prior to running the setup script, decide whether you want to use DNS forwarding. If you are unsure, see [Section 15.7, “Managing DNS Forwarding”](#) before continuing. If you decide to use DNS forwarding, open the **/etc/resolv.conf** file and note the IP addresses in the file; you will need these addresses later to supply them as DNS forwarders during installation.

1. Run the **ipa-server-install** utility with the **--setup-dns** option. The script displays a list of services to be installed, including DNS.

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

2. The script prompts to overwrite the existing BIND configuration. Enter **yes** for the installation to proceed.

```
Existing BIND configuration detected, overwrite? [no]: yes
```

3. The script prompts for several required settings. Providing them is described in [Section 3.2.1, “Basic Interactive Installation”](#).
4. The script then prompts for DNS forwarders.

```
Do you want to configure DNS forwarders? [yes]:
```

Answer **yes** to configure the DNS forwarders. When prompted to specify the forwarders, provide the IP addresses from the **/etc/resolv.conf** file. Note that the forwarder IP addresses will be added to the **/etc/named.conf** file on the installed IdM server as global forwarders with the **forward first** policy.

If you do not want to use DNS forwarding, enter **no**.

5. The script then prompts for the reverse DNS zone. Only create the reverse zone if it does not exist on another DNS server.

To create a reverse zone, enter **yes**, and then specify the reverse zone name. If you do not want to create a reverse zone, enter **no**.

```
Do you want to configure the reverse zone? [yes]:
Please specify the reverse zone name [2.0.192.in-addr.arpa.]:
Using reverse zone 2.0.192.in-addr.arpa.
```



6. The script displays the configuration settings you provided and prompts for confirmation. Enter **yes** for the installation to proceed.

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

7. The script now configures the IdM server. Wait for the operation to complete.
8. Complete the **ipa-server-install** setup process and verify that everything is working as expected, as explained in [Section 3.2.1, “Basic Interactive Installation”](#).

### 3.2.4.2. Installing with an Integrated DNS Service Non-Interactively

To instruct the **ipa-server-install** setup script to configure the IdM server as a DNS server non-interactively, add the **--setup-dns** to **ipa-server-install**, and also provide the required DNS information to the **ipa-server-install** setup script:

#### DNS forwarders

To configure DNS forwarders, use the **--forwarder** option to add a DNS forwarder. To specify multiple forwarders, use **--forwarder** multiple times.

If you want no external forwarders to be used with the IdM DNS service, add the **--no-forwarders** option, indicating that only root DNS servers are used.



#### Note

Either **--forwarder** or **--no-forwarders** is always required.

#### reverse DNS zones

By default, the script automatically configures a default value for the reverse DNS zone.

To instruct **ipa-server-install** not to create a reverse DNS zone, use the **--no-reverse** option.

These options provide the required DNS settings directly to the **ipa-server-install** setup script, which ensures that the script does not prompt for the information during the installation process.



#### Note

In the following procedure, only the DNS settings are provided to **ipa-server-install**, not the other IdM server settings. Therefore, the script in the procedure still requires some input from the user.

To achieve a completely automated and unattended installation, also provide the required IdM server settings directly to **ipa-server-install**, as described in [Section 3.2.2, “Basic Silent Non-Interactive Installation”](#).

To install an IdM server with an integrated DNS service non-interactively, providing the required DNS information directly to the setup script:

1. Run the **ipa-server-install** utility with the **--setup-dns** option, and add any additional options that are required to pass the DNS settings. The script displays a

```
[root@server ~]# ipa-server-install --setup-dns --
forwarder=1.2.3.0 --forwarder=1.2.255.0 --no-reverse
```

```
The log file for this installation can be found in
/var/log/ipaserver-install.log
```

```
=====
```

```
This program will set up the IPA Server.
```

```
This includes:
```

- \* Configure a stand-alone CA (dogtag) for certificate management
- \* Configure the Network Time Daemon (ntpd)
- \* Create and configure an instance of Directory Server
- \* Create and configure a Kerberos Key Distribution Center (KDC)
- \* Configure Apache (httpd)
- \* **Configure DNS (bind)**

```
...
```

2. The script prompts to overwrite the existing BIND configuration. Enter **yes** for the installation to proceed.

```
Existing BIND configuration detected, overwrite? [no]: yes
```

3. The script then prompts for several settings required to configure the IdM server. Providing them is described in [Section 3.2.1, “Basic Interactive Installation”](#).
4. The script displays the configuration settings you provided and prompts for confirmation. Enter **yes** for the installation to proceed.

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

5. The script now configures the IdM server. Wait for the operation to complete.
6. Complete the setup process and verify that everything is working as expected, as described in [Section 3.2.1, “Basic Interactive Installation”](#).

## Chapter 4. Setting up IdM Replicas

Replicas are created by cloning the configuration of existing Identity Management servers; therefore, servers and their replicas share identical core configuration. The replica installation process consists of two phases:

1. Copying the existing server configuration
2. Installing the replica based on the copied configuration

Maintaining several server replicas is a recommended backup solution to avoid data loss, as described in the ["Backup and Restore in IdM/IPA" Knowledgebase solution](#).



### Note

Another backup solution, recommended primarily for situations when rebuilding the IdM deployment from replicas is not possible, is the **ipa-backup** utility, as described in [Chapter 9, Backing Up and Restoring Identity Management](#).

### 4.1. Planning the Server and Replica Topologies

Three types of machines exist in the IdM domain:

#### Servers

Servers manage all of the services used by domain members.

#### Replicas

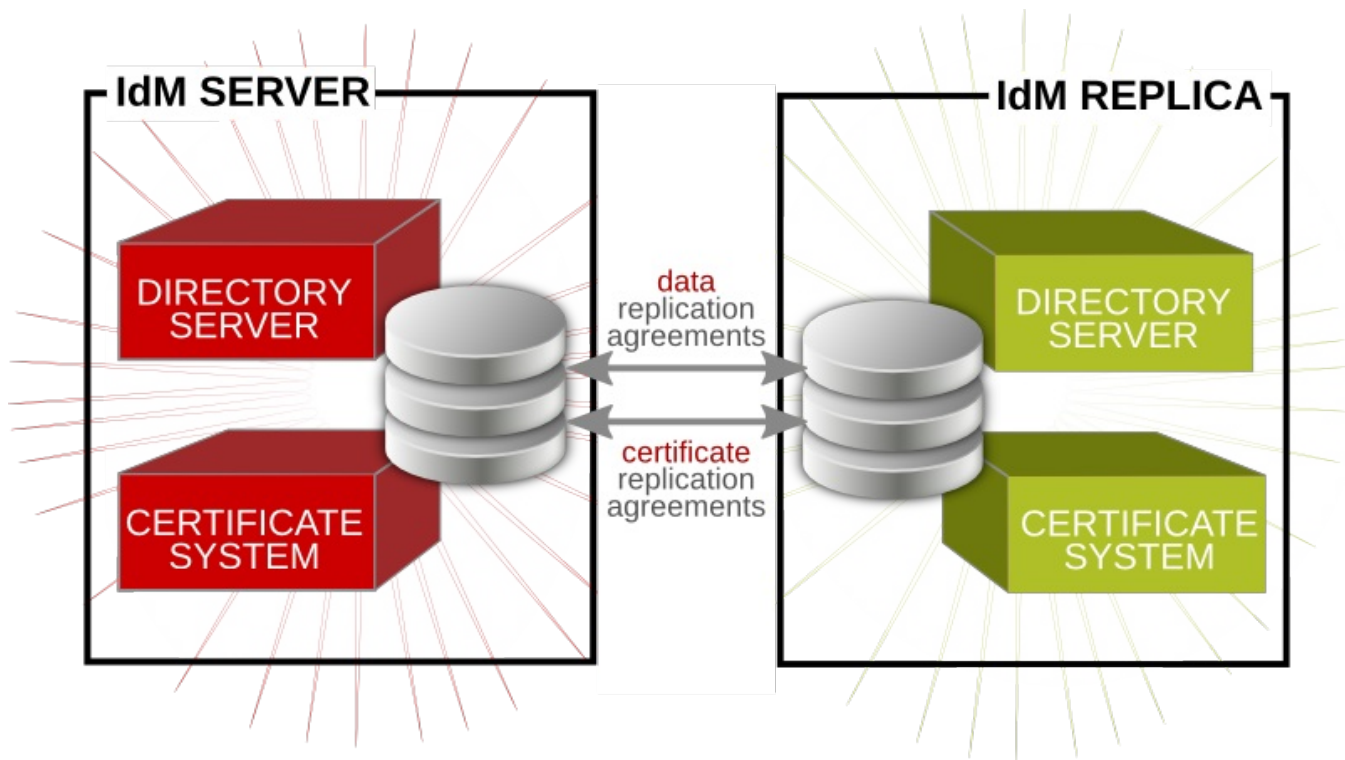
Replicas are copies of servers. Once a replica is installed, it is functionally identical to a server.

#### Clients

Clients, which belong to the Kerberos domains, receive certificates and tickets issued by the servers, and use other centralized services for authentication and authorization.

Servers and replicas created from these servers share the same internal information about users, machines, certificates, and configured policies. This data is copied from the server to the replica in a process called *replication*.

An IdM server uses a single Directory Server instance. The instance is used by the IdM server as a data store and by the Red Hat Certificate System to store certificate information. During replication, this instance is replicated over to corresponding consumer Directory Server instance used by the IdM replica, with replication agreements managed separately for the realm data and for the certificate data.



**Figure 4.1. Server and Replica Agreements**

We recommend that you follow these guidelines when planning your server and replica topology:

- ✧ Configure no more than four replication agreements on a single server or replica.
- ✧ Do not involve more than 20 servers and replicas in a single Identity Management domain.
- ✧ Configure a minimum of two replication agreements for every server or replica. This ensures that no orphan servers or replicas are 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 and replicas with a small number of servers in a cell. Each of these cell is a *tight cell*, meaning that all the servers inside have replication agreements with each other. In addition, each server has one replication agreement with another server *outside* the cell, loosely coupling that cell to every other cell in the overall domain.

To accomplish this resilient cell topology, you can follow these recommendations:

- ✧ 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 back end.

## 4.2. Prerequisites for Installing a Replica Server

The installation requirements and packages for replicas are the same as for IdM servers. Make sure that the machine on which replicas is to be installed meets all of the prerequisites listed in [Chapter 2, Prerequisites for Installation](#).

In addition to the general server requirements, the following conditions must also be met when installing a replica:

### The replica must be running the same or later version of IdM

For example, if the master server is running on Red Hat Enterprise Linux 7 and uses the IdM 4.1 packages, then the replica must also run on Red Hat Enterprise Linux 7 or later and use IdM version 4.1 or later. This ensures that configuration can be properly copied from the server to the replica.



#### Important

IdM does not support creating a replica of an earlier version than the version of the master. If you try to create a replica using an earlier version, the installation fails during the attempt to configure the Red Hat Directory Server instance.

### The replica requires additional ports to be open

In addition to the standard IdM server port requirements described in [Section 2.4.4, “System Ports”](#), make sure the following port requirements are complied as well:

- During the replica setup process, keep the *TCP port 22* open. This port is required in order to use SSH to connect to the master server.
- If one of the servers is running Red Hat Enterprise Linux 6 and has a CA installed, keep also *TCP port 7389* open during and after the replica configuration. In a purely Red Hat Enterprise Linux 7 environment, port 7389 is not required.



#### Note

The **ipa-replica-install** script includes the **ipa-replica-conncheck** utility that verifies the status of the required ports. You can also run **ipa-replica-conncheck** separately for troubleshooting purposes. For information on how to use the utility, see the `ipa-replica-conncheck(1)` man page.

For information on how to open ports using the **firewall-cmd** utility, see [Section 2.4.4, “System Ports”](#).

### If the replica manages certificate requests, it must use the same CA configuration as the server

For example, if the server is its own root CA (using Red Hat 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; and if the server was installed without a CA by providing the required certificates manually, the same certificates must be provided when installing the replica.

## 4.3. Creating the Replica

The package requirements for IdM replicas are the same as for IdM servers:

- ✱ the *ipa-server* package
- ✱ the *ipa-server-dns* package if you want the replica to also host DNS services

During the replica creation process, the **ipa-replica-prepare** utility creates a *replica information file* named after the replica server in the `/var/lib/ipa/` directory. The replica information file is a GPG-encrypted file containing realm and configuration information for the master server.

The **ipa-replica-install** replica setup script configures a Directory Server instance based on the information contained in the replica information file and initiates the *replica initialization* process, during which the script copies over data from the master server to the replica. A replica information file can only be used to install a replica on the specific machine for which it was created. It cannot be used to create multiple replicas on multiple machines.

While much of the core configuration of the replica is identical to the configuration of the initial server, such as the realm name and directory settings, services on the replica and on the server are not required to match: the replica does not have to manage the same services as the server. For example, it is possible to install a replica without DNS from a server that runs the DNS services or to install a replica without a CA or without NTP.



### Note

The following procedures and examples are not mutually exclusive; it is possible to use the CA, DNS, and other configuration options simultaneously. Examples in the following sections are called out separately simply to make it more clear what each configuration area requires.

### 4.3.1. Installing a Replica without DNS

1. On the master IdM server, run the **ipa-replica-prepare** utility and add the fully-qualified domain name (FQDN) of the *replica* machine. Note that the **ipa-replica-prepare** script does not validate the IP address or verify if the IP address of the replica is reachable by other servers.



### Important

The fully-qualified domain name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the host name cause DNS failures. Additionally, the host name must be all lower-case; no capital letters are allowed. For other recommended naming practices, see the [Red Hat Enterprise Linux Security Guide](#).

If the master server is configured with integrated DNS, specify the IP address of the replica machine using the **--ip-address** option. The installation script then asks if you want to configure the reverse zone for the replica. Only pass **--ip-**

**address** if the IdM server was configured with integrated DNS. Otherwise, there is no DNS record to update, and the attempt to create the replica fails when the DNS record operation fails.

Enter the initial master server's Directory Manager (DM) password when prompted. The output of **ipa-replica-prepare** displays the location of the replica information file. For example:

```
[root@server ~]# ipa-replica-prepare replica.example.com --ip-
address 192.0.2.0
Directory Manager (existing master) password:

Do you want to configure the reverse zone? [yes]: no
Preparing replica for replica.example.com from server.example.com
Creating SSL certificate for the Directory Server
Creating SSL certificate for the dogtag Directory Server
Saving dogtag Directory Server port
Creating SSL certificate for the Web Server
Exporting RA certificate
Copying additional files
Finalizing configuration
Packaging replica information into /var/lib/ipa/replica-info-
replica.example.com.gpg
Adding DNS records for replica.example.com
Waiting for replica.example.com. A or AAAA record to be resolvable
This can be safely interrupted (Ctrl+C)
The ipa-replica-prepare command was successful
```



### Warning

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

For other options that can be added to **ipa-replica-prepare**, see the `ipa-replica-prepare(1)` man page.

2. *On the replica machine*, install the *ipa-server* package.

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

3. Copy the replica information file from the initial server to the replica machine:

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

4. *On the replica machine*, run the **ipa-replica-install** utility and add the location of the replication information file to start the replica initialization process. Enter the original master server's Directory Manager and admin passwords when prompted, and wait for the replica installation script to complete.

```
[root@replica ~]# ipa-replica-install /var/lib/ipa/replica-info-
replica.example.com.gpg
Directory Manager (existing master) password:
```



```

Run connection check to master
Check connection from replica to remote master
'server.example.com':

...

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@MASTER.EXAMPLE.COM password:

Check SSH connection to remote master

...

Connection from master to replica is OK.

...

Configuring NTP daemon (ntpd)
  [1/4]: stopping ntpd
  [2/4]: writing configuration

...

Restarting Directory server to apply updates
  [1/2]: stopping directory server
  [2/2]: starting directory server
Done.
Restarting the directory server
Restarting the KDC
Restarting the web server

```



### Note

If the replica file being installed does not match the current host name, the replica installation script displays a warning message and asks for confirmation. In some cases, such as on multi-homed machines, you can confirm to continue with the mismatched host names.

For command-line options that can be added to **ipa-replica-install**, see the **ipa-replica-prepare(1)** man page. Note that one of the options **ipa-replica-install** accepts is the **--ip-address** option. When added to **ipa-replica-install**, **--ip-address** only accepts IP addresses associated with the local interface.

## 4.3.2. Installing a Replica with DNS

To install a replica with integrated DNS, follow the procedure for installing without DNS described in [Section 4.3.1, “Installing a Replica without DNS”](#), but add the following options to the **ipa-replica-install** utility:

```
--setup-dns
```



The **--setup-dns** option creates a DNS zone if it does not exist already and configures the replica as the DNS server.

### **--forwarder or --no-forwarders**

To specify a DNS forwarder, use the **--forwarder** option. To specify multiple forwarders, use **--forwarder** multiple times.

If you do not want to specify any forwarders, use the **--no-forwarders** option.

For example:

```
[root@replica ~]# ipa-replica-install /var/lib/ipa/replica-info-  
replica.example.com.gpg --setup-dns --forwarder 198.51.100.0
```



### **Note**

You can set up the replica to serve as the DNS server even if the initial master server was not installed with integrated DNS.

The **ipa-replica-install** utility accepts a number of other options related to DNS settings, such as the **--no-reverse** or **--no-host-dns** options. For more information about them, see the `ipa-replica-install(1)` man page.

If you install a replica without DNS, you can set it up as the DNS server later using the **ipa-dns-install** utility, as described in [Section 15.1, “Installing DNS Services Into an Existing Server”](#), and add the DNS records manually, as described in [Section 15.5.4, “Adding Records to DNS Zones”](#).

## **Verifying the DNS Records**

After installing a new replica, you can make sure that proper DNS entries were created so that IdM clients can discover the new server. The following DNS records are necessary for required domain services:

- » **\_ldap.\_tcp**
- » **\_kerberos.\_tcp**
- » **\_kerberos.\_udp**
- » **\_kerberos-master.\_tcp**
- » **\_kerberos-master.\_udp**
- » **\_ntp.\_udp**
- » **\_kpasswd.\_tcp**
- » **\_kpasswd.\_udp**

If the initial IdM server was created with DNS enabled, then the replica is automatically created with the proper DNS entries. To verify the entries are present, follow this example:

```
[root@replica ~]# DOMAIN=example.com
```

```
[root@ipareplica ~]# NAMESERVER=replica
[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
server1.example.com.
_ldap._tcp.example.com. 86400 IN SRV 0 100 389
server2.example.com.
_kerberos._tcp.example.com. 86400 IN SRV 0 100 88
server1.example.com.
...
```

### 4.3.3. Installing a Replica with Various CA Configurations

While an integrated Red Hat Certificate System CA instance or a CA-less server installation are required for master servers, they are only optional for replicas. A replica can be set up without the certificate services, in which case it forwards all requests for certificate operations to the initial master server.



#### Warning

If only one server in the whole IdM deployment has a CA installed, the CA configuration is lost if this server fails without any chance for recovery.

If you choose to set up a CA on the replica, the CA configuration on the replica must mirror the CA configuration of the server.

### Installing a replica from a server with a Certificate System CA installed

To set up a CA on the replica when the initial server was configured with an integrated Red Hat Certificate System instance (regardless of whether it was a root CA or whether it was subordinate to an external CA), follow the basic installation procedure described in [Section 4.3.1, “Installing a Replica without DNS”](#), but add the **--setup-ca** option to the **ipa-replica-install** utility. The **--setup-ca** option copies the CA configuration from the initial server's configuration.

```
[root@replica ~]# ipa-replica-install /var/lib/ipa/replica-info-
replica.example.com.gpg --setup-ca
```

### Installing a replica from a server without a Certificate System CA installed

For a CA-less replica installation, follow the basic procedure described in [Section 4.3.1, “Installing a Replica without DNS”](#), but add the following options when running the **ipa-replica-prepare** utility on the initial server:

- \* **--dirsrv-cert-file**
- \* **--dirsrv-pin**

- ✧ **--http-cert-file**

- ✧ **--http-pin**

Do not add the **--ca-cert-file** option to **ipa-replica-prepare**; the utility takes this part of the certificate information automatically from the master server. For detailed information about the files that are provided using these four options, see [Section 3.2.3.3, “Installing without a CA”](#).

For example:

```
[root@server ~]# ipa-replica-prepare replica.example.com --dirsrv-cert-file /tmp/server.key --dirsrv-pin secret --http-cert-file /tmp/server.crt --http-cert-file /tmp/server.key --http-pin secret --dirsrv-cert-file /tmp/server.crt
```

#### 4.3.4. Installing a Replica with Other Settings

The **ipa-replica-install** utility accepts a number of other configuration options, such as:

- ✧ **--no-ntp** specifies that the replica is configured without the NTP service
- ✧ **--no-ssh** specifies that no OpenSSH client is configured on the replica
- ✧ **--no-sshd** specifies that the replica is configured without the OpenSSH server

For a complete list of the **ipa-replica-install** configuration options, see the `ipa-replica-install(1)` man page.

#### 4.3.5. Testing the New Replica

To verify that replication works after creating a new replica, you can create a user on one of the servers and then make sure the user is visible on the other server. For example:

```
[root@master_server ~]$ ipa user-add test_user --first=Test --last=User
[root@replica_server ~]$ ipa user-show test_user
```

### 4.4. Adding Additional Replication Agreements

Installing a replica using **ipa-replica-install** creates an initial replication agreement between the master server and the replica. To connect the replica to other servers or replicas, add additional agreements using the **ipa-replica-manage** utility.

If the master server and the new replica have a CA installed, a replication agreement for CA is also created. To add additional CA replication agreements to other servers or replicas, use the **ipa-cs-replica-manage** utility.

For more information on creating replication agreements, see [Section 29.1.5, “Creating Replication Agreements”](#).

## 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 System-Level Authentication 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. Opening the IdM Required System Ports

IdM uses a number of ports to communicate with its services. These ports must be open and available for IdM to work. For more information on which ports IdM requires, see [Section 2.4.4, “System Ports”](#).

Opening ports requires the **firewalld** service to be running. To start **firewalld** as well as to configure it to start automatically when the system boots:

```

[root@server ~]# systemctl start firewalld.service
[root@server ~]# systemctl enable firewalld.service

```

To open all the IdM required ports in the default zone and make the change both permanent and runtime:

1. Run the **firewall-cmd** command with the **--permanent** option specified.

```

[root@server ~]# firewall-cmd --permanent --add-port=
{80/tcp,443/tcp,389/tcp,636/tcp,88/tcp,464/tcp,53/tcp,88/udp,464/udp,53/udp,123/udp}

```

2. Reload the **firewall-cmd** configuration to ensure that the change takes place immediately.

```

[root@server ~]# firewall-cmd --reload

```

## 5.3. Configuring a 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.

### 5.3.1. Installing the Client (Full Example)

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:

```
[root@client ~]# yum install ipa-client
```

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

```
[root@client ~]# yum install ipa-client ipa-admintools
```

2. Employ proper DNS delegation, and do not alter **resolv.conf** on clients.



#### Note

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

3. Run the **ipa-client-install** command, which sets up the IdM client.

The command automatically sets a NIS domain name to the IdM domain name by default. To configure the client without setting a NIS domain name, add the **--no-nisdomain** option. To specify a custom NIS domain name, specify it using the **--nisdomain** option.

The command also automatically configures the SSSD service as the data provider for the sudo service by default. To avoid this, add the **--no-sudo** option.

To update DNS with the client machine's IP address, add the **--enable-dns-updates** option. You should only use **--enable-dns-updates** 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.

For information about other options that you can use with **ipa-client-install**, see the `ipa-client-install(1)` man page.

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



### Important

The fully-qualified domain name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the host name cause DNS failures. Additionally, the host name must be all lower-case; no capital letters are allowed. For other recommended naming practices, see the [Red Hat Enterprise Linux Security Guide](#).

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]: y
User authorized to enroll computers: admin
Synchronizing time with KDC...
Password for admin@EXAMPLE.COM:
Successfully retrieved CA cert
Subject: CN=Certificate Authority,O=EXAMPLE.COM
Issuer: CN=Certificate Authority,O=EXAMPLE.COM
Valid From: Tue Aug 13 09:29:07 2015 UTC
Valid Until: Sat Aug 13 09:29:07 2033 UTC
Enrolled in IPA realm EXAMPLE.COM
Created /etc/ipa/default.conf
New SSSD config will be created
Configured /etc/sss/sss.conf
Configured /etc/krb5.conf for IPA realm EXAMPLE.COM
Adding SSH public key from /etc/ssh/ssh_host_rsa_key.pub
Adding SSH public key from /etc/ssh/ssh_host_dsa_key.pub
SSSD enabled
Configured /etc/openldap/ldap.conf
NTP enabled
Configured /etc/ssh/ssh_config
Configured /etc/ssh/sshd_config
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:

```
[jsmith@client ~]$ id
[jsmith@client ~]$ getent passwd admin
[jsmith@client ~]$ getent group admins
```

8. Run the **ipa-client-automount** command which automatically configures NFS for IdM. For more information on **ipa-client-automount**, see [Section 16.2.1, “Configuring NFS Automatically”](#).

### 5.3.2. Examples of Other Client Installation Options

There are a number of different configuration options with the **ipa-client-install** command which can be used to configure the client system in different ways, depending on the infrastructure requirements.

#### Example 5.1. Enabling DNS Updates

Depending on the DHCP configuration, the IP addresses of clients can change with some regularity. If the IP address changes, this can cause discrepancies between the DNS records in the IdM server and the actual IP addresses in use, which could affect policies set within IdM and communications between clients and services.

The **--enable-dns-updates** option sets the System Security Services Daemon to update the DNS entries whenever the IP address for a client changes.

```
[root@client ~]# ipa-client-install --enable-dns-updates
```

#### Example 5.2. Specifying Domain Information

When just running the client installation command, the script prompts for required IdM domain information, including the name of an IdM server to register with, the DNS domain name, and the Kerberos realm and principal.

All of the basic information can be passed with the installation command (which is useful for automated installations).

- ✧ **--domain** for the DNS domain name (which is only used if the IdM server is configured to host DNS services)
- ✧ **--server** for the IdM server to register with (which can be any server or replica in the topology)



#### Important

The fully-qualified domain name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the host name cause DNS failures. Additionally, the host name must be all lower-case; no capital letters are allowed. For other recommended naming practices, see the [Red Hat Enterprise Linux Security Guide](#).

- ✧ **--realm** for the Kerberos realm name and, optionally, **-p** for a Kerberos principal name

```
[root@client ~]# ipa-client-install --domain EXAMPLE.COM --server  
ipaserver.example.com --realm EXAMPLE -p host/server.example.com
```



**Example 5.3. Setting a Specific IdM Server**

There can be multiple servers and replicas within the IdM server topology. When a client needs to connect to a server for updates or to retrieve user information, it (by default) uses a service scan to discover available servers and replicas in the domain. This means that the actual server to which the client connects is random, depending on the results of the discovery scan.

It is possible to set a specific server within the IdM domain which is used for client updates; if for some reason, connecting to that server fails, then the client can discover another server within the domain for failover.

The preferred server is set in the **--fixed-primary** option.

```
[root@client ~]# ipa-client-install --fixed-primary
ipaserver.example.com
```

**Example 5.4. Disabling System Authentication Tools**

Red Hat Enterprise Linux uses the **authconfig** tool to set and update authentication clients and settings for a local system. Identity Management uses the System Security Services Daemon (SSSD) to store IdM server configuration and to retrieve policy information, users, passwords, and groups configured within the IdM domain.

**It is strongly recommended that you use `authconfig` and `SSSD` to manage your user, group, and other IdM client configuration.**

There may be some situations where an administrator wants to disable dynamic changes to system authentication configuration. In that case, it is possible to disable IdM from making updates to **authconfig** or **SSSD**.

The **--noac** option prevents any changes through **authconfig**. The **--no-sssd** option prevents IdM from using **SSSD**.

```
[root@client ~]# ipa-client-install --noac --no-sssd
```

A related option is **--preserve-sssd**. While this allows the client to change the **SSSD** configuration file to configure the IdM domain, it saves the old **SSSD** configuration.

**Example 5.5. Disabling Password Caching**

One of the primary functions of **SSSD** is *password caching*. Normally, when a system uses an external password store, authentication fails if that password store is ever inaccessible. However, **SSSD** can cache passwords after a successful authentication attempt and store those passwords locally. This allows users to log in and access domain services (which they have previously accessed) even if the IdM server is inaccessible.

In highly-secure environments, it may be necessary to prevent password caching to prevent potentially unauthorized access. In that case, the **--no-krb5-offline-passwords** option can be used to prevent passwords from being cached in **SSSD**.

```
[root@client ~]# ipa-client-install --no-krb5-offline-passwords
```

## 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.

### 5.4.1. Setting up an IdM Client (Full Procedure)

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

```
[root@client ~]# yum install ipa-admintools
```

3. On the IdM server, create a host entry for the client.

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa host-add --force --ip-
address=192.168.166.31 ipaclient.example.com
```

Creating hosts manually is covered in [Section 5.4.2, “Other Examples of Adding a Host Entry”](#).

4. On the IdM server, set the client host to be managed by the server.

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa host-add-managedby --
hosts=ipaserver.example.com ipaclient.example.com
```

5. On the client, configure SSSD by editing the **/etc/sss/sss.conf** file to point to the IdM domain.

```
[root@client ~]# touch /etc/sss/sss.conf
[root@client ~]# vim /etc/sss/sss.conf

[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
```

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

```
[root@client ~]# vim /etc/nsswitch.conf

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

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

```
[root@client ~]# vim /etc/krb5.conf

[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
```

8. Generate the keytab for the client.

```
[root@client ~]# kinit admin
[root@client ~]# ipa-getkeytab -s ipaserver.example.com -p
host/ipaclient.example.com -k /etc/krb5.keytab
```

9. 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
```

✳ For **/etc/pam.d/smartcard-auth**:

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

10. Install the IdM server's CA certificate.

a. Obtain the certificate from the server.

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

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

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

```
[root@client ~]# systemctl start certmonger.service
```



### Note

You can use the **systemctl** utility to make the **certmonger** service start by default.

```
[root@client ~]# systemctl enable certmonger.service
```

- b. Use the **ipa-getcert** command, which creates and manages the certificate through **certmonger**. The options are described more in the **certmonger** manpage.

```
[root@client ~]# 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**.

12. Configure the NIS domain name for the client.

- a. Set the NIS domain name.

```
[root@client ~]# authconfig --nisdomain=example.org --update
```

- b. Restart the domain name service to apply the change.

```
[root@client ~]# systemctl restart rhel-domainname.service
```

Note that the NIS domain does not actually have to exist, and that it is not required to have a NIS server installed. For information about the NIS domain name requirements, see [Section 19.1.2, “sudo and Netgroups”](#).

13. Configure the **sudo** utility to be used with SSSD.

- a. Create the **[sudo]** section in the **/etc/sss/sss.conf** file. The section can stay empty.
- b. Add **sudo** to the list of services in the **[sss]** section in **/etc/sss/sss.conf**.

```
[root@client ~]# vim /etc/sss/sss.conf

[sss]
services = nss, pam, sudo
```

- c. Enable SSSD as a source for **sudo** rules by adding the following **sudors** entry to the **/etc/nsswitch.conf** file.

```
[root@client ~]# vim /etc/nsswitch.conf
```

```
...
sudoers: files sss
...
```

- d. Restart SSSD.

```
[root@client ~]# systemctl restart sssd.service
```

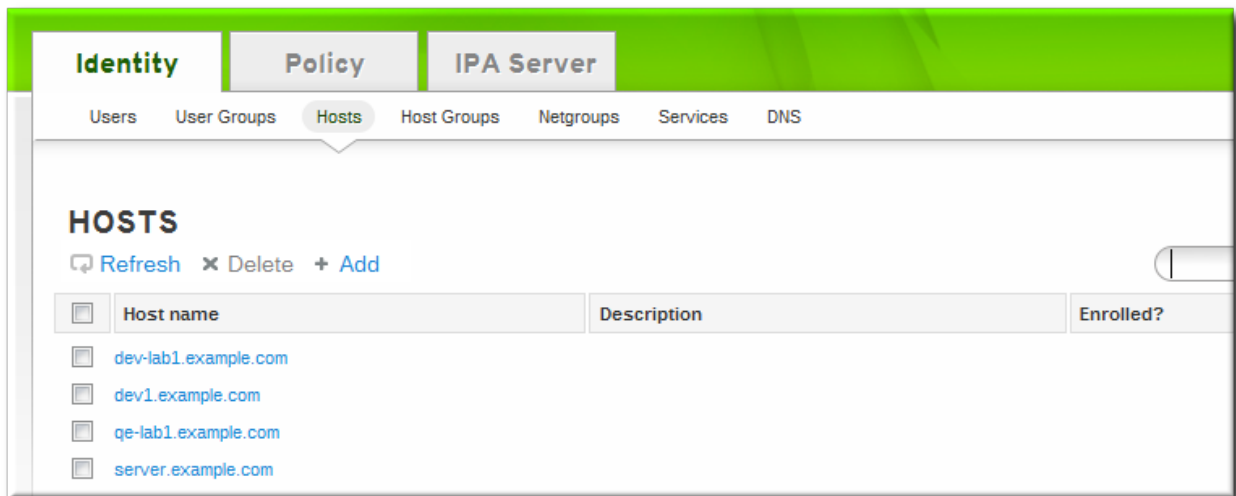
14. Run the **ipa-client-automount** command which automatically configures NFS for IdM. For more information on **ipa-client-automount**, see [Section 16.2.1, “Configuring NFS Automatically”](#).

## 5.4.2. Other Examples of Adding a Host Entry

[Section 5.4.1, “Setting up an IdM Client \(Full Procedure\)”](#) covers the full procedure for configuring an IdM client manually. One of those steps is creating a host entry, and there are several different ways and options to perform that.

### 5.4.2.1. 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.

**Add Host** [X]

Host Name:  DNS Zone:

IP Address:

Force: ☒

[Add] [Add and Add Another] [Add and Edit] [Cancel]

DNS zones can be created in IdM, which is described in [Section 15.5.1, “Adding and Removing Master 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 host name 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.

- 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.

Hosts » server.example.com

## HOST: server.example.com

server.exempl... is a member of:

server.exempl... is managed by:

Settings Host Groups Netgroups Roles HBAC Rules Sudo Rules Hosts (1)

Refresh Reset Update Collapse

Principal name: host/server.example.com@EXAMPLE.COM

Description:

Locality:

Location:

Platform:

Operating system:

SSH public keys: C9:26:8B:F6:32:D2:0D:08:41:5F:4A:27:7F:93:06:8C (ssh-rsa) Show/Set key Delete  
DD:0E:B4:8B:0E:E0:D9:A8:FF:49:3A:64:3A:75:30:A7 (ssh-dss) Show/Set key Delete  
Add

MAC address: 12-34-56-78-9A-BC undo  
Add undo all

▼ ENROLLMENT

Kerberos Key: ✓ Kerberos Key Present, Host Provisioned

One-Time-Password: ⚠ One-Time-Password Not Present

ACTIONS  
Unprovision  
Set One-Time-Password

#### 5.4.2.2. 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 host name 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 5.6. 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 5.7. 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
```

**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 ensures SSH keys are generated before enrollment, 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
```

```
# Generate SSH keys to ensure that ipa-client-install uploads them
to the IdM server
/usr/sbin/sshd-keygen

# 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
```



## Note

Red Hat recommends not to start the **sshd** service prior to the kickstart enrollment. While starting **sshd** before enrolling the client generates the SSH keys automatically, using the above script is the preferred solution.

4. Run the kickstart script.

## 5.6. Re-enrolling a Host

There can be instances when host information is corrupt or compromised or when a system is being reprovisioned, and the host needs to be re-enrolled to the IdM domain. Re-enrollment updates identifying information for the host:

- » It revokes the original host certificate.
- » It generates a new host certificate.
- » It creates new SSH keys.
- » It retains the unique identifier for the host within the domain, and any historical configuration.

A host can be re-enrolled as long as its domain entry is active. This means that it cannot have been unenrolled (the **ipa-client-install --uninstall** command has never been run), and the host entry is not disabled (**ipa host-disable**).



## Note

The host entry must be active for it to be re-enrolled. Disabling a host revokes all associated certificates, Kerberos keys, and services, which prevents that host from participating in the IdM domain.

The **ipa-client-install** command can re-enroll a host. There are two ways to re-enroll:

- » If the enrollment is being done interactively, then it is possible to force a new enrollment operation with the **--force-join** option. This requires the administrator password for the domain.

```
[root@server ~]# ipa-client-install --force-join --password secret
```

- ✱ If the re-enrollment is automated (such as a kickstart enrollment through a provisioning system) or if it is not feasible to use the administrator password, then it is possible to re-enroll using the existing keytab to authenticate. This is passed in the **--keytab** option. By default, the host keytab location is **/etc/krb5.keytab**.

```
[root@server ~]# ipa-client-install --keytab /etc/krb5.keytab
```

The existing keytab is used to authenticate to initiate the enrollment. As part of the re-enrollment process, a new keytab is generated.

## 5.7. Renaming Machines and Reconfiguring IdM Client Configuration

The host name of a system is critical for the correct operation of Kerberos and SSL. Both of these security mechanisms rely on the host name 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 host name, 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.

```
[root@client ~]# kinit admin
[root@client ~]# ipa hostgroup-find server.example.com
```

3. 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:

```
[root@client ~]# ldapsearch -x -b "cn=accounts,dc=example,dc=com"
"(&(objectclass=ipaservice)(userCertificate=*))" krbPrincipalName
-D "cn=directory manager" -w secret -h ipaserver.example.com -p
389
```

4. 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**.

5. Unenroll the client machine from the IdM domain:

```
[root@client ~]# ipa-client-install --uninstall
```

6. For each identified keytab other than **/etc/krb5.keytab**, remove the old principals:

```
[root@client ~]# ipa-rmkeytab -k /path/to/keytab -r EXAMPLE.COM
```

7. On an IdM server, as an IdM administrator, remove the host entry. This removes all services and revokes all certificates issued for that host:

```
[root@server ~]# kinit admin
[root@server ~]# ipa host-del server.example.com
```

At this point, the host is completely removed from IdM.

8. Rename the machine.
9. Re-enroll the system with IdM:

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

This generates a host principal for with the new host name in **/etc/krb5.keytab**.

10. On an IdM server, add a new keytab for every service:

```
[root@server ~]# ipa service-add serviceName/new-hostname
```

11. To generate certificates for services, use either **certmonger** or the IdM administration tools.
12. Re-add the host to any applicable host groups.

## 5.8. 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 the IdM server configuration 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 5.4.2, “Other Examples of Adding a Host Entry”](#).
2. The second administrator installs the IdM client packages on the machine, as in [Section 5.3, “Configuring a 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.

## 5.9. Removing Clients from the Domain

There are a number of different situations where an IdM client needs to be removed or reconfigured. For example, a client system could be moved from one IdM domain to another or a virtual system could be cloned or moved between systems.

Unenrolling a client (either permanently or as part of reconfiguring the client) is done using the **ipa-client-install** command with the **--uninstall** option. This automatically removes all of the IdM-specific configuration for system services like SSSD and restores its previous configuration.

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

Use the **--updatedns** option, as when installing a client, to update the domain DNS configuration automatically.

**Warning**

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

## 5.10. Manually Unconfiguring Client Machines

There are a number of different situations where an IdM client needs to be reconfigured. If it is not possible to uninstall the client directly, then the IdM configuration can be manually removed from the client system.



## Warning

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

1. On the client, remove the old host name 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:

```
[jsmith@client ~]$ ipa-rmkeytab -k /etc/krb5.keytab -r EXAMPLE.COM
```

To remove specific principals:

```
[jsmith@client ~]$ ipa-rmkeytab -k /etc/krb5.keytab -p  
host/server.example.com@EXAMPLE.COM
```

2. On the client system, disable tracking in **certmonger** for every certificate. Each certificate must be removed from tracking individually.

First, list every certificate being tracked, and extract the database and nickname for each certificate. The number of certificates depends on the configured services for the host.

```
[jsmith@client ~]$ ipa-getcert list
```

Then, disable tracking for each. For example:

```
[jsmith@client ~]$ ipa-getcert stop-tracking -n "Server-Cert" -d  
/etc/httpd/alias
```

3. On the IdM server, 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.

```
[jsmith@server ~]$ kinit admin  
[jsmith@server ~]$ 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 on the client system.

```
[jsmith@client ~]$ ipa-join
```

## Chapter 6. Upgrading Identity Management

Identity Management can be migrated from a Red Hat Enterprise Linux 6.5 system to a Red Hat Enterprise Linux 7 system. This is similar to creating and promoting a replica to replace a server; this process migrates the data and configuration from one instance to another. The older IdM instance can then be decommissioned and replaced by the new IdM instance.



### Warning

If any of the instances in your IdM deployment are using Red Hat Enterprise Linux 6.5 or earlier, upgrade them to Red Hat Enterprise Linux 6.6 before upgrading a Red Hat Enterprise Linux 7.0 IdM server to the 7.1 version or before connecting a Red Hat Enterprise Linux 7.1 IdM replica.

Before upgrading IdM, make sure you have applied the [RHBA-2015:0231-2](#) advisory, which provides the 2.3-6.el6\_6 version of the **bind-dyndb-ldap** packages and is available with the Red Hat Enterprise Linux 6.6 Extended Update Support (EUS). Using a previous **bind-dyndb-ldap** version results in inconsistent behavior in DNS forward zones serving between the Red Hat Enterprise Linux 6.6 DNS servers and Red Hat Enterprise Linux 7 DNS servers.

The following migration rules should be noted when upgrading Identity Management:

**When a replica is created, it must be of an equal or later version than the master it is based on.**

For example, you can install a Red Hat Enterprise Linux 7 replica against a Red Hat Enterprise Linux 6 master, but you cannot install a Red Hat Enterprise Linux 6 replica against a Red Hat Enterprise Linux 7 master.

**Schema changes are replicated between servers.**

Once one master server is updated, all servers and replicas receive 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.



## Important

Due to [CVE-2014-3566](#), the Secure Socket Layer version 3 (SSLv3) protocol needs to be disabled in the `mod_nss` module. You can ensure that by following these steps:

1. Edit the `/etc/httpd/conf.d/nss.conf` file and set the **`NSSProtocol`** parameter to **`TLSv1.0`** (for backward compatibility) and **`TLSv1.1`**.

```
NSSProtocol TLSv1.0,TLSv1.1
```

2. Restart the **`httpd`** service.

```
# systemctl restart httpd.service
```

Note that Identity Management in Red Hat Enterprise Linux 7 automatically performs the above steps when the **`yum update ipa-*`** command is launched to upgrade the main packages.

## 6.1. Migrating the IdM Server to Red Hat Enterprise Linux 7

As is covered in [Section 26.7, “Promoting a Replica to a Master CA Server”](#), only one server within the IdM domain generates certificate revocation lists (CRLs) and has the root signing key to generate certificates. This is the master certificate authority (CA), and it is the master server within the IdM environment.

When migrating an IdM server from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7, the process is very similar to promoting a replica to a master:

1. A new server is created on Red Hat Enterprise Linux 7.
2. All data are migrated over to the new server.
3. All services, such as CRL and certificate creation, DNS management, Kerberos KDC administration, are transitioned over to the new system.



## Important

Migrating an IdM server from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7 involves installing a replica, which requires certain system configuration. For information on these prerequisites, see [Section 4.2, “Prerequisites for Installing a Replica Server”](#).

To migrate an IdM server from Red Hat Enterprise Linux 6 to Red Hat Enterprise Linux 7:

1. Update the Red Hat Enterprise Linux 6 system to the latest Red Hat Enterprise Linux 6 version, and upgrade the *ipa* packages.

```
[root@rhel6 ~]# yum update ipa-*
```



2. Open the required ports. Note that the **firewalld** service needs to be running. You can find information on which ports IdM requires and how to start **firewalld** in [Section 2.4.4, “System Ports”](#).

For example, to open all the IdM required ports in the default zone and make the change both permanent and runtime:

- a. Run the **firewall-cmd** command with the **--permanent** option specified.

```
[root@rhel7 ~]# firewall-cmd --permanent --add-port={80/tcp,443/tcp,389/tcp,636/tcp,88/tcp,464/tcp,88/udp,464/udp,22/tcp}
```

- b. Reload the **firewall-cmd** configuration to ensure that the change takes place immediately.

```
[root@rhel7 ~]# firewall-cmd --reload
```

3. Install the IdM packages on the Red Hat Enterprise Linux 7 system.

```
[root@rhel7 ~]# yum install ipa-server ipa-server-dns
```

4. Copy the Python schema update script from the Red Hat Enterprise Linux 7 system to the Red Hat Enterprise Linux 6 system.

```
[root@rhel7 ~]# scp /usr/share/ipa/copy-schema-to-ca.py  
rhel6:/root/
```

Updating the script in this way is necessary due to schema changes between IdM version 3.1 and later IdM versions.

5. Run the schema update script on the Red Hat Enterprise Linux 6 system.

```
[root@rhel6 ~]# python copy-schema-to-ca.py  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60kerberos.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60samba.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60ipaconfig.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60basev2.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60basev3.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/60ipadns.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/61kerberos-ipav3.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/65ipasudo.ldif  
ipa          : INFO      Installed /etc/dirsrv/slapd-PKI-  
IPA//schema/05rfc2247.ldif  
ipa          : INFO      Restarting CA DS  
ipa          : INFO      Schema updated successfully
```

6. On the Red Hat Enterprise Linux 6 system, create the replica file for the Red Hat Enterprise Linux 7 system; in this example, the new replica server is **rhel7.example.com** with the **192.0.2.1** IP address.

```
[root@rhel6 ~]# ipa-replica-prepare rhel7.example.com --ip-address 192.0.2.1
```

```
Directory Manager (existing master) password:
Preparing replica for rhel7.example.com from rhel6.example.com
Creating SSL certificate for the Directory Server
Creating SSL certificate for the dogtag Directory Server
Saving dogtag Directory Server port
Creating SSL certificate for the Web Server
Exporting RA certificate
Copying additional files
Finalizing configuration
Packaging replica information into /var/lib/ipa/replica-info-
rhel7.example.com.gpg
Adding DNS records for rhel7.example.com
Using reverse zone 2.0.192.in-addr.arpa.
The ipa-replica-prepare command was successful
```

7. Install the replica, using the new replica file, on the Red Hat Enterprise Linux 7 system. Use the **--setup-ca** option to set up a Dogtag Certificate System instance and the **--setup-dns** option to configure the DNS server. The replica server's IP address in this example is **192.0.2.1**.

```
[root@rhel7 ~]# ipa-replica-install --setup-ca --ip-
address=192.0.2.1 -p secret -w secret -N --setup-dns --
forwarder=192.0.2.20 -U /var/lib/ipa/replica-info-
rhel7.example.com.gpg
```

```
Run connection check to master
Check connection from replica to remote master
'rhel6.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
  PKI-CA: Directory Service port (7389): OK
```

```
...
```

8. Verify the configuration.
  - a. Verify that the IdM services are running:

```
[root@rhel7 ~]# ipactl status
Directory Service: RUNNING
krb5kdc Service: RUNNING
kadmin Service: RUNNING
named Service: RUNNING
ipa_memcached Service: RUNNING
```

```

httpd Service: RUNNING
pki-tomcatd Service: RUNNING
ipa-otpd Service: RUNNING
ipa: INFO: The ipactl command was successful

```

- b. Verify that both IdM CAs are configured as master servers.

```

[root@rhel7 ~]# kinit admin
[root@rhel7 ~]# ipa-replica-manage list
rhel6.example.com: master
rhel7.example.com: master
[root@rhel7 ~]# ipa-replica-manage list -v rhel7.example.com
rhel6.example.com: replica
    last init status: None
    last init ended: None
    last update status: 0 Replica acquired successfully:
Incremental update started
    last update ended: None

```

9. *On the Red Hat Enterprise Linux 6 system.* Edit the Red Hat Enterprise Linux 6 IdM server so that it no longer renews the CA subsystem certificates or issues CRLs.

- a. Identify which server instance is the master CA server. Both CRL generation *and* renewal operations are handled by the same CA server. So, the master CA can be identified by having the **renew\_ca\_cert** certificate being tracked by **certmonger**.

```

[root@rhel6 ~]# getcert list -d /var/lib/pki-ca/alias -n
"subsystemCert cert-pki-ca" | grep post-save
post-save command: /usr/lib64/ipa/certmonger/renew_ca_cert
"subsystemCert cert-pki-ca"

```

- b. *On the original master CA*, disable tracking for all of the original CA certificates.

```

[root@rhel6 ~]# getcert stop-tracking -d /var/lib/pki-
ca/alias -n "auditSigningCert cert-pki-ca"
Request "20151127184547" removed.
[root@rhel6 ~]# getcert stop-tracking -d /var/lib/pki-
ca/alias -n "ocspSigningCert cert-pki-ca"
Request "20151127184548" removed.
[root@rhel6 ~]# getcert stop-tracking -d /var/lib/pki-
ca/alias -n "subsystemCert cert-pki-ca"
Request "20151127184549" removed.
[root@rhel6 ~]# getcert stop-tracking -d /etc/httpd/alias -n
ipaCert
Request "20151127184550" removed.

```

- c. Reconfigure the original master CA to retrieve renewed certificates from a new master CA.
- a. Copy the renewal helper into the **certmonger** service directory, and set the appropriate permissions.

```
[root@rhel6 ~]# cp /usr/share/ipa/ca_renewal
/var/lib/certmonger/cas/ca_renewal
[root@rhel6 ~]# chmod 0600
/var/lib/certmonger/cas/ca_renewal
```

- b. Update the SELinux configuration.

```
[root@rhel6 ~]# /sbin/restorecon
/var/lib/certmonger/cas/ca_renewal
```

- c. Restart **certmonger**.

```
[root@rhel6 ~]# service certmonger restart
```

- d. Check that the CA is listed to *retrieve* certificates. This is printed in the CA configuration.

```
[root@rhel6 ~]# getcert list-cas
...
CA 'dogtag-ipa-retrieve-agent-submit':
    is-default: no
    ca-type: EXTERNAL
    helper-location: /usr/libexec/certmonger/dogtag-ipa-
retrieve-agent-submit
```

- e. Get the CA certificate database PIN.

```
[root@rhel6 ~]# grep internal= /var/lib/pki-
ca/conf/password.conf
```

- f. Configure **certmonger** to track the certificates for external renewal. This requires the database PIN.

```
[root@rhel6 ~]# getcert start-tracking -c dogtag-ipa-
retrieve-agent-submit -d /var/lib/pki-ca/alias -n
"auditSigningCert cert-pki-ca" -B
/usr/lib64/ipa/certmonger/stop_pkicad -C
'/usr/lib64/ipa/certmonger/restart_pkicad
"auditSigningCert cert-pki-ca"' -T "auditSigningCert
cert-pki-ca" -P database_pin
New tracking request "20151127184743" added.
[root@rhel6 ~]# getcert start-tracking -c dogtag-ipa-
retrieve-agent-submit -d /var/lib/pki-ca/alias -n
"ocspSigningCert cert-pki-ca" -B
/usr/lib64/ipa/certmonger/stop_pkicad -C
'/usr/lib64/ipa/certmonger/restart_pkicad
"ocspSigningCert cert-pki-ca"' -T "ocspSigningCert
cert-pki-ca" -P database_pin
New tracking request "20151127184744" added.
[root@rhel6 ~]# getcert start-tracking -c dogtag-ipa-
retrieve-agent-submit -d /var/lib/pki-ca/alias -n
"subsystemCert cert-pki-ca" -B
/usr/lib64/ipa/certmonger/stop_pkicad -C
'/usr/lib64/ipa/certmonger/restart_pkicad
```

```
"subsystemCert cert-pki-ca"' -T "subsystemCert cert-
pki-ca" -P database_pin
New tracking request "20151127184745" added.
[root@rhel6 ~]# getcert start-tracking -c dogtag-ipa-
retrieve-agent-submit -d /etc/httpd/alias -n ipaCert -C
/usr/lib64/ipa/certmonger/restart_httpd -T ipaCert -p
/etc/httpd/alias/pwdfilere.txt
New tracking request "20151127184746" added.
```

d. Stop CRL generation on the original master CA.

a. Stop CA service.

```
[root@rhel6 ~]# service pki-cad stop
```

b. Open the CA configuration file.

```
[root@rhel6 ~]# vim /var/lib/pki-ca/conf/CS.cfg
```

c. Change the values of the **ca.crl.MasterCRL.enableCRLCache** and **ca.crl.MasterCRL.enableCRLUpdates** parameters to **false** to disable CRL generation.

```
ca.crl.MasterCRL.enableCRLCache=false
ca.crl.MasterCRL.enableCRLUpdates=false
```

d. Start the CA service.

```
[root@rhel6 ~]# service pki-cad start
```

e. Configure Apache to redirect CRL requests to the new master.

a. Open the CA proxy configuration.

```
[root@rhel6 ~]# vim /etc/httpd/conf.d/ipa-pki-
proxy.conf
```

b. Uncomment the **RewriteRule** on the last line and replace the example server URL with the new Red Hat Enterprise Linux 7 server URL.

```
RewriteRule ^/ipa/crl/MasterCRL.bin
https://rhel7.example.com/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL [L,R=301,NC]
```

c. Restart Apache.

```
[root@rhel6 ~]# systemctl restart httpd.service
```

10. On the Red Hat Enterprise Linux 7 system. Configure the new Red Hat Enterprise Linux 7 IdM instance as the master:

a. Configure CA renewal using the **ipa-csreplica-manage** utility.

```
[root@rhel7 ~]# ipa-csreplica-manage set-renewal-master
```

- b. Configure the new master CA to generate CRLs.

- a. Stop CA service.

```
[root@rhel7 ~]# systemctl stop pki-tomcatd@pki-  
tomcat.service
```

- b. Open the CA configuration file.

```
[root@rhel7 ~]# vim /etc/pki/pki-tomcat/ca/CS.cfg
```

- c. Change the values of the **ca.crl.MasterCRL.enableCRLCache** and **ca.crl.MasterCRL.enableCRLUpdates** parameters to **true** to enable CRL generation.

```
ca.crl.MasterCRL.enableCRLCache=true  
ca.crl.MasterCRL.enableCRLUpdates=true
```

- d. Start CA service.

```
[root@rhel7 ~]# systemctl start pki-tomcatd@pki-  
tomcat.service
```

- c. Configure Apache to disable redirect CRL requests. As a clone, all CRL requests were routed to the original master. As the new master, this instance will respond to CRL requests.

- a. Open the CA proxy configuration.

```
[root@rhel7 ~]# vim /etc/httpd/conf.d/ipa-pki-  
proxy.conf
```

- b. Comment out the **RewriteRule** argument on the last line.

```
#RewriteRule ^/ipa/crl/MasterCRL.bin  
https://server.example.com/ca/ee/ca/getCRL?  
op=getCRL&crlIssuingPoint=MasterCRL [L,R=301,NC]
```

- c. Restart Apache.

```
[root@rhel7 ~]# systemctl restart httpd.service
```

11. Stop all services on the Red Hat Enterprise Linux 6 system; this forces domain discovery to the Red Hat Enterprise Linux 7 server.

```
[root@rhel6 ~]# ipactl stop  
Stopping CA Service  
Stopping pki-ca: [ OK  
]  
Stopping HTTP Service  
Stopping httpd: [ OK
```

```

]
Stopping MEMCACHE Service
Stopping ipa_memcached: [ OK
]
Stopping DNS Service
Stopping named: . [ OK
]
Stopping KPASSWD Service
Stopping Kerberos 5 Admin Server: [ OK
]
Stopping KDC Service
Stopping Kerberos 5 KDC: [ OK
]
Stopping Directory Service
Shutting down dirsrv:
    EXAMPLE-COM... [ OK
]
    PKI-IPA... [ OK
]

```

12. For each server in the environment, create a replica file from the Red Hat Enterprise Linux 7 master server, and install it on the new Red Hat Enterprise Linux 7 replica system. Creating replicas is covered in [Chapter 4, Setting up IdM Replicas](#).
13. Decommission the Red Hat Enterprise Linux 6 host.
  - a. Remove the Red Hat Enterprise Linux 6 server from the IdM server topology by running the **ipa-replica-manage del** command *on the Red Hat Enterprise Linux 7 system*.

```

[root@rhel7 ~]# ipa-replica-manage del rhel6.example.com
Connection to 'rhel6.example.com' failed:
Forcing removal of rhel6.example.com
Skipping calculation to determine if one or more masters
would be orphaned.
Deleting replication agreements between rhel6.example.com and
rhel7.example.com
Failed to get list of agreements from 'rhel6.example.com':
Forcing removal on 'rhel7.example.com'
Any DNA range on 'rhel6.example.com' will be lost
Deleted replication agreement from 'rhel7.example.com' to
'rhel6.example.com'
Background task created to clean replication data. This may
take a while.
This may be safely interrupted with Ctrl+C

```

- b. Remove the local IdM configuration.

```

[root@rhel6 ~]# ipa-server-install --uninstall

```

## Chapter 7. Uninstalling IdM Servers and Replicas

To uninstall an IdM server, add the **--uninstall** option to the **ipa-server-install** command:

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

The procedure for uninstalling an IdM replica is described in [Section 29.2, “Removing a Replica”](#).



## 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 servers 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. <sup>[1]</sup> 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 **systemctl** command sets what services to start automatically when the system restarts.

```
[root@server ~]# systemctl enable ipa.service
```

The **ipactl** command can be used to start the domain services in the proper order, without having to configure each one individually in the **systemctl** run order.

### 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
```



#### Note

To get a list of all available topics:

```
ipa help topics
```

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 10.10.3.1.2, “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 is 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** command uses 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 **--dynamic-update** 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.



### Important

The value of the attribute being added is not validated by the modify command or the **--setattr** or **--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. Setting a List of Values

In LDAP, multi-valued attributes have a single attribute-value assertion (*attribute: value*) that can be used multiple times in an entry. For example:

```
mail: admin@example.com
mail: jsmith@example.com
```

There are some attributes, however, where the value itself can contain a list. For example, in IdM, the attributes that are searched when searching for users or groups is defined in a list, not in multiple AVAs:

```
ipaUserSearchFields: uid,givenname,sn,telephonenumber,ou,title
```

This applies to search fields, permissions and other access control settings, and some types of group lists, like for sudo command groups, host-based access control rules, and service groups.

Attempting to add a single item to the list when modifying the entry overwrites all previous settings, because there is only a single AVA. It is possible to create the list value in either of two ways:

- ✧ Use the same command-line argument multiple times *within the same command invocation*. For example:

```
--permissions=read --permissions=write --permissions=delete
```

- ✧ Enclose the list in curly braces, which allows the shell to do the expansion. For example:

```
--permissions={read,write,delete}
```



### Important

When adding or updating an attribute with a list as a value, include every item in the list with the update. The list is updated every time, and any previous value is overwritten with the new update.

## 8.2.5. 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.

## 8.2.6. 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 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 than 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 is 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.

Server sessions last for 20 minutes and are refreshed on user actions. To display updated details for a new user, log out and log back in or wait for the session to expire.

### 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: KEYRING:persistent:0:0
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
```

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 **kinit** have some limitation, one of them being that the current ticket is overwritten with any new invocation of **kinit**. 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:

- ✧ Mozilla Firefox 17 and higher
- ✧ Google Chrome web browser

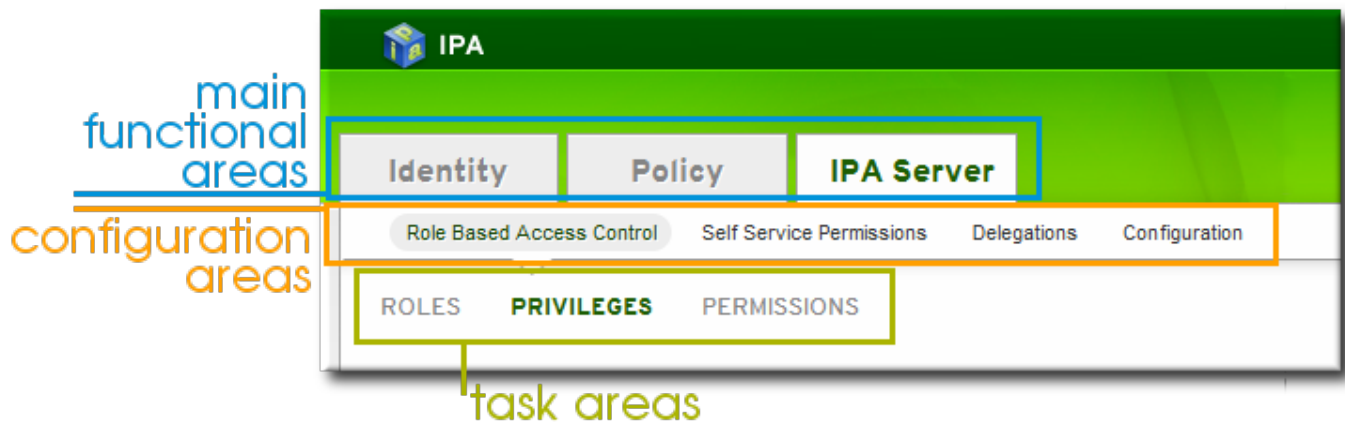
### 8.4.2. About the Web UI

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	<ul style="list-style-type: none"> <li>✧ User entries</li> <li>✧ User groups entries</li> <li>✧ Host/client entries</li> <li>✧ Host group entries</li> <li>✧ Netgroups entries</li> <li>✧ Domain services entries</li> <li>✧ DNS (if configured)</li> </ul>
Policy	<ul style="list-style-type: none"> <li>✧ Host-based access control</li> <li>✧ Sudo rules</li> <li>✧ Automount</li> <li>✧ User password policies</li> <li>✧ Kerberos ticket policy</li> </ul>
IdM Server (access controls within Identity Management)	<ul style="list-style-type: none"> <li>✧ Role-based access control (permissions based on group membership)</li> <li>✧ Self permissions</li> <li>✧ Delegations (user access control over other users)</li> </ul>

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.

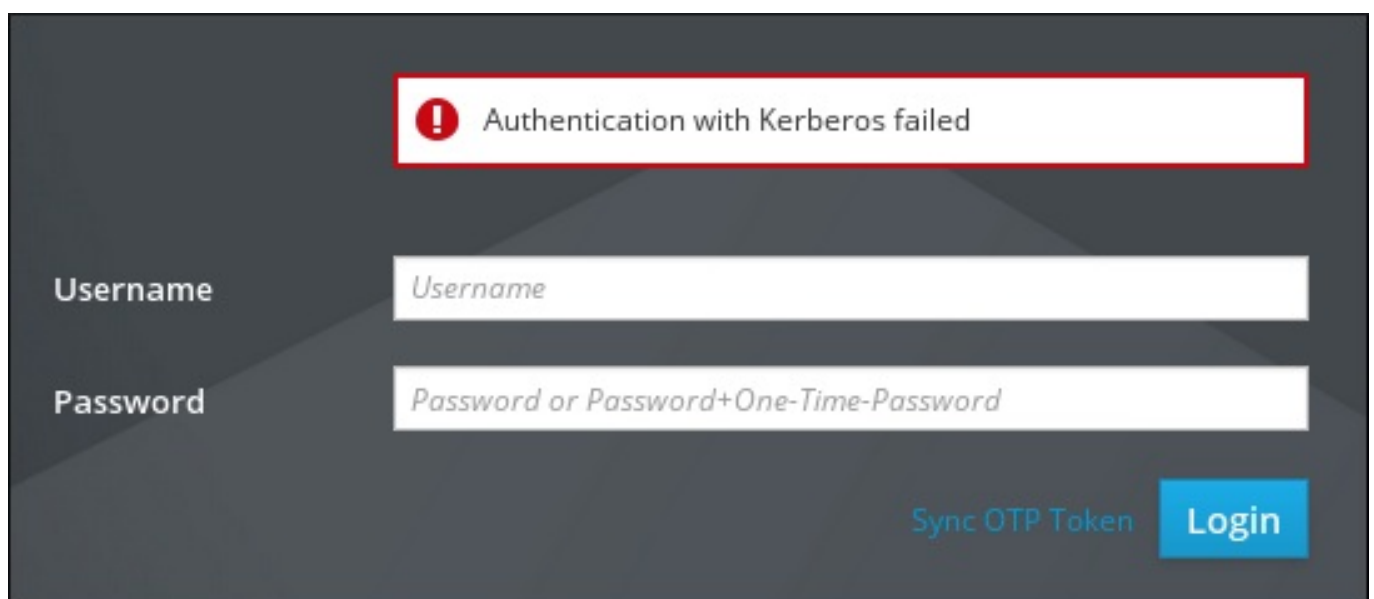
To open the web UI:

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. If Firefox is not configured to support Kerberos authentication, an error message appears after clicking **Login** on the main IdM web UI page.



## Figure 8.2. Kerberos Authentication Error

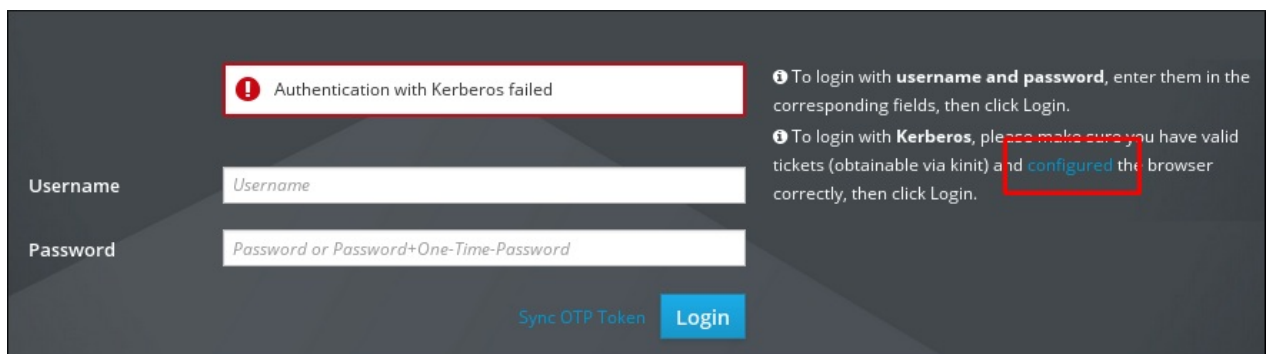
You can configure Firefox for Kerberos authentication:

- ✧ automatically from the IdM web UI
- ✧ automatically from the command line during client installation
- ✧ manually in the Firefox configuration settings

### 8.4.4.1. Automatic Firefox Configuration in the Web UI

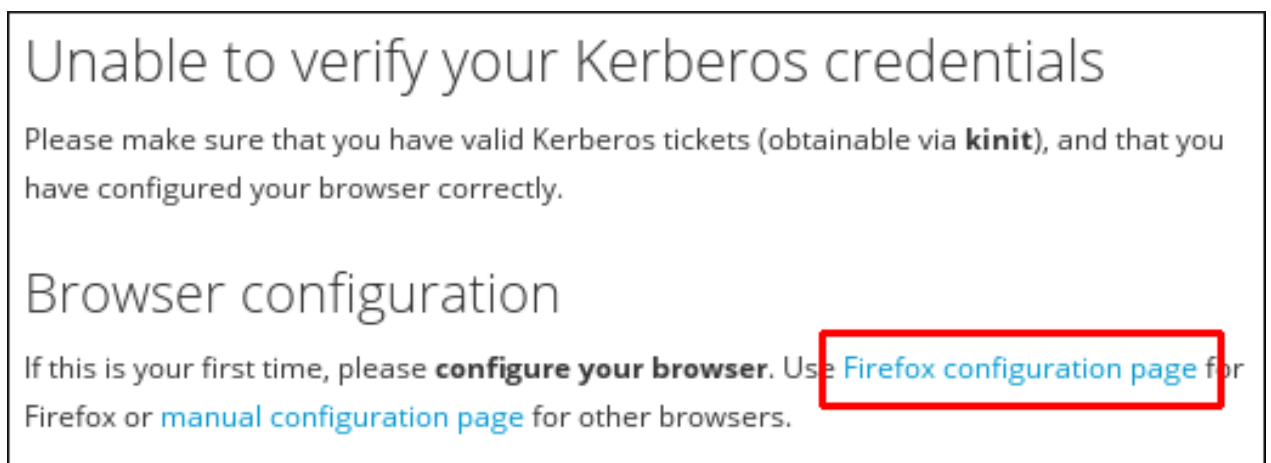
To automatically configure Firefox from the IdM web UI:

1. Click the link on the main IdM web UI page inviting you to configure the browser.



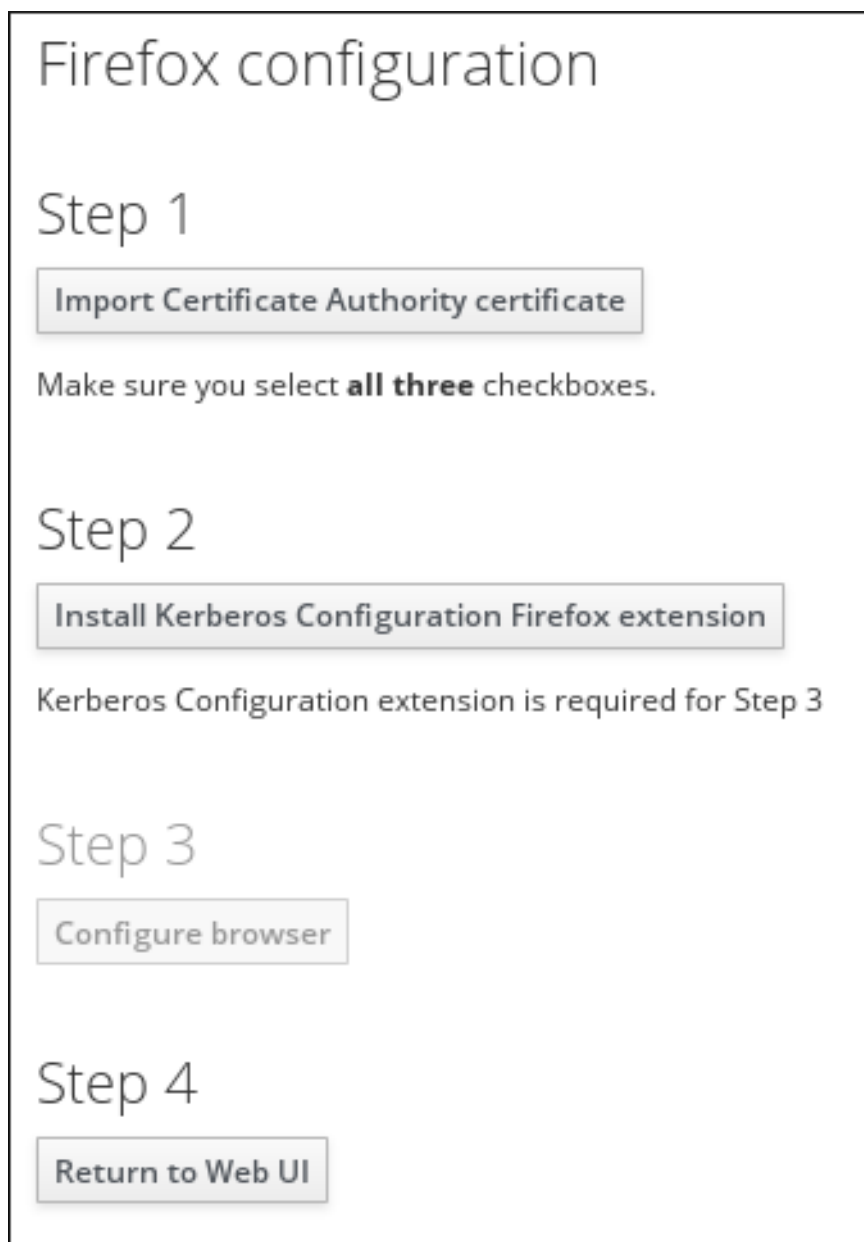
**Figure 8.3. Link to Configure Firefox in the Web UI**

2. Choose the link for Firefox configuration.



**Figure 8.4. Link to Firefox Configuration Page**

3. Follow the steps to configure Firefox.



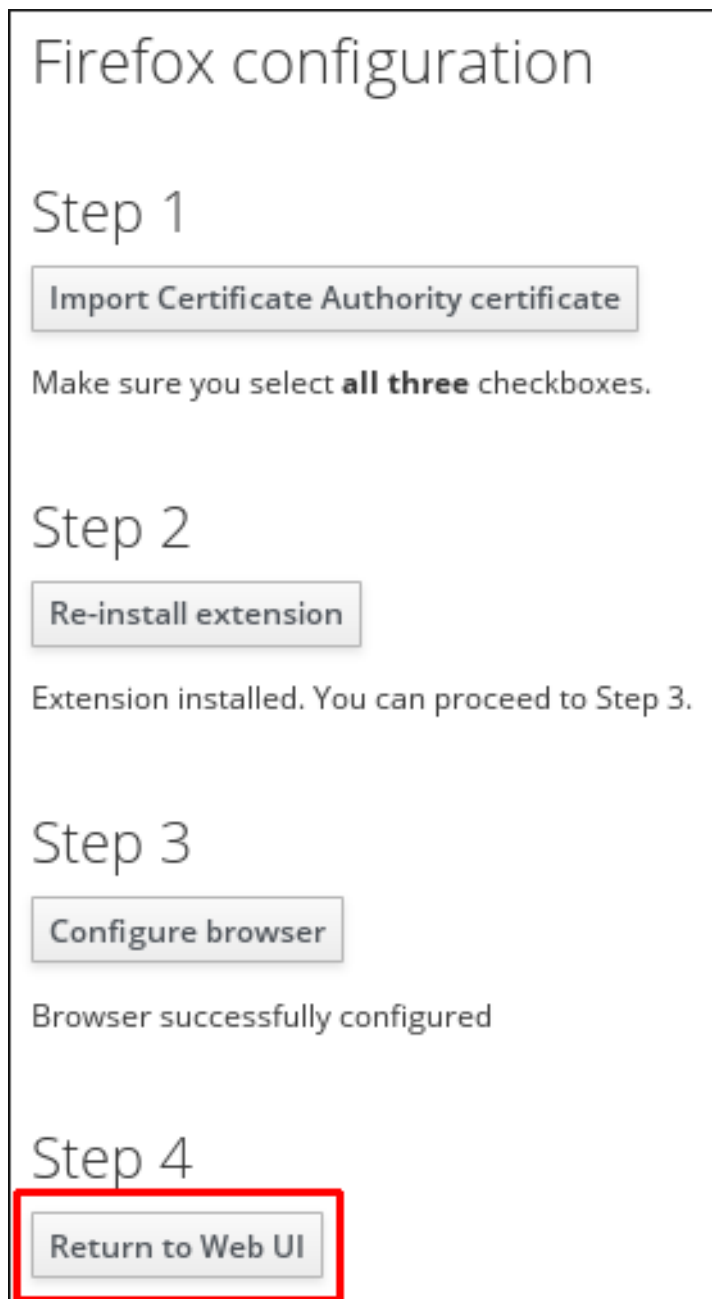
**Figure 8.5. Steps to Configure Firefox**

- a. Click **Import Certificate Authority certificate** and select all three checkboxes in the form that shows up.



**Figure 8.6. Importing the CA certificate**

- b. Click **Install Kerberos Configuration Firefox extension** and allow Firefox to install the extension.
- c. Click **Configure browser**, which automatically configures the required **negotiate** settings in Firefox configuration.
- d. Firefox is now correctly configured. Click **Return to Web UI**.



**Figure 8.7. Importing the CA certificate**

#### **8.4.4.2. Automatic Firefox Configuration from the Command Line**

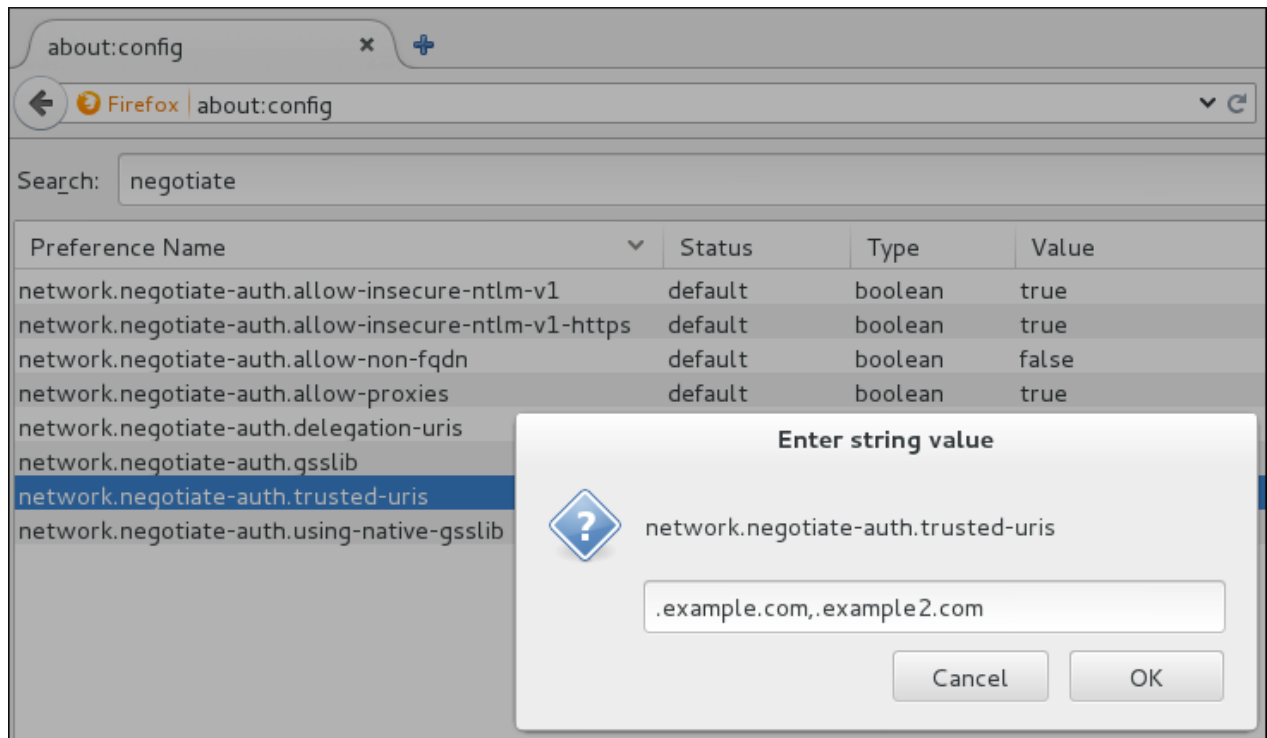
During IdM client installation, you can automatically configure Firefox from the command line. To do this, add the `--configure-firefox` option to the `ipa-client-install` command. The `--configure-firefox` option creates a global configuration file with default Firefox settings that enable Kerberos for SSO.

#### **8.4.4.3. Manual Firefox Configuration**

To configure Firefox for Kerberos authentication manually:

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

- On Red Hat Enterprise Linux double-click the **network.negotiate-auth.trusted-uris** entry and enter the name of the domain against which to authenticate, including the preceding period (.). If you want to add multiple domains, enter them in a comma-separated list.



**Figure 8.8. Manual Firefox Configuration for Kerberos**

On Windows, double-click and fill out the following entries:

- ✧ Enter the domain name or names against which to authenticate, including the preceding period (.), in the **network.negotiate-auth.trusted-uris** entry. For example:

.example.com

- ✧ Enter the **GSSLib** library path in the **network.negotiate-auth.gsslib** entry:

C:\Program Files\MIT\Kerberos\bin\gssapi32.dll

Note that on a 64-bit system, the **GSSLib** location is **C:\Program Files(x86)\MIT\Kerberos\bin\gssapi32.dll**.

- ✧ Disable the built-in Microsoft Kerberos for authentication by setting the **network.auth.use-sspi** entry to **false**.
- 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** and select all three checkboxes in the form that shows up to properly install the certificate.



**Figure 8.9. Installing the IdM Server CA Certificate**

If Kerberos authentication is not working after performing these steps, refer to [the troubleshooting section in the System-Level Authentication Guide](#).

### 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.

1. 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.

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

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

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

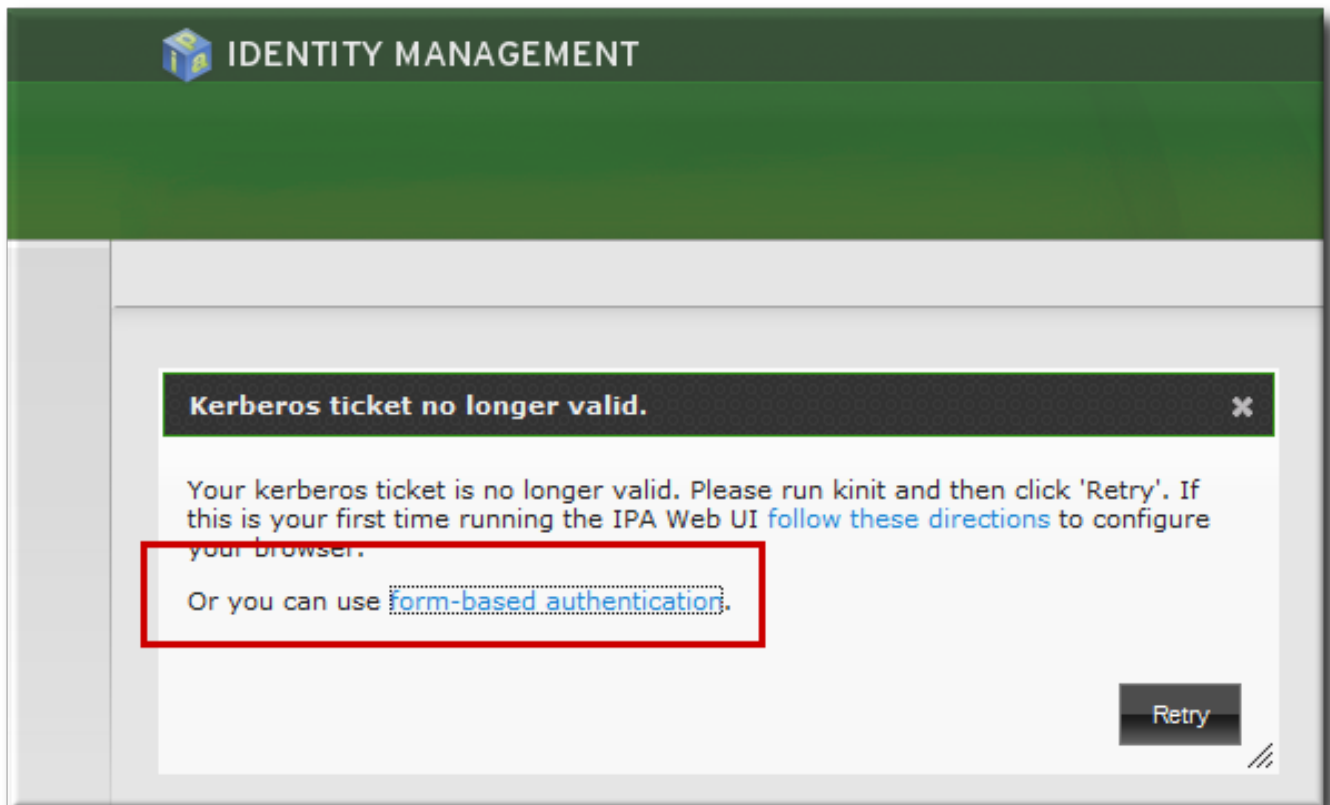
### 8.4.6. Logging in with Simple Username/Password Credentials



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.

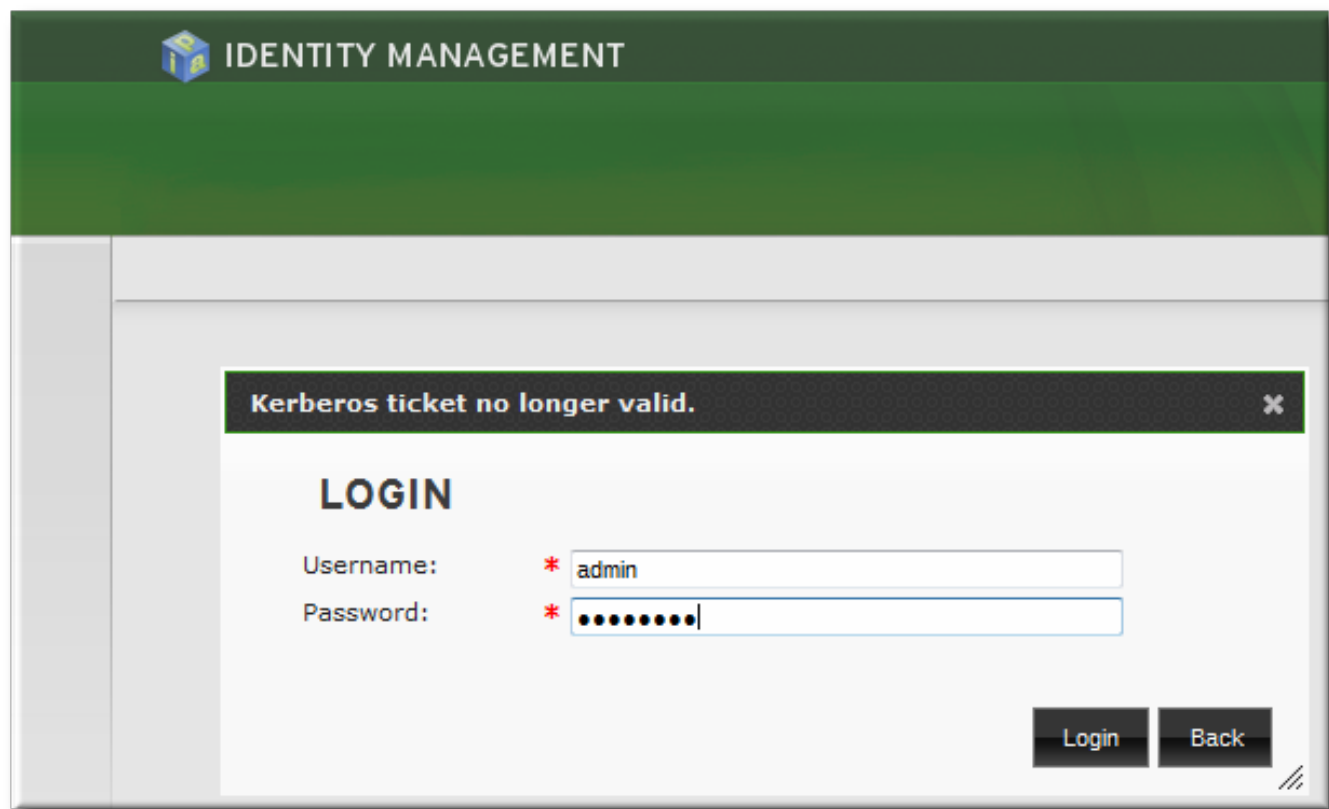
When the IdM server cannot find a valid Kerberos ticket for the user attempting to log into the web UI, it splashes an error message. Since the preferred method of connecting to IdM domain services (including the UI) is using Kerberos authentication, the error first says to renew the Kerberos credentials 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 8.10. IdM Form-Based Login Option**

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



**Figure 8.11. 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. See the `ssh(1)` manual page for details on how to set this up using the **-D** option of **ssh**.

---

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

## Chapter 9. Backing Up and Restoring Identity Management

Red Hat Enterprise Linux Identity Management provides a solution to manually back up and restore the IdM system, for example when a server stops performing correctly or data loss occurs. During backup, the system creates a directory containing information on your IdM setup and stores it. During restore, you can use this backup directory to bring your original IdM setup back.



### Important

Use the backup and restore procedures described in this chapter only if you cannot rebuild the lost part of the IdM server group from the remaining servers in the deployment, by reinstalling the lost replicas as replicas of the remaining ones.

The ["Backup and Restore in IdM/IPA" Knowledgebase solution](#) describes how to avoid losses by maintaining several server replicas. Rebuilding from an existing replica with the same data is preferable, because the backed-up version usually contains older, thus potentially outdated, information.

The potential threat scenarios that backup and restore can prevent include:

- Catastrophic hardware failure on a machine occurs and the machine becomes incapable of further functioning. In this situation, you can reinstall the operating system from scratch, configure the machine with the same fully-qualified domain name (FQDN) and host name, install the IdM packages as well as all other optional packages relating to IdM that were present on the original system, and restore the fully-backed-up IdM server.
- An upgrade on an isolated machine fails. The operating system remains functional, but the IdM data is corrupted, which is why you want to restore the IdM system to a known good state.



### Important

In cases of hardware or upgrade failure, such as the two mentioned above, restore from backup only if all replicas or a replica with a special role, such as the only certificate authority (CA), were lost. If a replica with the same data still exists, it is recommended to delete the lost replica and then rebuild it from the remaining one.

- Undesirable changes were made to the LDAP content, for example entries were deleted, and you want to revert them. Restoring backed-up LDAP data returns the LDAP entries to the previous state without affecting the IdM system itself.

The restored server becomes the only source of information for IdM; other master servers are re-initialized from the restored server. Any data created after the last backup was made are lost. Therefore you should not use the backup and restore solution for normal system maintenance. If possible, always rebuild the lost server by reinstalling it as a replica.

The backup and restore features can be managed only from the command line and are not available in the IdM web UI.

## 9.1. Full-Server Backup and Data-Only Backup

IdM offers two backup options:

### Full-IdM server backup

Full-server backup creates a backup copy of all the IdM server files as well as LDAP data, which makes it a standalone backup. IdM affects hundreds of files; the files that the backup process copies is a mix of whole directories and specific files, such as configuration files or log files, and relate directly to IdM or to various services that IdM depends on. Because the full-server backup is a raw file backup, it is performed offline. The script that performs the full-server backup stops all IdM services to ensure a safe course of the backup process.

For the full list of files and directories that the full-server backup copies, see [Section 9.1.3, “List of Directories and Files Copied During Backup”](#).

### Data-only Backup

The data-only backup only creates a backup copy of LDAP data and the changelog. The process backs up the **IPA-REALM** instance and can also back up multiple back ends or only a single back end; the back ends include the **IPA** back end and the **CA Dogtag** back end. This type of backup also backs up a record of the LDAP content stored in LDIF (LDAP Data Interchange Format). The data-only backup can be performed both online and offline.

By default, IdM stores the created backups in the `/var/lib/ipa/backup/` directory. The naming conventions for the subdirectories containing the backups are:

- ✱ **ipa-full-YEAR-MM-DD-HH-MM-SS** in the GMT time zone for the full-server backup
- ✱ **ipa-data-YEAR-MM-DD-HH-MM-SS** in the GMT time zone for the data-only backup

### 9.1.1. Creating a Backup

Both full-server and data-only backups are created using the **ipa-backup** utility which must always be run as root.

To create a full-server backup, run **ipa-backup**.



#### Important

Performing a full-server backup stops all IdM services because the process must run offline. The IdM services will start again after the backup is finished.

To create a data-only backup, run the **ipa-backup --data** command.

You can add several additional options to **ipa-backup**:

- ✱ **--online** performs an online backup; this option is only available with data-only backups

- » **--logs** includes the IdM service log files in the backup

For further information on using **ipa-backup**, see the `ipa-backup(1)` man page.

### 9.1.2. Encrypting Backup

You can encrypt the IdM backup using the GNU Privacy Guard (GPG) encryption.

To create a GPG key:

1. Create a **keygen** file containing the key details, for example, by running **cat >keygen <<EOF** and providing the required encryption details to the file from the command line:

```
[root@server ~]# cat >keygen <<EOF
> %echo Generating a standard key
> Key-Type: RSA
> Key-Length:2048
> Name-Real: IPA Backup
> Name-Comment: IPA Backup
> Name-Email: root@example.com
> Expire-Dat: 0
> %pubring /root/backup.pub
> %secring /root/backup.sec
> %commit
> %echo done
> EOF
[root@server ~]#
```

2. Generate a new key pair called **backup** and feed the contents of **keygen** to the command. The following example generates a key pair with the path names **/root/backup.sec** and **/root/backup.pub**:

```
[root@server ~]# gpg --batch --gen-key keygen
[root@server ~]# gpg --no-default-keyring --secret-keyring
/root/backup.sec \
    --keyring /root/backup.pub --list-secret-keys
```

To create a GPG-encrypted backup, pass the generated **backup** key to **ipa-backup** by supplying the following options:

- » **--gpg**, which instructs **ipa-backup** to perform the encrypted backup
- » **--gpg-keyring=GPG\_KEYRING**, which provides the full path to the GPG keyring without the file extension.

For example:

```
[root@server ~]# ipa-backup --gpg --gpg-keyring=/root/backup
```

**Note**

You might experience problems if your system uses the **gpg2** utility to generate GPG keys because **gpg2** requires an external program to function. To generate the key purely from console in this situation, add the **pinentry-program /usr/bin/pinentry-curses** line to the **.gnupg/gpg-agent.conf** file before generating a key.

**9.1.3. List of Directories and Files Copied During Backup**

Directories:

```
/usr/share/ipa/html
/root/.pki
/etc/pki-ca
/etc/pki/pki-tomcat
/etc/sysconfig/pki
/etc/httpd/alias
/var/lib/pki
/var/lib/pki-ca
/var/lib/ipa/sysrestore
/var/lib/ipa-client/sysrestore
/var/lib/ipa/dnssec
/var/lib/sss/pubconf/krb5.include.d/
/var/lib/authconfig/last
/var/lib/certmonger
/var/lib/ipa
/var/run/dirsrv
/var/lock/dirsrv
```

Files:

```
/etc/named.conf
/etc/named.keytab
/etc/resolv.conf
/etc/sysconfig/pki-ca
/etc/sysconfig/pki-tomcat
/etc/sysconfig/dirsrv
/etc/sysconfig/ntpd
/etc/sysconfig/krb5kdc
/etc/sysconfig/pki/ca/pki-ca
/etc/sysconfig/ipa-dnskeysyncd
/etc/sysconfig/ipa-ods-exporter
/etc/sysconfig/named
/etc/sysconfig/ods
/etc/sysconfig/authconfig
/etc/ipa/nssdb/pwdfilere.txt
/etc/pki/ca-trust/source/ipa.p11-kit
/etc/pki/ca-trust/source/anchors/ipa-ca.crt
/etc/nsswitch.conf
/etc/krb5.keytab
/etc/sss/sss.conf
/etc/openldap/ldap.conf
```

```

/etc/security/limits.conf
/etc/httpd/conf/password.conf
/etc/httpd/conf/ipa.keytab
/etc/httpd/conf.d/ipa-pki-proxy.conf
/etc/httpd/conf.d/ipa-rewrite.conf
/etc/httpd/conf.d/nss.conf
/etc/httpd/conf.d/ipa.conf
/etc/ssh/sshd_config
/etc/ssh/ssh_config
/etc/krb5.conf
/etc/ipa/ca.crt
/etc/ipa/default.conf
/etc/dirsrv/ds.keytab
/etc/ntp.conf
/etc/samba/smb.conf
/etc/samba/samba.keytab
/root/ca-agent.p12
/root/cacert.p12
/var/kerberos/krb5kdc/kdc.conf
/etc/systemd/system/multi-user.target.wants/ipa.service
/etc/systemd/system/multi-user.target.wants/sss.service
/etc/systemd/system/multi-user.target.wants/certmonger.service
/etc/systemd/system/pki-tomcatd.target.wants/pki-tomcatd@pki-
tomcat.service
/var/run/ipa/services.list
/etc/openssl/conf.xml
/etc/openssl/kasp.xml
/etc/ipa/dnssec/softsm2.conf
/etc/ipa/dnssec/softsm_pin_so
/etc/ipa/dnssec/ipa-ods-exporter.keytab
/etc/ipa/dnssec/ipa-dnskeysyncd.keytab
/etc/pki/nssdb/cert8.db
/etc/pki/nssdb/key3.db
/etc/pki/nssdb/secmod.db
/etc/ipa/nssdb/cert8.db
/etc/ipa/nssdb/key3.db
/etc/ipa/nssdb/secmod.db

```

Log files and directories:

```

/var/log/pki-ca
/var/log/pki/
/var/log/dirsrv/slapd-PKI-IPA
/var/log/httpd
/var/log/ipaserver-install.log
/var/log/kadmind.log
/var/log/pki-ca-install.log
/var/log/messages
/var/log/ipaclient-install.log
/var/log/secure
/var/log/ipaserver-uninstall.log
/var/log/pki-ca-uninstall.log
/var/log/ipaclient-uninstall.log
/var/named/data/named.run

```

## 9.2. Restoring a Backup

If you have a directory with a backup created using **ipa-backup**, you can restore your IdM server or the LDAP content to the state in which they were when the backup was performed. You cannot restore a backup on a host different from the host on which the backup was originally created.



### Note

Uninstalling an IdM server does not automatically remove the backup of this server.

### 9.2.1. Restoring from the Full-Server or Data-Only Backup



### Important

It is recommended that you uninstall a server before performing a full-server restore on it.

Both full-server and data-only backups are restored using the **ipa-restore** utility which must always be run as root. Pass the backup to the command:

- ✧ Pass only the name of the directory with the backup if it is located in the default **/var/lib/ipa/backup/** directory.
- ✧ Pass the full path to the backup if the directory containing the backup is not located in the default directory. For example:

```
[root@server ~]# ipa-restore /path/to/backup
```

The **ipa-restore** utility automatically detects what type of backup the backup directory contains and by default performs the same type of restore.

You can add the following options to **ipa-restore**:

- ✧ **--data** performs a data-only restore from a full-server backup, that is, restores only the LDAP data component from a backup directory containing the full-server backup
- ✧ **--online** restores the LDAP data in a data-only restore online
- ✧ **--instance** specifies which 389 DS instance is restored. IdM in Red Hat Enterprise Linux 7 only uses the **IPA-REALM** instance, but it might be possible, for example, to create a backup on a system with separate instances; in such cases, **--instance** allows you to restore only **IPA-REALM**. For example:

```
[root@server ~]# ipa-restore --instance=IPA-REALM /path/to/backup
```

You can use this option only when performing a data-only restore.



- ✱ **--backend** specifies which back end is restored; without this option, **ipa-restore** restores all back ends it discovers. The arguments defining the possible back ends are **userRoot**, which restores the IPA data back end, and **ipaca**, which restores the CA back end.

You can use this option only when performing a data-only restore.

- ✱ **--no-logs** restores the backup without restoring the log files



### Note

It is recommended that you reboot your system after restoring from backup.

For further information on using **ipa-restore**, see the `ipa-restore(1)` man page.

## 9.2.2. Restoring with Multiple Master Servers

Restoring from backup sets the restored server as the new data master, and you will be required to reinitialize all other masters after the restore. To reinitialize the other masters, run the **ipa-replica-manage** command and, on masters that have a CA installed, the **ipa-csrelica-manage** command. For example:

```
[root@server ~]# ipa-replica-manage re-initialize --
from=restored_master_FQDN
```

For further information on replication during restore and on restoration on other masters, see the `ipa-restore(1)` man page.

## 9.2.3. Restoring from an Encrypted Backup

If you want to restore from a backup encrypted with GPG, provide the full path to the private and public keys using the **--gpg-keyring** option. For example:

```
[root@server ~]# ipa-restore --gpg-keyring=/root/backup /path/to/backup
```

## **Part II. Managing User Identities in a Linux Domain**

## Chapter 10. 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.

### 10.1. 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.

#### 10.1.1. 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 10.1.2, “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 10.1.3, “Manually Mounting Home Directories”](#).

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

### 10.1.2. 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.

### 10.1.3. 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:

```
[bjensen@server ~]$ 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**:

```
[bjensen@server ~]$ 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 16, Using Automount](#).

## 10.2. Managing User Entries

### 10.2.1. About User Name Formats

The default length for user names is 32 characters.

IdM supports a wide range of user name formats, based on the following regular expression. Note that the trailing dollar sign (\$) symbol is permitted for Samba 3.x machine support.

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

System limits apply to the user names in IdM. Due to POSIX requirements, portable names are not allowed to start with hyphens (-).



#### Important

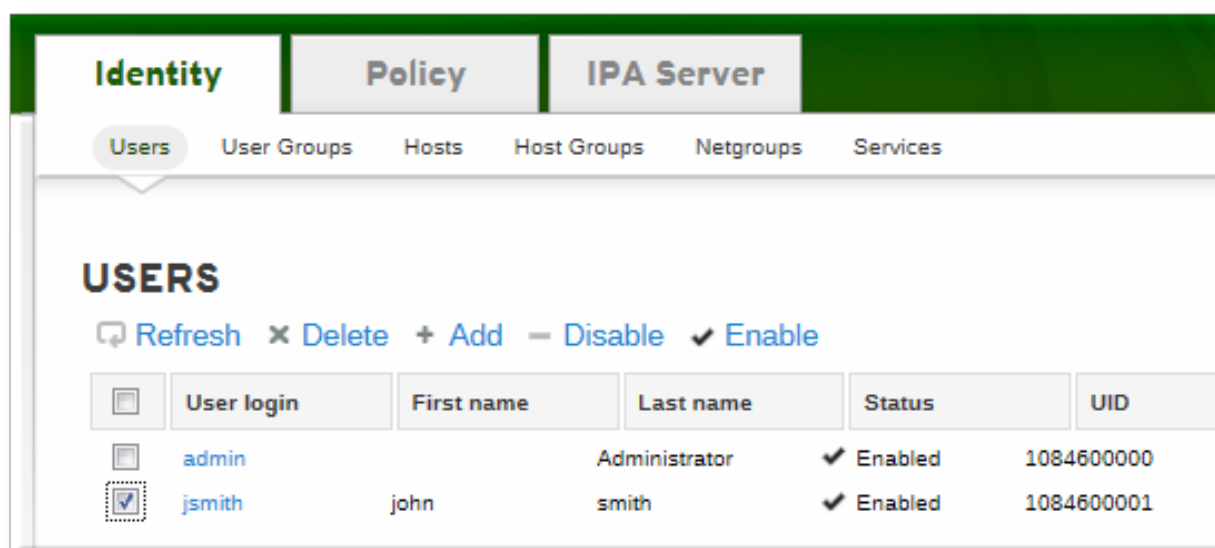
If the user name you enter contains uppercase characters, IdM converts them to lowercase characters when the user name is saved.

Even if you define a user name with one or more uppercase characters, IdM always requires the user to enter the user name all lowercase during log in. It is also not possible to add two user names that only differ in letter casing, for example **User** and **user**.

### 10.2.2. Adding Users

#### 10.2.2.1. 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.



3. 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.

**Add User**

User login:

First name: \*

Last name: \*

New Password:

Verify Password:

\* Required field

Buttons: Add, Add and Add Another, Add and Edit, Cancel

4. Click the **Add and Edit** button to go directly to the expanded entry page and fill in more attribute information, as in [Section 10.2.3.1, "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 IPA web interface. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Under 'Identity', there are sub-tabs: '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 it, it says 'jsmith is a member of:'. There are six tabs: 'Settings' (selected), 'User Groups (1)', 'Netgroups', 'Roles', 'HBAC Rules', and 'Sudo Rules'. Below these tabs are three buttons: 'Refresh', 'Reset', and 'Update'. The 'IDENTITY SETTINGS' section contains several text input fields: 'Job Title', 'First name: \* John', 'Last name: \* Smith', 'Full name: \* John Smith', 'Display name: John Smith', and 'Initials: JS'. The 'ACCOUNT SETTINGS' section contains: 'Status: Enabled: Click to Disable', 'User login: jsmith', 'Password: Reset Password', 'UID: \* 172200003', and a partially visible 'CUID: \*' field.

#### 10.2.2.2. From the Command Line

New user entries are added with the **user-add** command. Attributes (listed in [Table 10.2, “Default Identity Management User Attributes”](#)) can be added to the entry with specific values or the command can be run with no arguments.

```
[bjensen@server ~]$ 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:

```
[bjensen@server ~]$ 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
Full name: John Smith
Display name: John Smith
Initials: JS
Home directory: /home/jsmith
GECOS: John Smith
Login shell: /bin/sh
Kerberos principal: jsmith@EXAMPLE.COM
Email address: jsmith@example.com
UID: 882600007
GID: 882600007
Password: False
Member of groups: ipausers
Kerberos keys available: False
```

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.

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



## 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 10.8, “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**.

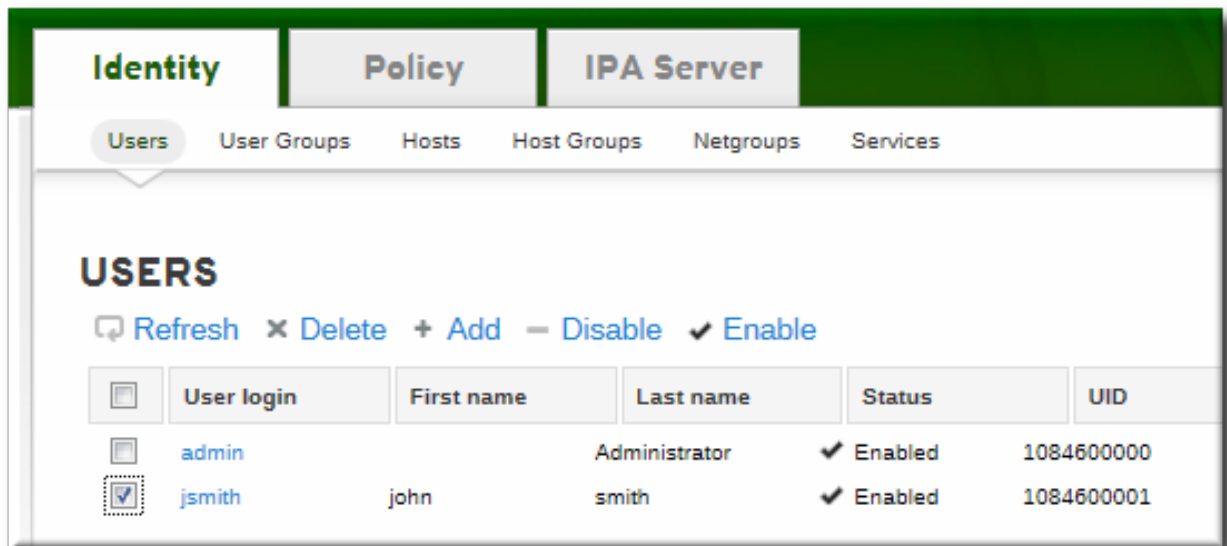
## 10.2.3. Editing Users

### 10.2.3.1. From the Web UI

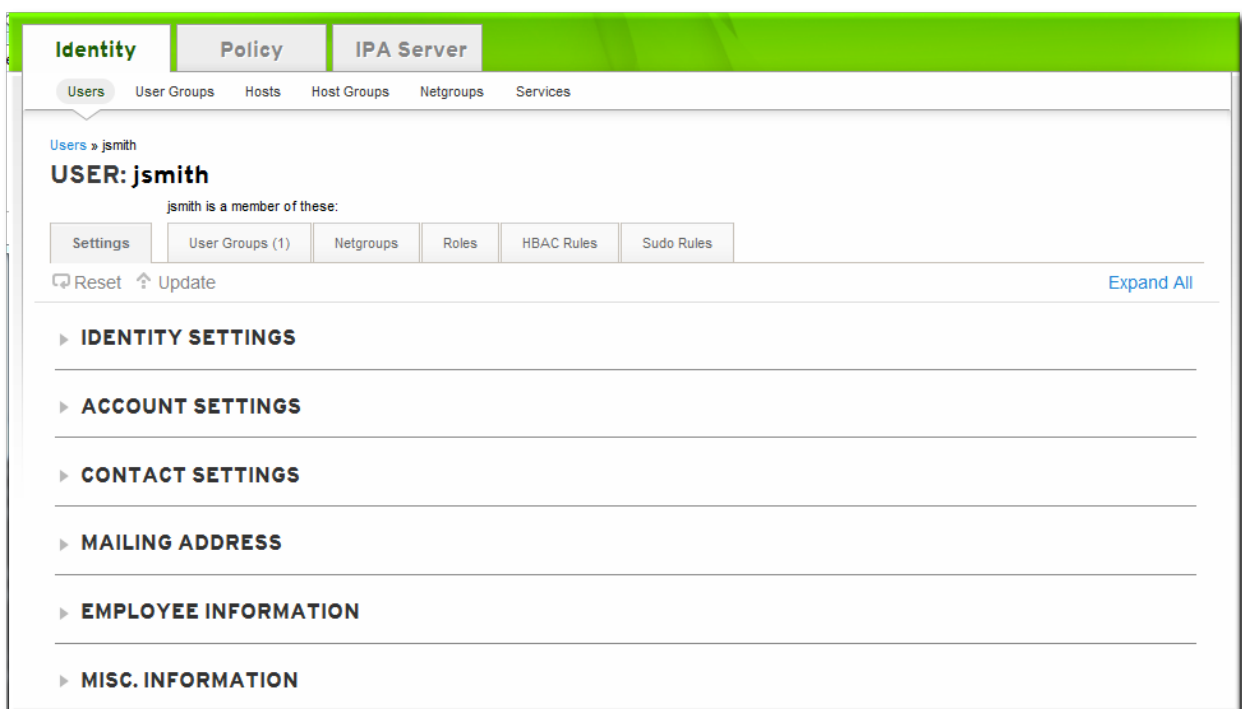
1. Open the **Identity** tab, and select the **Users** subtab.



- Click the name of the user to edit.



- 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 10.2, "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.



- 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

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

### 10.2.3.2. 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:

```
[bjensen@server ~]$ ipa user-mod loginID --attributeName=newValue
```

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

```
[bjensen@server ~]$ 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 10.1. Multiple Mail Attributes

A user is created first using his work email account.

```
[bjensen@server ~]$ ipa user-add jsmith --first=John --last=Smith --email=johnls@example.com
```

Then, his personal email account is added.

```
[bjensen@server ~]$ ipa user-mod jsmith --addattr=mail=johnnys@me.com
```

Both email addresses are listed for the user.

```
[bjensen@server ~]$ 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:

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

### 10.2.4. 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.



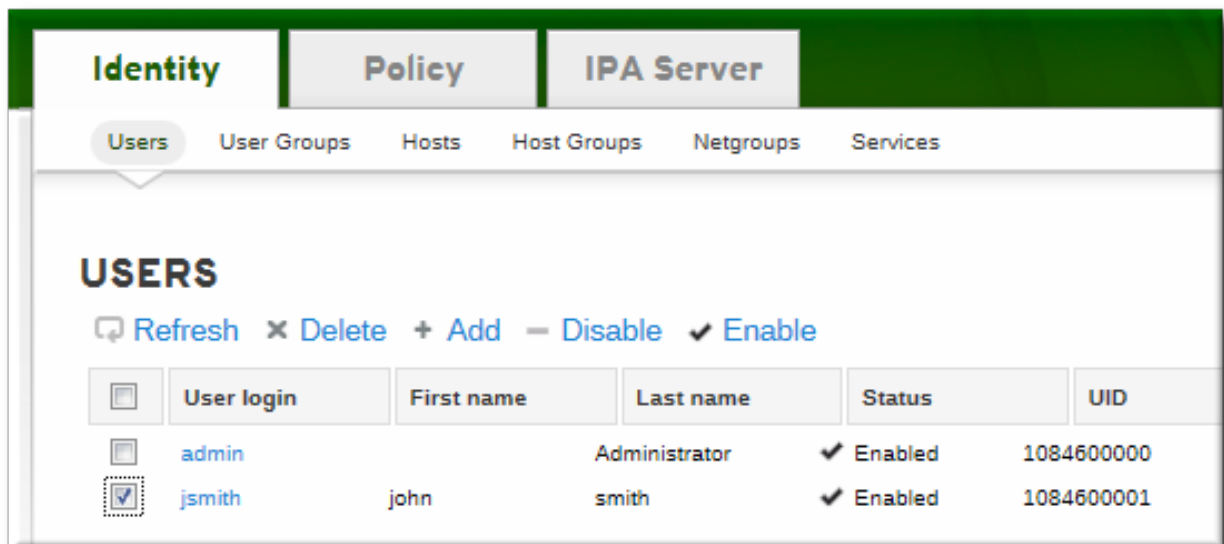
#### Note

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

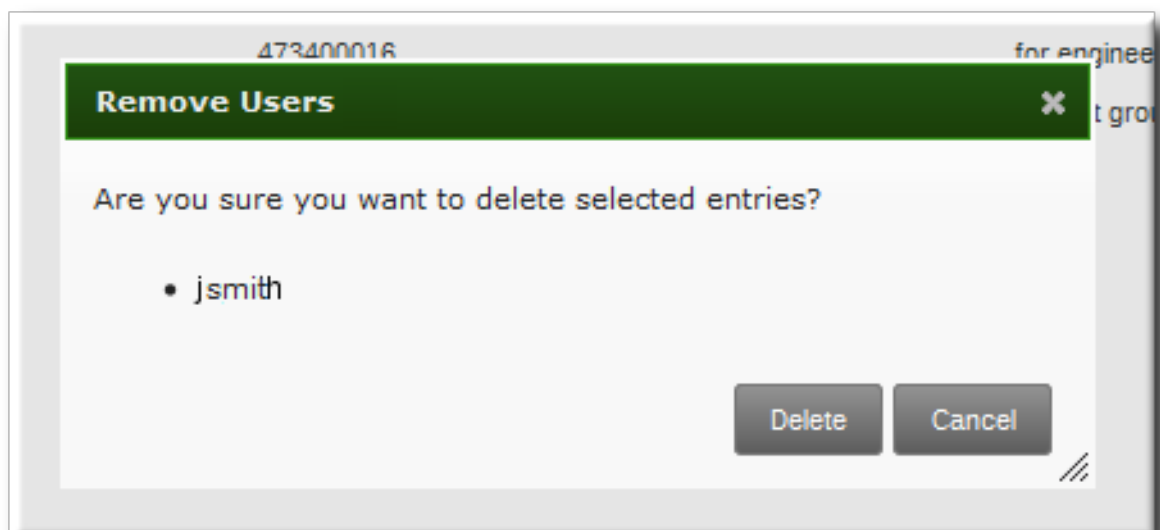
Alternatively, any user who belongs in the group management role can also add a new admin user.

#### 10.2.4.1. 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.



#### 10.2.4.2. From the Command Line

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

```
[bjensen@server ~]$ ipa user-del jsmith
```

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

```
[bjensen@server ~]$ 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.

### 10.3. 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 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. If the target system does not share a common home directory, the user must copy the public part of his SSH key to the target system he intends to log in to. The public portion of the SSH key must be copied to each target system the user intends to log in to.



## Note

SSH keys have to be distributed manually and separately to all machines in an environment.

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.

### 10.3.1. 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.

### 10.3.2. 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.

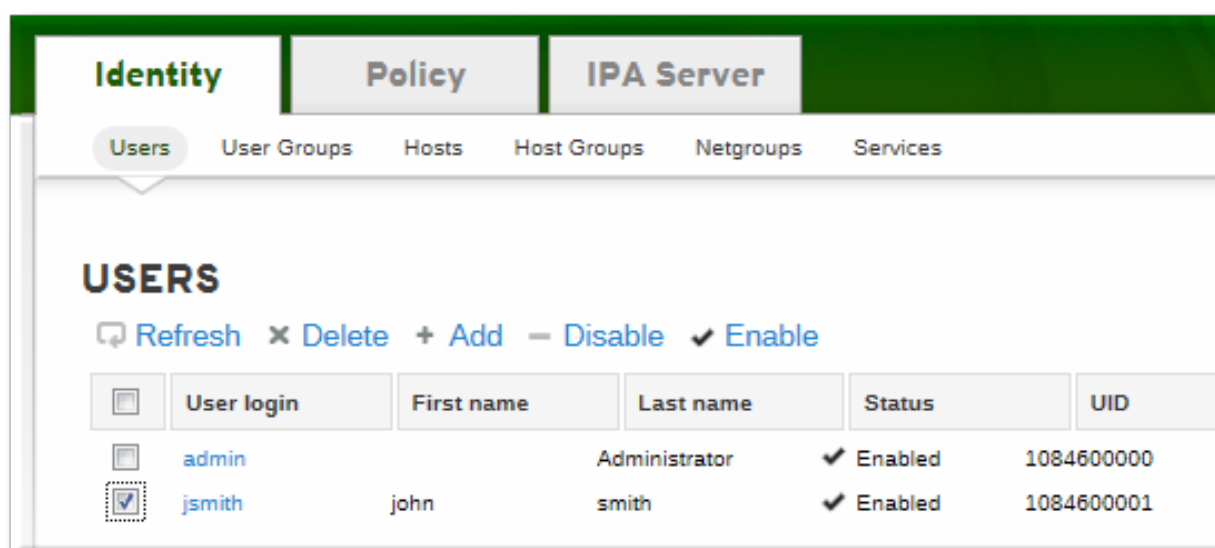
```

[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 shows the Red Hat Identity Management (IdM) web console interface. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Under the 'Identity' tab, there are sub-tabs for 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', and 'Services'. The 'Users' sub-tab is selected, and the user 'jsmith' is chosen. The main heading is 'USER: jsmith'. Below this, there is a dropdown menu for '-- select action --' and an 'Apply' button. A message states 'jsmith is a member of:'. Below this, there are tabs for 'Settings', 'User Groups (1)', 'Netgroups', 'Roles', 'HBAC Rules', and 'Sudo Rules'. The 'Settings' tab is selected. There are buttons for 'Refresh', 'Reset', and 'Update'. The main content area is divided into sections: 'IDENTITY SETTINGS', 'ACCOUNT SETTINGS', and 'PASSWORD POLICY'. The 'ACCOUNT SETTINGS' section contains the following fields: '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'. The 'SSH public keys' field is highlighted with a red box, and it contains the text 'SSH public keys: Add'. The 'PASSWORD POLICY' section is partially visible at the bottom.

6. Click the **Add** link by the **SSH public keys** field.

▼ **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+55SdJIrdMIPelfKJEzucLNioGsC7
59JVwpul6sQE1Kuu6Vec4Q5Jh7Ork2ERkxSxwwf+Pka5oN+M3sbaA+PBaQykBn4LAQ2
DvG0BSox8ObU2CydsoZuk+jQ7Ni13Qxka0rAllCgyGJT5T0nmFBHqTWhOKs81RBFbtmj
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 <sup>[2]</sup>.

**ACCOUNT SETTINGS**

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**

Home directory: **/home/jsmith**

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 10.1. 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 SSSD for managing user keys. This is covered in [the "Configuring Services: OpenSSH and Cached Keys" section in the System-Level Authentication Guide](#).

### 10.3.3. Uploading User SSH Keys Through the Command Line

The **--sshpubkey** option uploads the base64-encoded public key to the user entry. For example:

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

A real key also usually ends with an equal sign (=) but is longer.

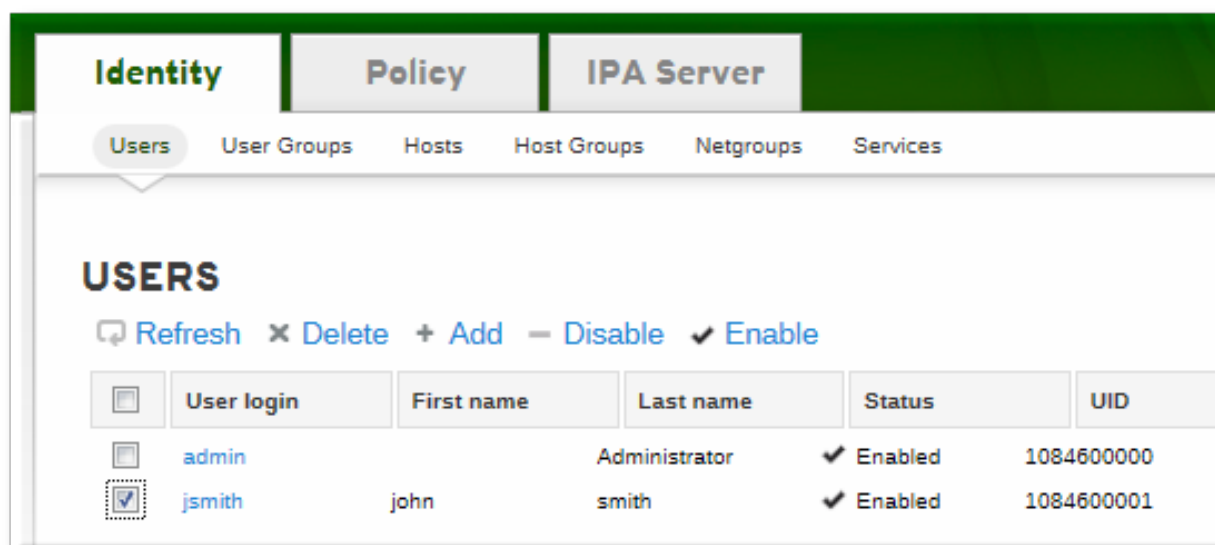
To upload more than one key, enter multiple **--sshpubkey** command-line parameters:

```
--sshpubkey="RjlzYQo=" --sshpubkey="ZEt0TAo="
```

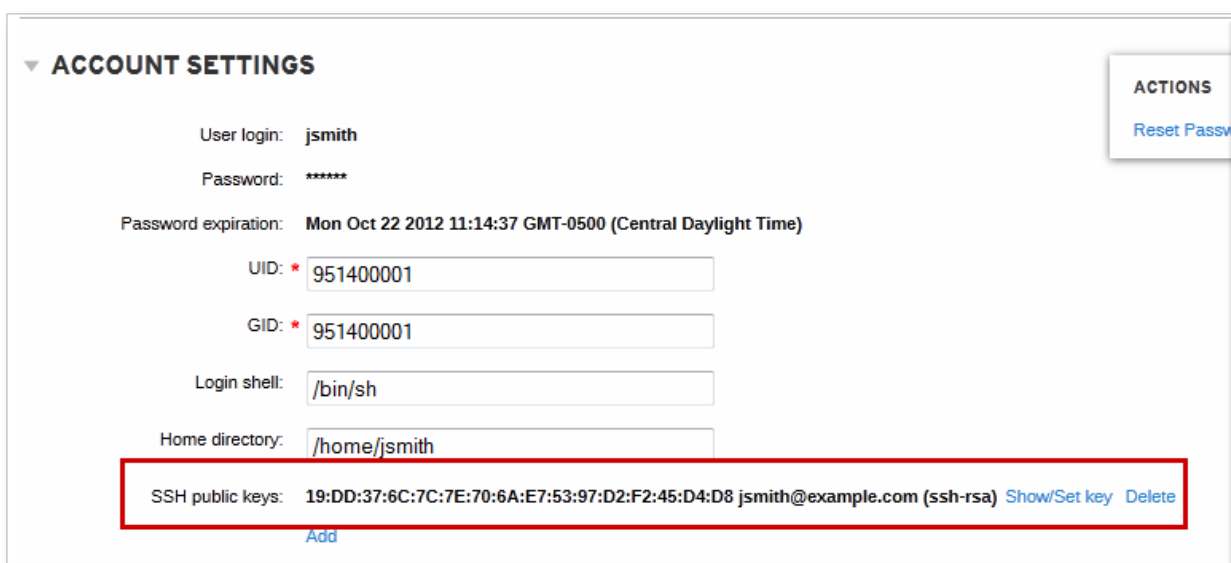
After uploading the user keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use SSSD for managing user keys. This is covered in [the "Configuring Services: OpenSSH and Cached Keys" section in the System-Level Authentication Guide](#).

### 10.3.4. 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
```

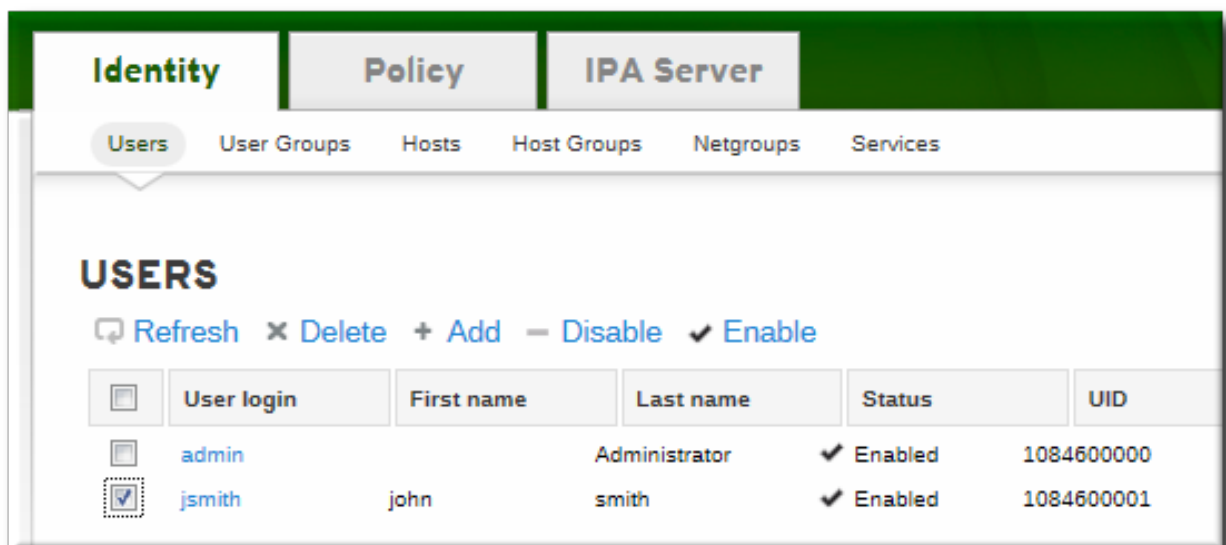
## 10.4. Changing Passwords

Password policies ([Chapter 17, Defining Password Policies](#)) and minimal access restrictions can be applied to a password change operation:

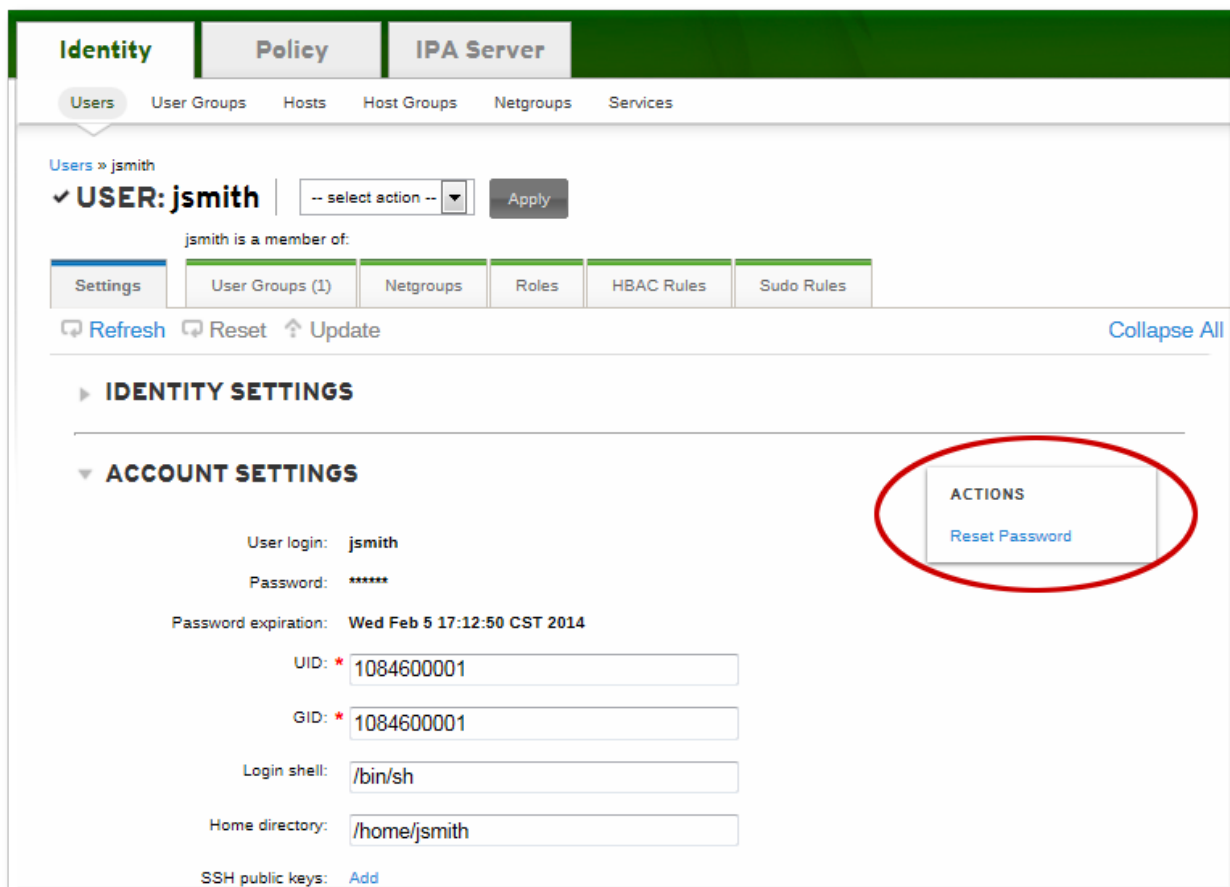
- ✧ Regular, non-administrative users can change only their personal passwords, and all passwords are constrained by the IdM password policies.
- This allows administrators to create intro passwords or to reset passwords easily, while still keeping the final password confidential. Since any password 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 login.
  - ✧ Changing a password as the LDAP Directory Manager user, *using LDAP tools*, overrides any IdM password policies.

### 10.4.1. From the Web UI

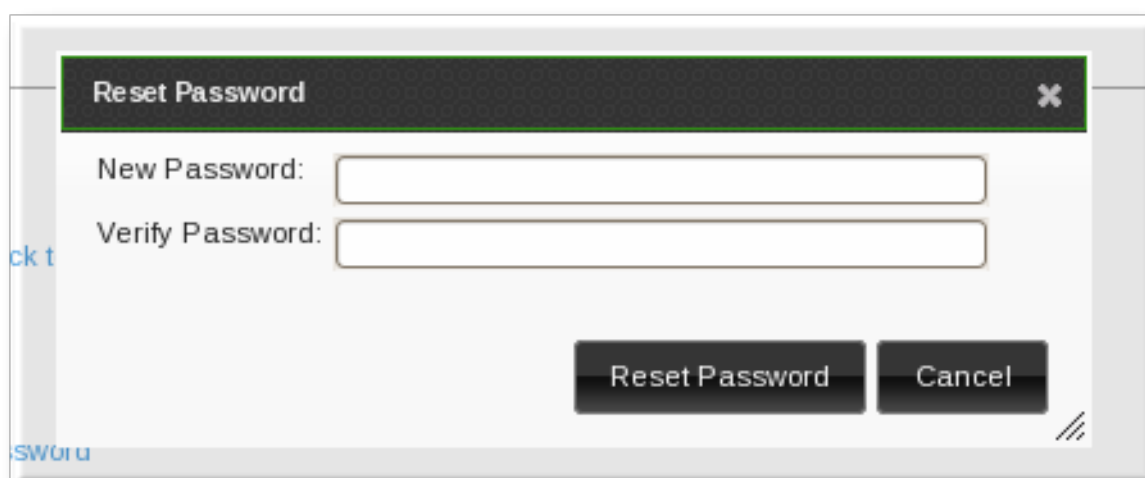
1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user for whom to reset the password. All users can change their own password; only administrators or users with delegated permissions can change other user's passwords.



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



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



### 10.4.2. 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
```

## 10.5. Enabling and Disabling User Accounts

User accounts can be deactivated or *disabled*. A disabled 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.

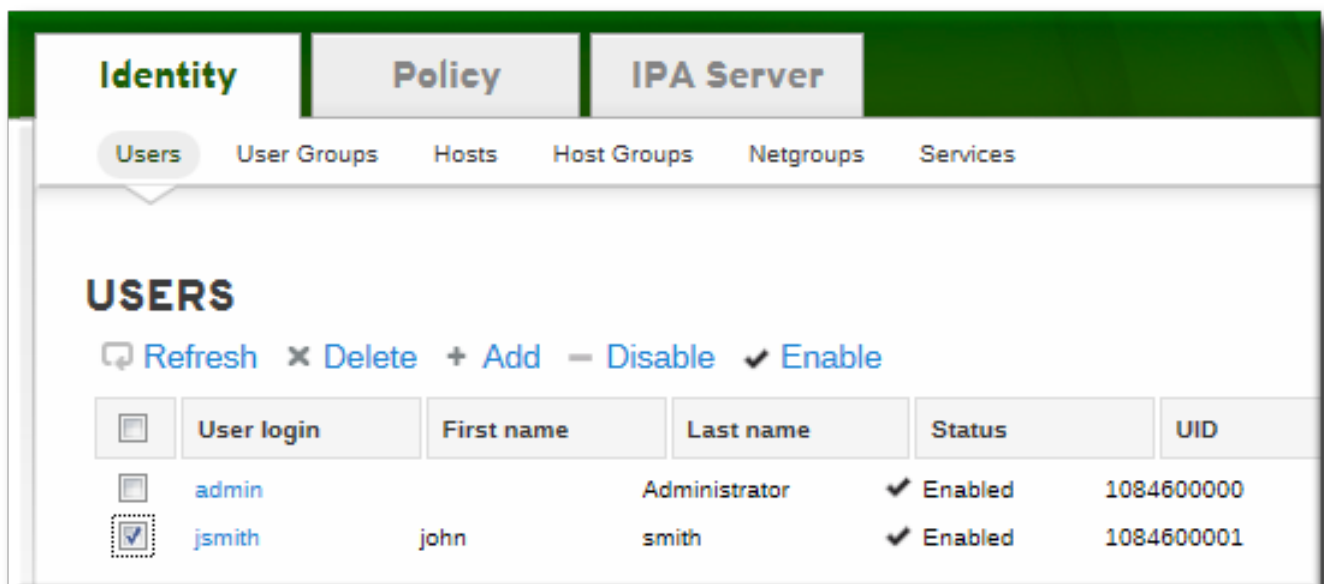


## Note

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

### 10.5.1. From the Web UI

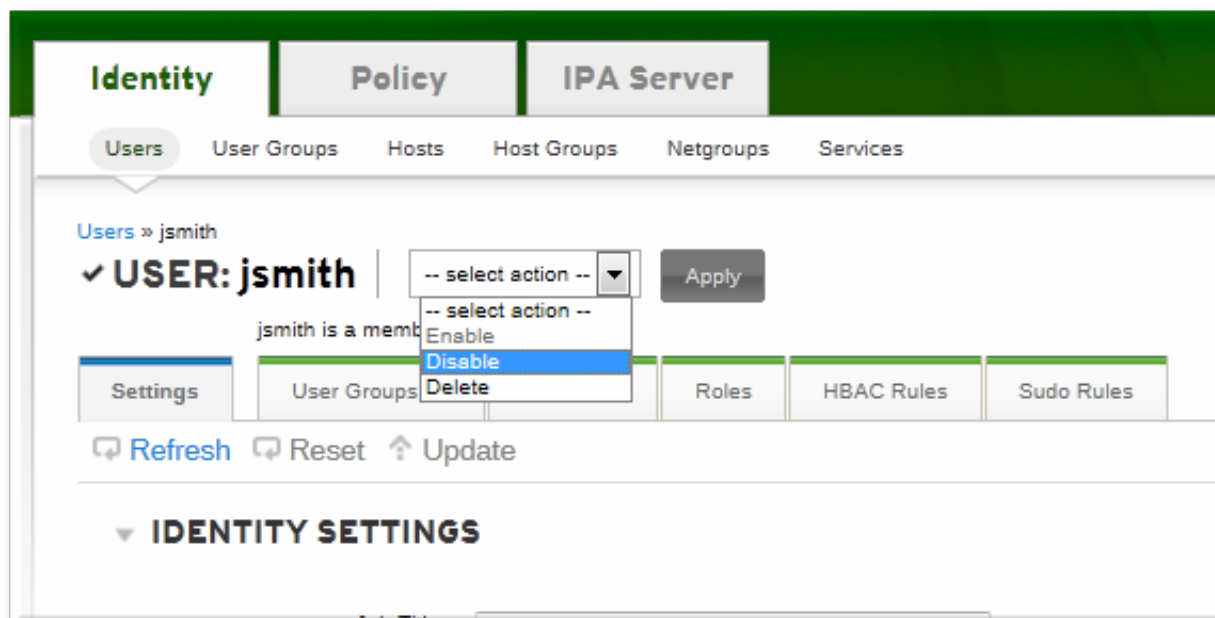
Multiple users can be disabled from the full users list by selecting the checkboxes by the desired users and then clicking the **Disable** link at the top of the list.



**Figure 10.2. Disable/Enable Options at the Top of the Users List**

A user account can also be disabled from the user's individual entry page.

1. Open the **Identity** tab, and select the **Users** subtab.
2. Click the name of the user to deactivate or activate.
3. In the actions drop-down menu, select the **Disable** item.



4. Click the **Accept** button.

When a user account is disabled, it is signified by a minus (-) icon for the user status in the user list and by the username on the entry page. Additionally, the text for the user is gray (to show it is inactive) instead of black.

USERS					
<input type="button" value="Refresh"/> <input type="button" value="Delete"/> <input type="button" value="Add"/> <input type="button" value="Disable"/> <input checked="" type="button" value="Enable"/>					
<input type="checkbox"/>	User login	First name	Last name	Status	UID
<input type="checkbox"/>	admin		Administrator	✓ Enabled	1084600000
<input checked="" type="checkbox"/>	jsmith	john	smith	— Disabled	1084600001

**Figure 10.3. Disable Icon for User Status**

### 10.5.2. From the Command Line

Users are enabled and disabled using **user-enable** and **user-disable** commands. All that is required is the user login. For example:

```
[bjensen@server ~]$ ipa user-disable jsmith
```

## 10.6. 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 17.6, “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
```

## 10.7. 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.

### 10.7.1. Listing User Private Groups

User private groups are specific to a single user and are only used by the system. They are private, so they are not viewable in the IdM UI. However, not every user has a private group, depending on the options when a user is created, so it can be useful to get a list of configured private groups within the IdM user domain. Private groups can be searched and listed by using the **--private** option with the **group-find** command. For example:

```
[root@server ~]# ipa group-find --private
-----
1 group matched
-----
  Group name: jsmith
  Description: User private group for jsmith
  GID: 1084600001
-----
Number of entries returned 1
-----
```

### 10.7.2. Disabling Private Groups for a Specific User

Private group creation can be disabled when a user is created by using the **--noprivate** option.

There is one thing to note when adding a user without a private group: the Linux system still expects a user GID for the new user. However, the one default user group (**ipausers**) is a non-POSIX group and, therefore, does not have an associated GID. So that the add operation does not fail, it is necessary either to set an explicit user GID with the **--gid**

option or to create a group with a GID and add the user to that group using an *automembership* rule (covered in [Chapter 22, Defining Automatic Group Membership for Users and Hosts](#)).

```
[jsmith@server ~]$ ipa user-add jsmith --first=John --last=Smith --  
noprivate --gid 10000
```

### 10.7.3. 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.

## 10.8. Managing Unique UID and GID Number Assignments

An IdM server generates user ID (UID) and group ID (GID) values and simultaneously ensures that replicas never generate the same IDs. The need for unique UIDs and GIDs might even be across IdM domains, if a single organization uses multiple separate domains.

### 10.8.1. ID Ranges

The UID and GID numbers are divided into *ID ranges*. By keeping separate numeric ranges for individual servers and replicas, the chances are minimal that an ID value issued for an entry is already used by another entry on another server or replica.

The Distributed Numeric Assignment (DNA) plug-in, as part of the back end 389 Directory Server instance for the domain, ensures that ranges are updated and shared between servers and replicas; the plug-in manages the ID ranges across all masters and replicas. Every server or replica has a current ID range and an additional



**next** ID range that the server or replica uses after the current range has been depleted. For more information about the DNA Directory Server plug-in, see the [Red Hat Directory Server Deployment Guide](#).

## 10.8.2. ID Range Assignments During Installation

During server installation, the **ipa-server-install** command by default automatically assigns a random current ID range to the installed server. The setup script randomly selects a range of 200,000 IDs from a total of 10,000 possible ranges. Selecting a random range in this way significantly reduces the probability of conflicting IDs in case you decide to merge two separate IdM domains in the future.

However, you can define a current ID range manually during server installation by using the following two options with **ipa-server-install**:

- » **--idstart** gives the starting value for UID and GID numbers; by default, the value is selected at random,
- » **--idmax** gives the maximum UID and GID number; by default, the value is the **--idstart** starting value plus 199,999.

If you have a single IdM server installed, a new user or group entry receives a random ID from the whole range. When you install a new replica and the replica requests its own ID range, the initial ID range for the server splits and is distributed between the server and replica: the replica receives half of the remaining ID range that is available on the initial master. The server and replica then use their respective portions of the original ID range for new entries. Also, if less than 100 IDs from the ID range that was assigned to a replica remain, meaning the replica is close to depleting its allocated ID range, the replica contacts the other available servers with a request for a new ID range.

A server receives an ID range the first time the DNA plug-in is used; until then, the server has no ID range defined. For example, when you create a replica from a master server, the replica does not receive an ID range immediately. The replica requests an ID range from the initial master only when the first ID is about to be assigned on the replica.



### Note

If the initial master stops functioning before the replica requests an ID range from it, the replica is unable to contact the master with a request for the ID range. An attempt to add a new user on the replica fails. In such situations, you can find out what ID range is assigned to the disabled master and assign an ID range to the replica manually, which is described in [Section 10.8.5, “Manual ID Range Extension and Assigning a New ID Range”](#).

## 10.8.3. Displaying Currently Assigned ID Ranges

To display which ID ranges are configured for a server, use the following commands:

- » **ipa-replica-manage dnarange-show** displays the current ID range that is set on all servers or, if you specify a server, only on the specified server, for example:

```
# ipa-replica-manage dnarange-show
masterA.example.com: 1001-1500
masterB.example.com: 1501-2000
```

```
masterC.example.com: No range set
```

```
# ipa-replica-manage dnarange-show masterA.example.com
masterA.example.com: 1001-1500
```

- » **ipa-replica-manage dnanextrange-show** displays the next ID range currently set on all servers or, if you specify a server, only on the specified server, for example:

```
# ipa-replica-manage dnanextrange-show
masterA.example.com: 1001-1500
masterB.example.com: No on-deck range set
masterC.example.com: No on-deck range set

# ipa-replica-manage dnanextrange-show masterA.example.com
masterA.example.com: 1001-1500
```

For more information about these two commands, see the `ipa-replica-manage(1)` man page.

#### 10.8.4. Automatic ID Range Extension After Deleting a Replica

When you delete a functioning replica, the **ipa-replica-manage del** command retrieves the ID ranges that were assigned to the replica and adds them as a next range to other available IdM replicas. This ensures that ID ranges remain available to be used by other replicas.

After you delete a replica, you can verify which ID ranges are configured for other servers by using the **ipa-replica-manage dnarange-show** and **ipa-replica-manage dnanextrange-show** commands, described in [Section 10.8.3, “Displaying Currently Assigned ID Ranges”](#).

#### 10.8.5. Manual ID Range Extension and Assigning a New ID Range

In certain situations, it is necessary to manually adjust an ID range:

##### An assigned ID range has been depleted

A replica has exhausted the ID range that was assigned to it, and requesting additional IDs failed because no more free IDs are available in the ID ranges of other replicas. You want to extend the ID range assigned to the replica. This might involve splitting an existing ID range or extending it past the initial configured ID range for the server. Alternatively, you might want to assign a new ID range.



##### Note

If you assign a new ID range, the UUIDs of the already existing entries on the server or replica stay the same. This does not pose a problem because even if you change the current ID range, IdM keeps a record of what ranges were assigned in the past.

##### A replica stopped functioning

ID range is not automatically retrieved when a replica dies and needs to be

deleted, which means the ID range previously assigned to the replica becomes unavailable. You want to recover the ID range and make it available for other replicas.

If you want to recover the ID range belonging to a server that stopped functioning and assign it to another server, first find out what are the ID range values using the **ipa-replica-manage dnanrange-show** command described in [Section 10.8.3, “Displaying Currently Assigned ID Ranges”](#), and then manually assign that ID range to the server. Also, to avoid duplicate UIDs or GIDs, make sure that no ID value from the recovered range was previously assigned to a user or group; you can do this by examining the UIDs and GIDs of existent users and groups.

To manually define the ID ranges, use the following two commands:

- » **ipa-replica-manage dnanrange-set** allows you to define the current ID range for a specified server:

```
# ipa-replica-manage dnanrange-set masterA.example.com 1250-1499
```

- » **ipa-replica-manage dnanextrange-set** allows you to define the next ID range for a specified server:

```
# ipa-replica-manage dnanextrange-set masterB.example.com 1001-5000
```

For more information about these commands, see the `ipa-replica-manage(1)` man page.



### Important

Be careful not to create overlapping ID ranges. If any of the ID ranges you assign to servers or replicas overlap, it could result in two different servers assigning the same ID value to different entries.

Do not set ID ranges that include UID values of 1000 and lower; these values are reserved for system use. Also, do not set an ID range that would include the **0** value; the SSSD service does not handle the **0** ID value.

When extending an ID range manually, make sure that the newly extended range is included in the IdM ID range; you can check this using the **ipa idrange-find** command. Run the **ipa idrange-find -h** command to display help for how to use **ipa idrange-find**.

## 10.8.6. Ensuring That ID Values Are Unique

It is recommended to avoid conflicting UIDs or GIDs. UIDs and GIDs should always be unique: two users should not have the same UID, and two groups should not have the same GID.

### Automatic ID assignment

When a user or a group is created interactively or without a manually specified ID number, the server assigns the next available ID number from the ID range to the user account. This ensures that the UID or GID is always unique.

### Manual ID assignment

When you assign an ID to a user or a group entry manually, the server does not verify that the specified UID or GID is unique; it does not warn you of a conflict if you choose a value that is already used by another entry.

As explained in [Section 10.8.7, “Repairing Changed UID and GID Numbers”](#), the SSSD service does not handle entries with identical IDs. If two entries share the same ID number, a search for this ID only returns the first entry. However, if you search for other attributes or run the **ipa user-find --all** command, both entries are returned.

UIDs and GIDs are both selected from the same ID range. A user and a group can have the same ID; no conflict arises in this situation because the UID and the GID are set in two different attributes: **uidNumber** and **gidNumber**.



## Note

Setting the same ID for both a user and a group allows you to configure user private groups. To create a unique system group for a user in this way, set the same ID value for a user and also for a group, in which the only member is the mentioned user.

### 10.8.7. Repairing Changed UID and GID Numbers

When a user logs into an IdM system or service, SSSD on that system caches their user name together with the UID and GID of the user. SSSD then uses the UID as the identifying key for the user. If a user with the same user name but a different UID attempts to log into the system, SSSD registers two different UIDs and assumes that there are two different users with conflicting user names. This can pose a problem if a UID of a user changes. In such a situation, SSSD incorrectly interprets the user with a modified UID as a new user, instead of recognizing that it as the same user with a different UID. If the UID of an existing user changes, the user cannot log into SSSD and associated services and domains. This also affects client applications that use SSSD for identity information.

To work around this problem, if a UID or GID changes, clear the SSSD cache, which ensures that the user is able to log in again. For example, to clear the SSSD cache for a specified user, use the **sss\_cache** utility as follows:

```
[root@server ~]# sss_cache -u user
```

## 10.9. 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 10.1. Default Identity Management User Object Classes**

Description	Object Classes
-------------	----------------

Description	Object Classes
IdM object classes	ipaobject ipasshuser
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 10.1, “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 10.9.4, “Specifying Default User and Group Attributes”](#).

**Table 10.2. 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 [b]	Optional
User ID number [c]	--uid	Default
Group ID number [c]	--gidnumber	Default
Street address	--street	Optional
City	--city	Optional
State/Province	--state	Optional
Zip code	--postalcode	Optional
Telephone number	--phone	Optional

UI Field	Command-Line Option	Required, Optional, or Default
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
	--noprivate	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] The script prompts for the new password, rather than accepting a value with the argument.

[c] 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 10.8, “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**.

### 10.9.1. 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 10.9, “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.

### 10.9.2. 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.

#### 10.9.2.1. 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:

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:  [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.

### 10.9.2.2. 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.

All object classes must be included in the list of object classes. The information passed with the **config-mod** command overwrites the previous values. This can be done by specifying each object class with a **--userobjectclasses** argument or by listing all of the object classes in a comma-separated list inside curly braces, such as `{attr1,attr2,attr3}`. For long lists, it can be easier to use the curly braces than multiple options. For example:

```
[bjensen@server ~]$ ipa config-mod --
userobjectclasses={top,person,organizationalperson,inetorgperson,i
netuser,posixaccount,krbprincipalaux,krbticketpolicyaux,ipaobject,
ipasshuser,employeeinfo}
```

### 10.9.3. 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.

#### 10.9.3.1. 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 web interface. 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 selected. Below the subtabs, the main heading is 'CONFIGURATION'. There are three buttons: '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.

CONFIGURATION											
<a href="#">Refresh</a> <a href="#">Reset</a> <a href="#">Update</a>											
<b>GROUP OPTIONS</b>											
Group search fields:	<input type="text" value="cn,description"/>										
Default group objectclasses:	<table border="1"> <tbody> <tr> <td>top</td> <td><a href="#">Delete</a></td> </tr> <tr> <td>groupofnames</td> <td><a href="#">Delete</a></td> </tr> <tr> <td>nestedgroup</td> <td><a href="#">Delete</a></td> </tr> <tr> <td>ipausergroup</td> <td><a href="#">Delete</a></td> </tr> <tr> <td>ipaobject</td> <td><a href="#">Delete</a></td> </tr> </tbody> </table>	top	<a href="#">Delete</a>	groupofnames	<a href="#">Delete</a>	nestedgroup	<a href="#">Delete</a>	ipausergroup	<a href="#">Delete</a>	ipaobject	<a href="#">Delete</a>
top	<a href="#">Delete</a>										
groupofnames	<a href="#">Delete</a>										
nestedgroup	<a href="#">Delete</a>										
ipausergroup	<a href="#">Delete</a>										
ipaobject	<a href="#">Delete</a>										
<a href="#">Add</a>											

- 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.

- When the changes are complete, click the **Update** link at the top of the **Configuration** page.

### 10.9.3.2. From the Command Line

- 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](#).
- 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.

All object classes must be included in the list of object classes. The information passed with the **config-mod** command overwrites the previous values. This can be done by specifying each object class with a **--groupobjectclasses** argument or by listing all of the object classes in a comma-separated list inside curly braces, such as `{attr1,attr2,attr3}`. For long lists, it can be easier to use the curly braces than multiple options. For example:

```
[bjensen@server ~]$ ipa config-mod --  
groupobjectclasses={top,groupofnames,nestedgroup,ipausergroup,ipaob  
ject,ipasshuser,employeegroup}
```

### 10.9.4. Specifying Default User and Group Attributes

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.

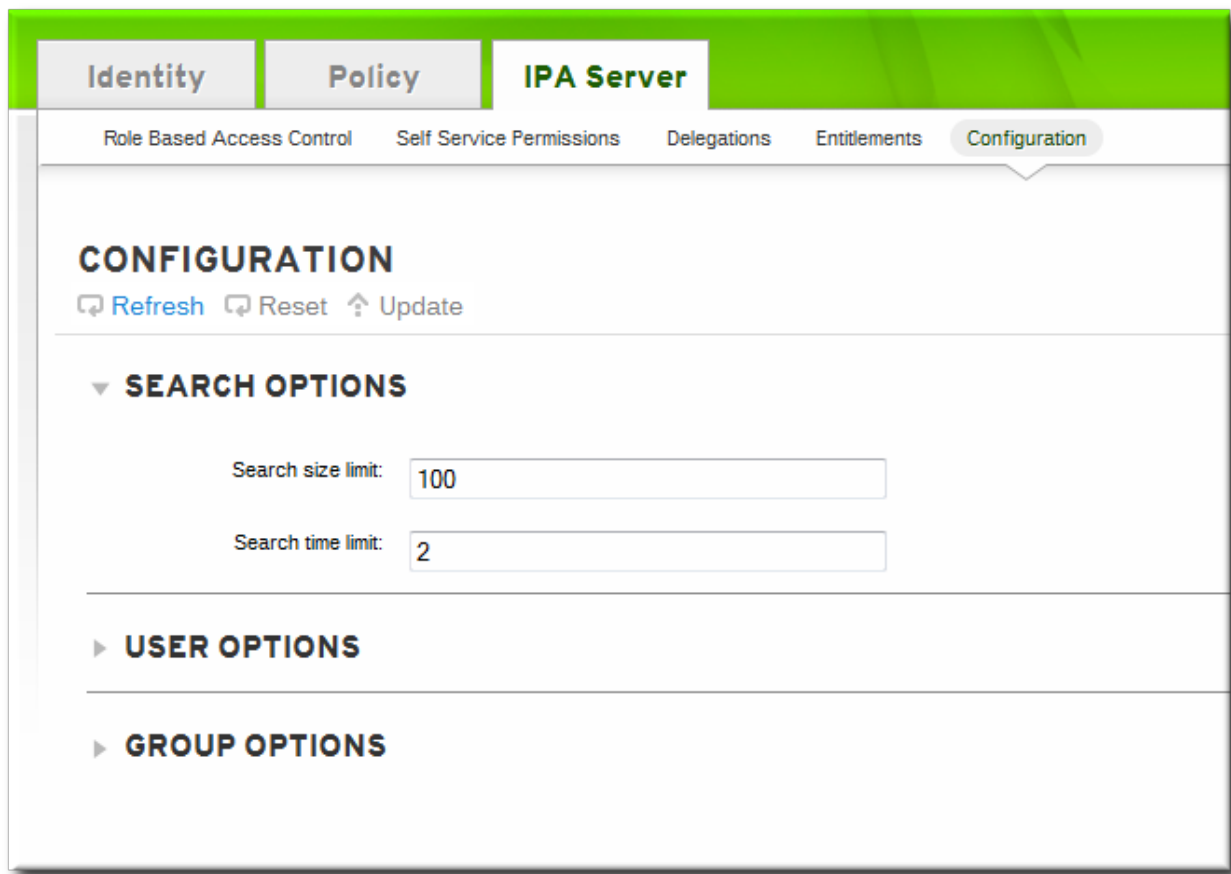
**Table 10.3. Default User Parameters**

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 <b>/home</b> .
Default shell	--defaultshell	Sets the default shell to use for users. The default value is <b>/bin/sh</b> .
Default user group	--defaultgroup	Sets the default group to which all newly created accounts are added. The default value is <b>ipausers</b> , 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 DNs for client certificates. This is configured when the server is set up.

Field	Command-Line Option	Descriptions
Default user object classes	--userobjectclasses	Defines an object class that is used to create IdM user accounts. This can be invoked multiple times. The complete list of object classes must be given because the list is overwritten when the command is run.
Default group object classes	--groupobjectclasses	Defines an object class that is used to create IdM group accounts. This can be invoked multiple times. The complete list of object classes must be given because the list is overwritten when the command is run.
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.

#### 10.9.4.1. Viewing Attributes 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.



#### 10.9.4.2. Viewing Attributes 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.

```
[bjensen@server ~]$ kinit admin
[bjensen@server ~]$ ipa config-show --all
dn: cn=ipaConfig,cn=etc,dc=example,dc=com
Maximum username length: 32
Home directory base: /home
Default shell: /bin/sh
Default users group: ipausers
Default e-mail domain: 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, ipasshuser
Password Expiration Notification (days): 4
Password plugin features: AllowNThash
SELinux user map order: guest_u:s0$guest_u:s0$user_u:s0$staff_u:s0-
s0:c0.c1023$unconfined_u:s0-s0:c0.c1023
```

```
Default SELinux user: unconfined_u:s0-s0:c0.c1023
Default PAC types: MS-PAC, nfs:NONE
cn: ipaConfig
objectclass: nsContainer, top, ipaGuiConfig, ipaConfigObject
```

## 10.10. Managing User Groups

User groups are a way of centralizing control over important management tasks, particularly access control and password policies. Four 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.
- `trusted admins`, which contains administrative users used to manage Active Directory trusts.
- `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.



### 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.

### 10.10.1. Types of Groups in IdM

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.

Automembership rules allow new users to be added to groups automatically, using attributes in the user entry to determine what groups the user should belong to.

Automembership rules are covered in [Chapter 22, Defining Automatic Group Membership for Users and Hosts](#).

The way groups are defined in IdM is simple, but there are different configuration options for groups which can change what kinds of members can be added.

Some types of groups in IdM are based not on how members are added, but rather where the member entries originate:

- Internal groups (the default), where all members belong to the IdM domain.
- External groups, where some or all of the members exist in an identity store outside of the IdM domain. This can be a local system, an Active Directory domain, or a directory service.

Another difference is whether groups are created with POSIX attributes. Most Linux users require some kind of POSIX attributes, but groups which interact with Active Directory or Samba must be non-POSIX. By default, IdM creates POSIX groups. There is an explicit option to create a non-POSIX group (by adding the **--nonposix** option).

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.

### 10.10.2. Group Object Classes

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 10.4. Default Identity Management Group Object Classes**

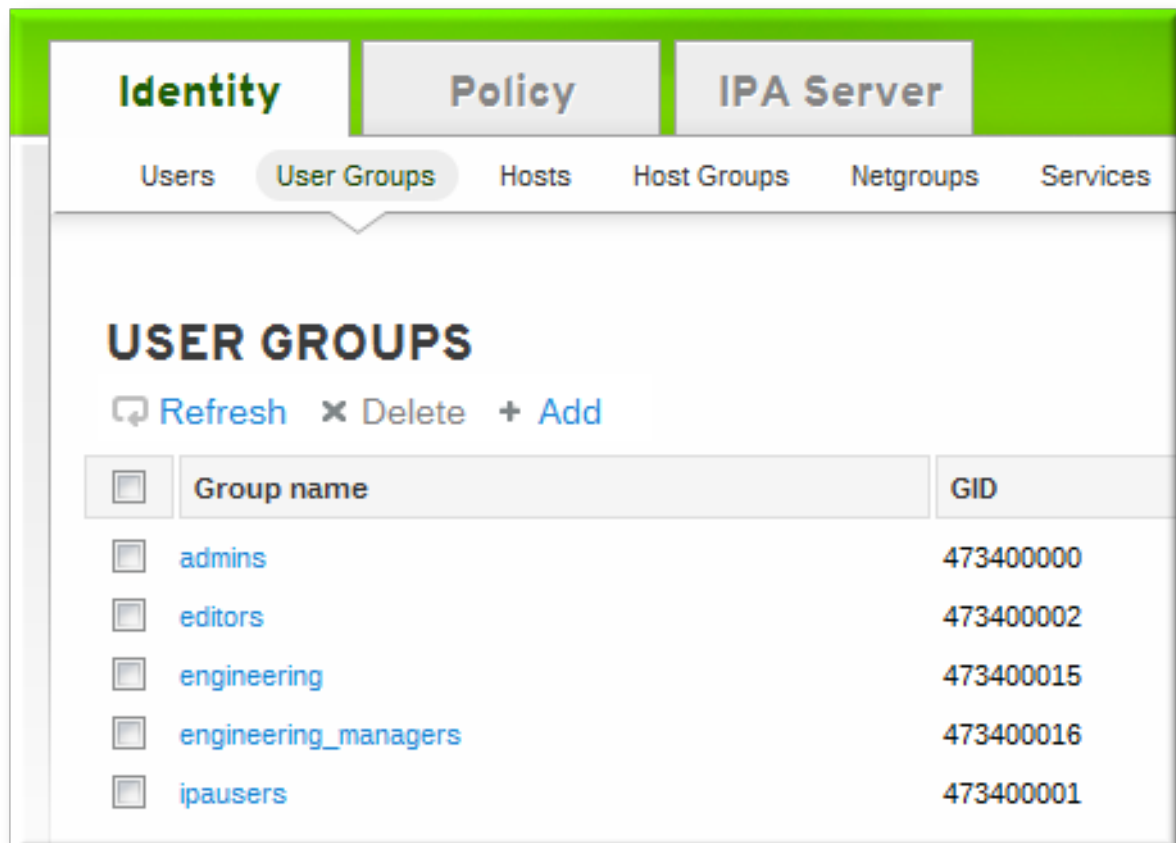
Description	Object Classes
IdM object classes	ipaobject
	ipausergroup
	nestedgroup
Group object classes	groupofnames

#### 10.10.2.1. Creating User Groups

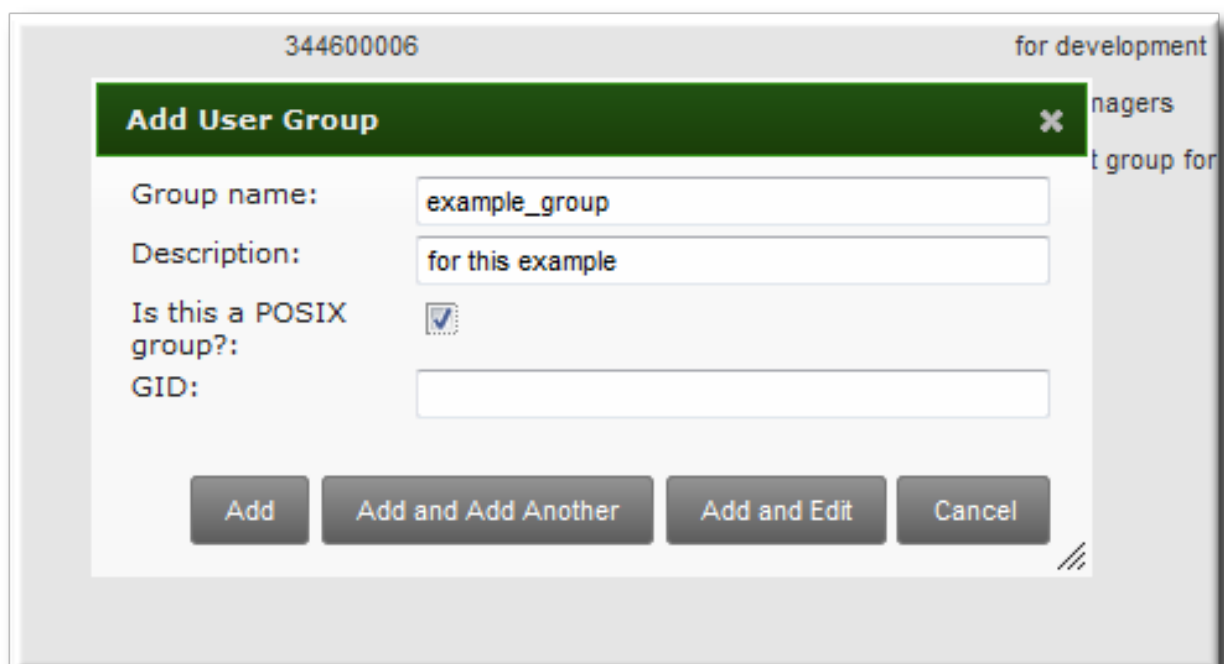
##### 10.10.2.1.1. With the Web UI

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 10.10.2.2.1, “With the Web UI \(Group Page\)”](#).

#### 10.10.2.1.2. 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.

```
[bjensen@server ~]$ 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:

```
[bjensen@server ~]$ 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:

```
[bjensen@server ~]$ 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 10.8, “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.

### 10.10.2.2. Adding Group Members

#### 10.10.2.2.1. With the Web UI (Group Page)



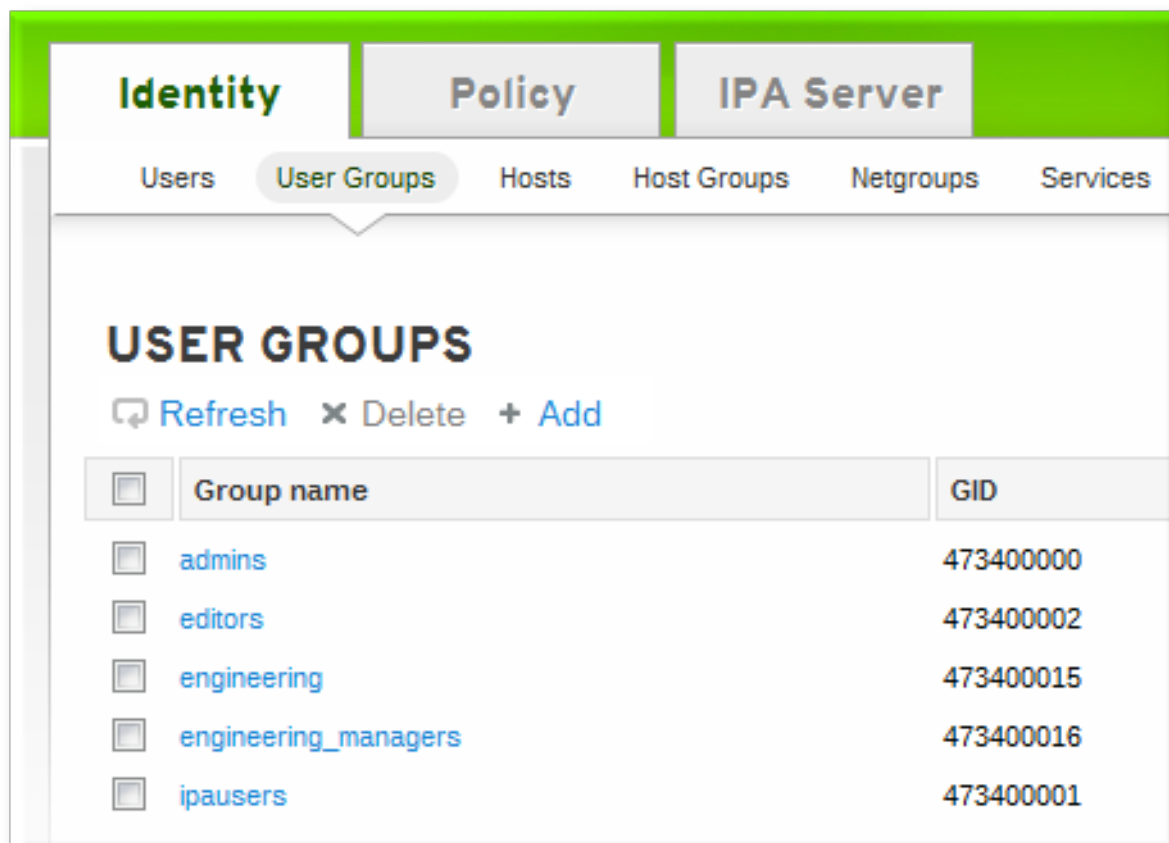
## Note

This procedure adds a user to a group. User groups can contain other user groups as their members. These are *nested* groups.

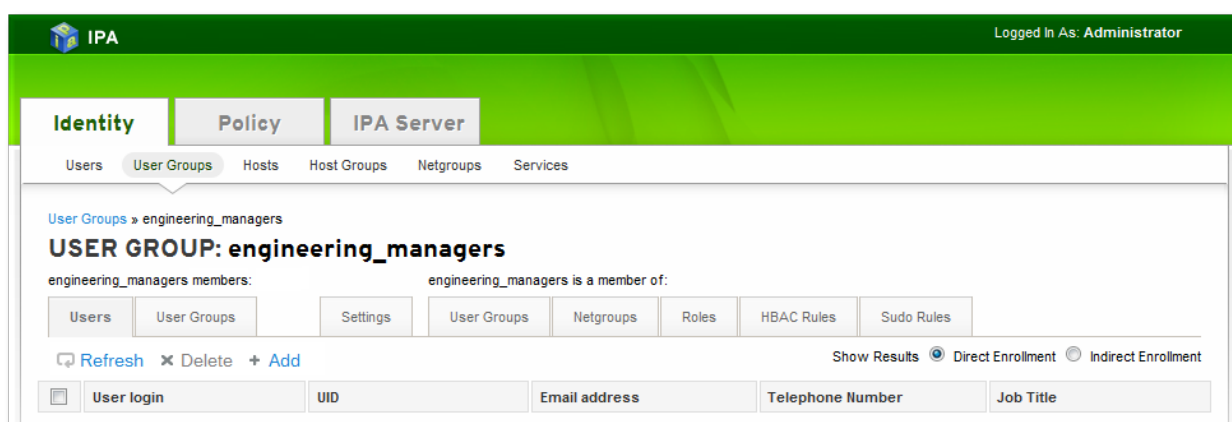
It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

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.

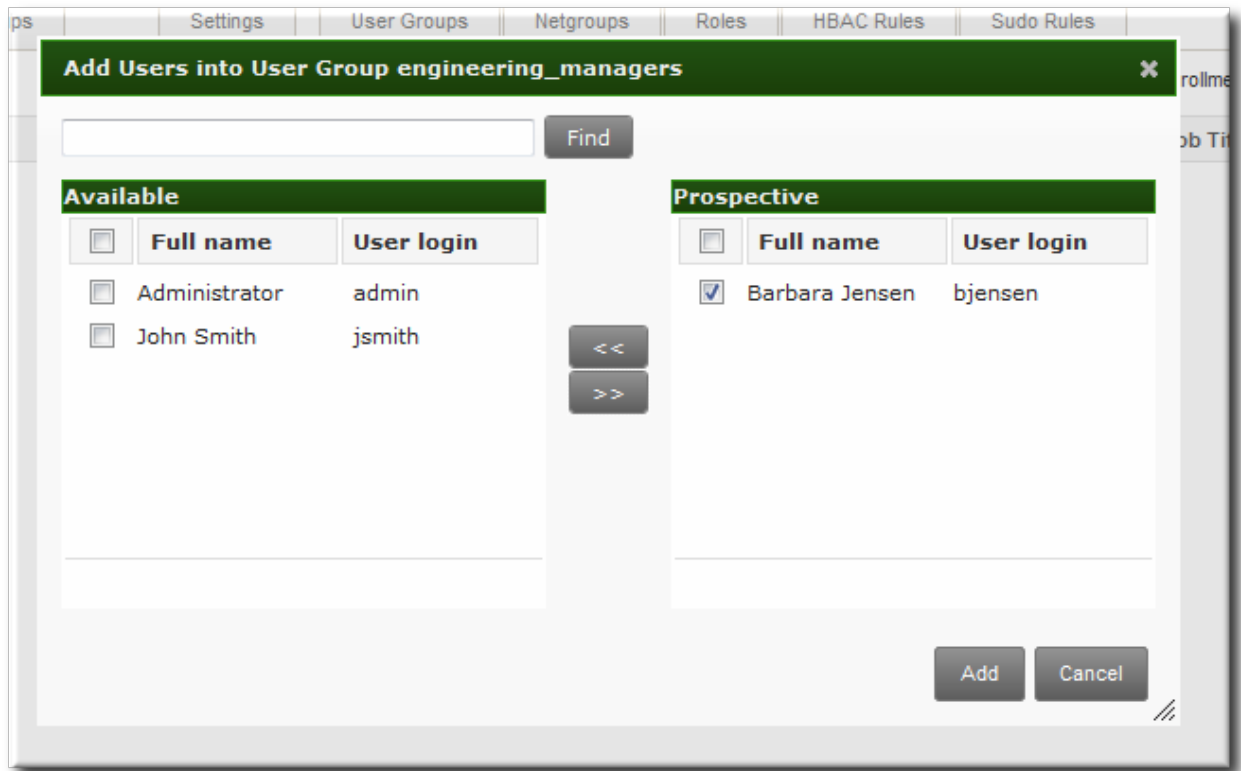
1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Click the name of the group to which to add members.



- 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.



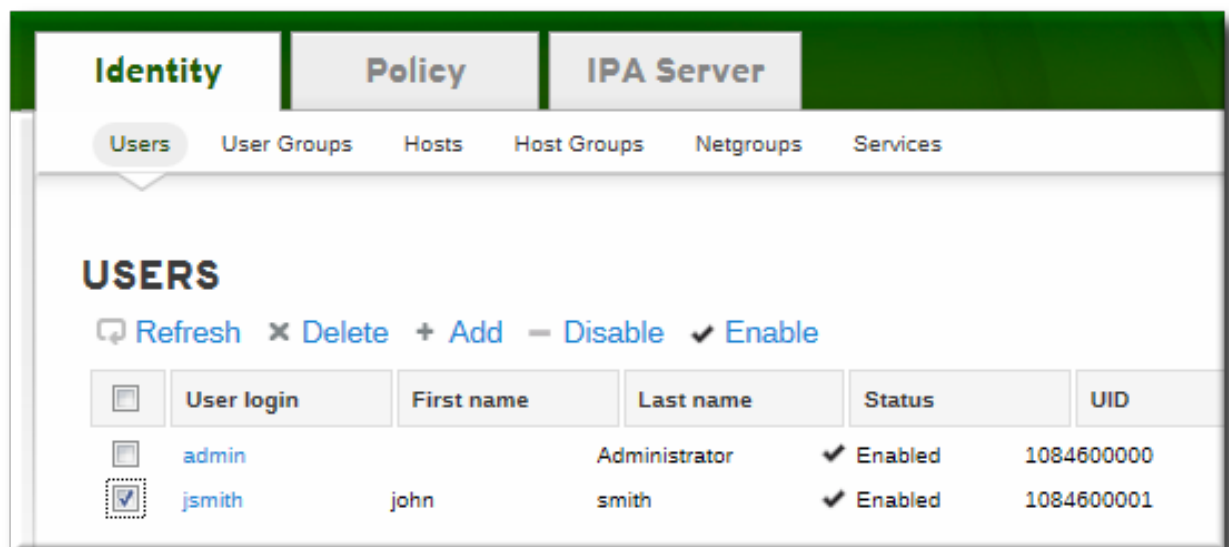
5. Click the **Add** button.

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.

#### 10.10.2.2.2. With the Web UI (User's Page)

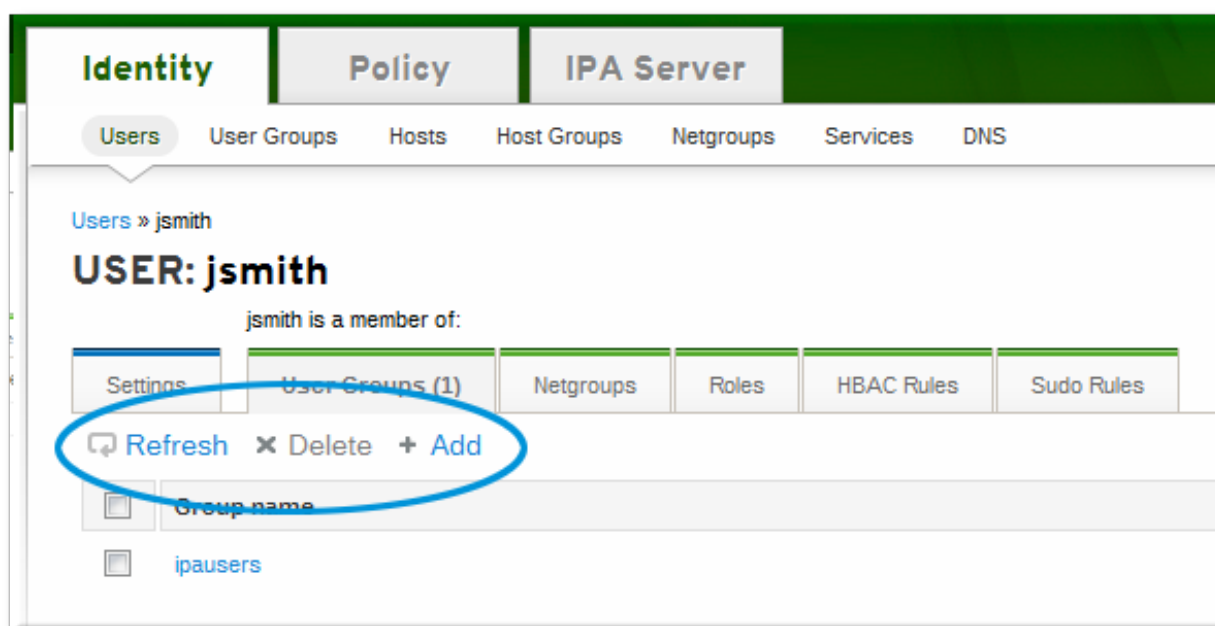
Users can also be added to a group through the user's page.

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

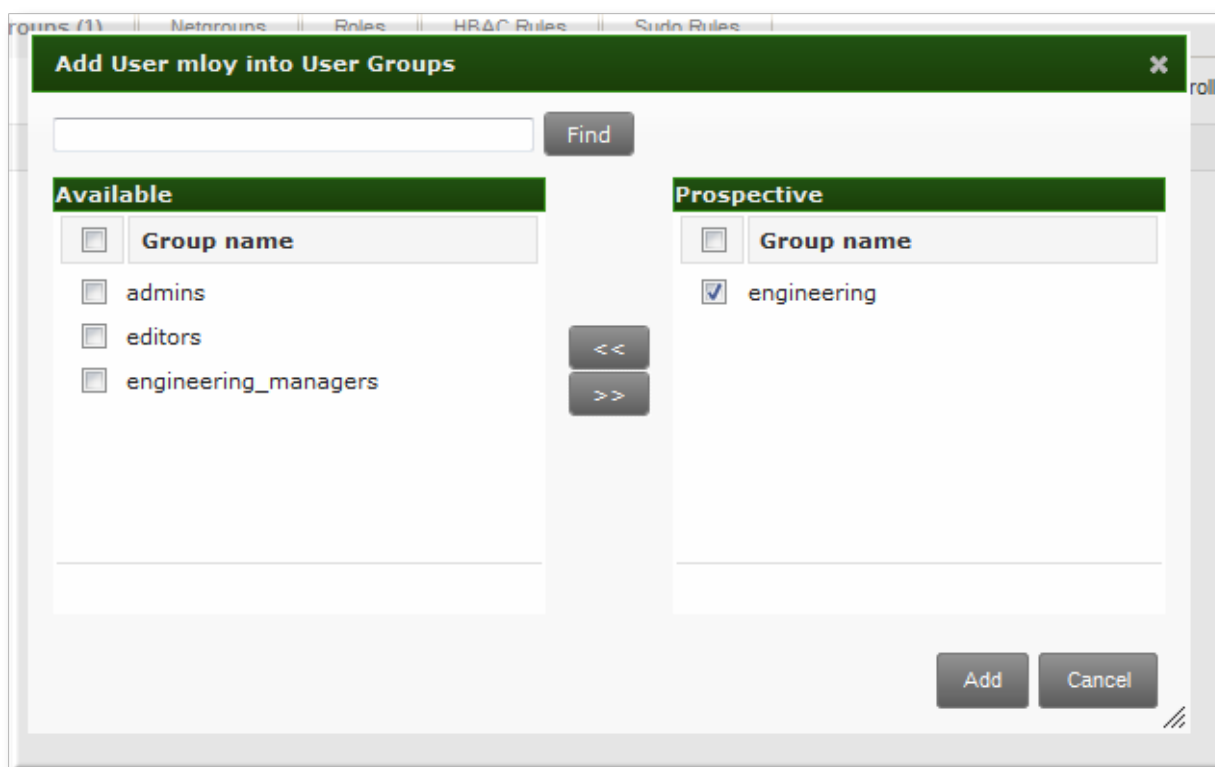


3. Open the **User Groups** tab on the user entry page.

- Click the **Add** link at the top of the task area.



- 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.



- Click the **Add** button.

#### 10.10.2.2.3. 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 the users or groups to add. Lists of entries can be set by using the option multiple times with the same command invocation or by listing the options in a comma-separated list inside curly braces, such as `--option={val1,val2,val3}`.

```
[bjensen@server ~]$ ipa group-add-member groupName [--users=user1 ...]
[--groups=groups1 ...]
```

For example, this adds three users to the **engineering** group:

```
[bjensen@server ~]$ ipa group-add-member engineering --users=jsmith --
users=bjensen --users=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:

```
[bjensen@server ~]$ ipa group-add-member engineering --groups=dev --
groups=qe1 --groups=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:

```
[bjensen@server ~]$ ipa group-show examplegroup
Group name: examplegroup
Description: for examples
GID: 93200002
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.

```
[bjensen@server ~]$ ipa group-remove-member engineering --users=jsmith

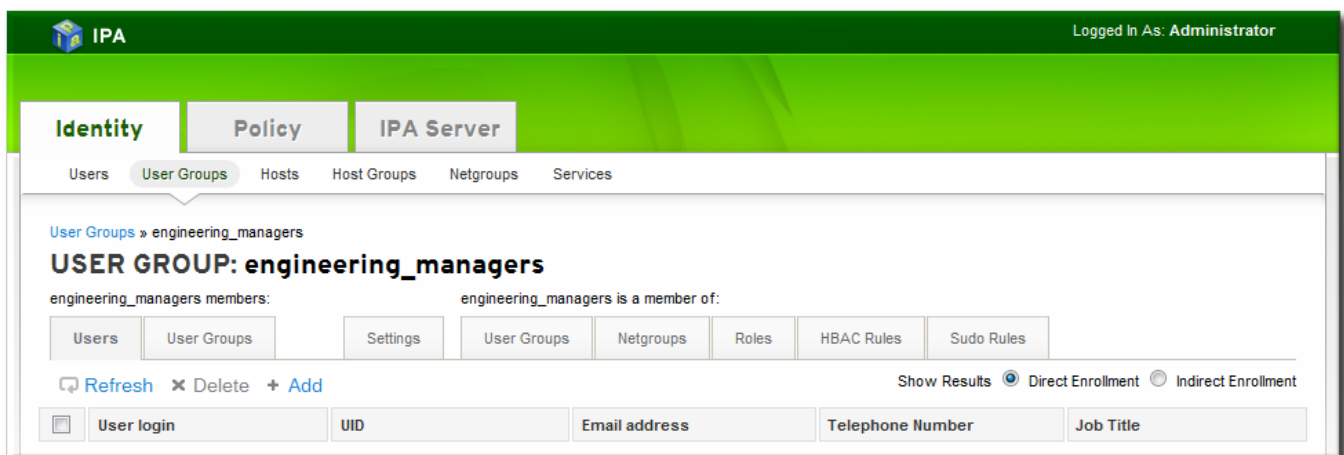
Group name: engineering
Description: for engineers
GID: 855800009
Member users: bjensen,mreynolds
-----
Number of members removed 1
-----
```

#### 10.10.2.2.4. 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 10.4. Indirect and Direct Members**

Being able to track indirect members makes it easier to assign group membership properly, without duplicating membership.

#### 10.10.2.3. 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.

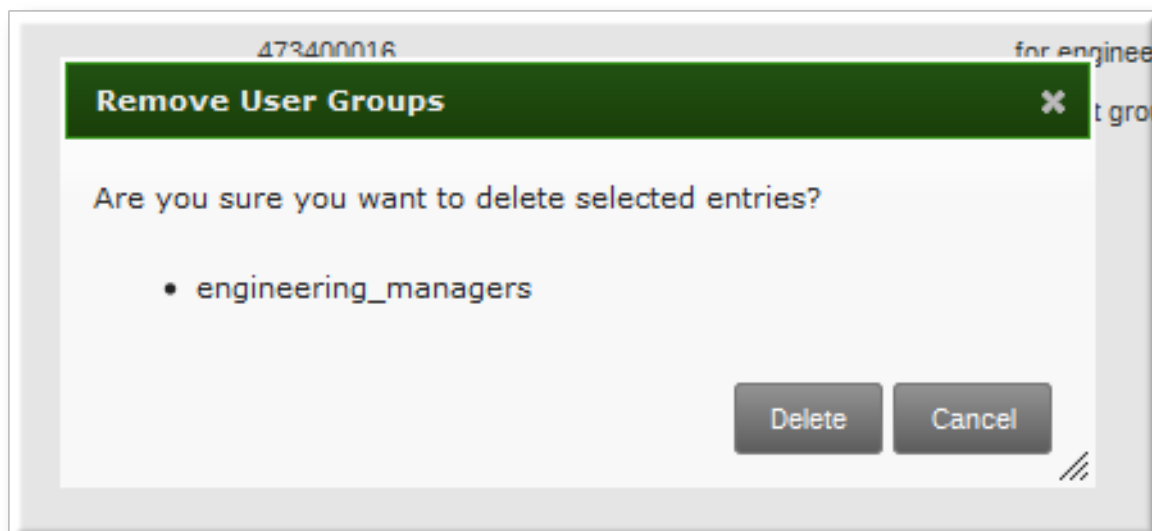
### 10.10.2.3.1. With the Web UI

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Select the checkbox by the name of the group to delete.

The screenshot shows the Red Hat Identity Management (IdM) Web UI. The 'Identity' tab is selected, and the 'User Groups' subtab is active. The 'USER GROUPS' section displays a table of user groups. The table has two columns: 'Group name' and 'GID'. There are five groups listed: 'admins', 'editors', 'engineering', 'engineering\_managers', and 'ipausers'. Each group has a checkbox to its left. Above the table, there are links for 'Refresh', 'Delete', and 'Add'. The 'Delete' link is highlighted with a red 'x' icon.

<input type="checkbox"/>	Group name	GID
<input type="checkbox"/>	admins	473400000
<input type="checkbox"/>	editors	473400002
<input type="checkbox"/>	engineering	473400015
<input type="checkbox"/>	engineering_managers	473400016
<input type="checkbox"/>	ipausers	473400001

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



#### 10.10.2.3.2. With the Command Line

The **group-del** command to deletes the specified group. For example:

```
[bjensen@server ~]$ ipa group-del examplegroup
```

### 10.10.3. 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 10.9.4, “Specifying Default User and Group Attributes”](#).

#### 10.10.3.1. Setting Search Limits

##### 10.10.3.1.1. 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 sorts and displays 20 entries per page. 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.

### 10.10.3.1.2. 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.

#### 10.10.3.1.2.1. With the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Search Options** area.

**Identity** **Policy** **IPA Server**

Role Based Access Control Self Service Permissions Delegations Entitlements **Configuration**

## CONFIGURATION

[Refresh](#) [Reset](#) [Update](#)

▼ **SEARCH OPTIONS**

Search size limit:

Search time limit:

► **USER OPTIONS**

► **GROUP OPTIONS**

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.



### Note

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.

#### 10.10.3.1.2.2. With the Command Line

The search limits can be changed using the **config-mod** command.

```
[bjensen@server ~]$ 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
```



## Note

Setting the time limit or size limit value to -1 means that there are no limits on searches.

### 10.10.3.1.3. 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
```

### 10.10.3.2. 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](#).

#### 10.10.3.2.1. 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 10.5, "Default Search Attributes"](#). All search attributes are searched in a user/group search.

**Table 10.5. Default Search Attributes**

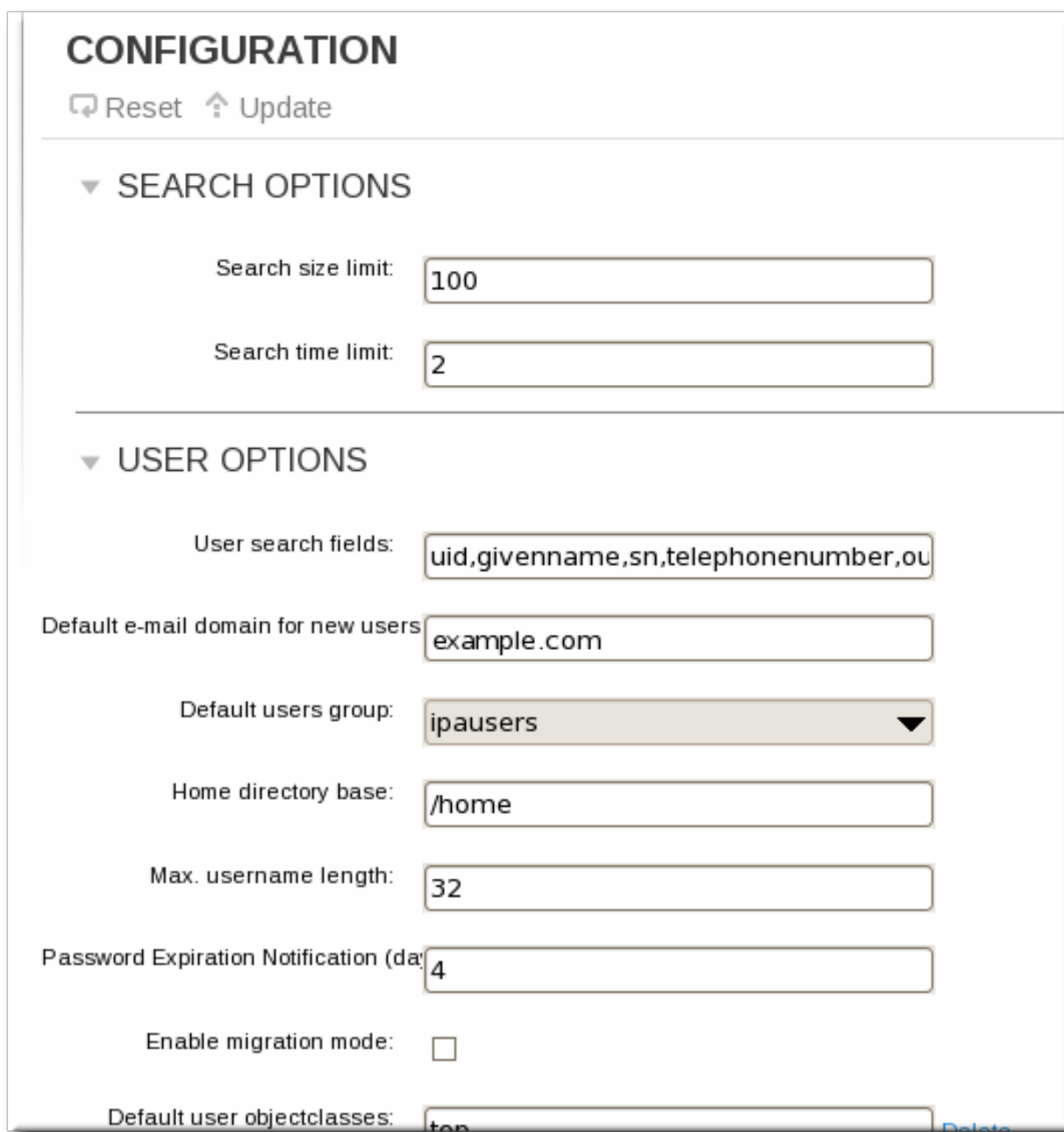
User Search Attributes	
First name	Last name
Login ID	Job title
Organizational unit	Phone number
Group Search Attributes	
Name	Description

The attributes which are searched in user and group searches can be changed, as described in [Section 10.10.3.2, “Setting Search Attributes”](#) and [Section 10.10.3.2.3, “Changing Group Search Attributes”](#).

### 10.10.3.2.2. Changing User Search Attributes

#### 10.10.3.2.2.1. From the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **User Options** area.



The screenshot shows the 'CONFIGURATION' page in the IPA web UI. At the top, there are 'Reset' and 'Update' buttons. Below this is a section titled 'SEARCH OPTIONS' with two input fields: 'Search size limit' set to '100' and 'Search time limit' set to '2'. A horizontal line separates this from the 'USER OPTIONS' section. This section contains several fields: 'User search fields' with the value 'uid,givenname,sn,telephonenumber,ou', 'Default e-mail domain for new users' with 'example.com', 'Default users group' with a dropdown menu showing 'ipausers', 'Home directory base' with '/home', 'Max. username length' with '32', 'Password Expiration Notification (days)' with '4', 'Enable migration mode' with an unchecked checkbox, and 'Default user objectclasses' with 'top'. A 'Delete' link is visible next to the 'Default user objectclasses' field.

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.

#### 10.10.3.2.2.2. From the Command Line

To change the search attributes, use the **--usersearch** option to set the attributes for user searches.

```
[bjensen@server ~]$ 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.

This can be done by specifying each attribute with a **--usersearch** argument or by listing all of the attributes in a comma-separated list inside curly braces, such as `{attr1,attr2,attr3}`. For long lists, it can be easier to use the curly braces than multiple options.

#### 10.10.3.2.3. 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](#).

##### 10.10.3.2.3.1. 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 interface. The 'Configuration' tab is selected, and the 'GROUP OPTIONS' section is expanded. The 'Group search fields' field contains 'cn,description'. The 'Default group objectclasses' section lists five objectclasses: 'top', 'groupofnames', 'nestedgroup', 'ipausergroup', and 'ipaobject', each with a 'Delete' link. An 'Add' link is at the bottom of the list.

**Identity** **Policy** **IPA Server**

Role Based Access Control Self Service Permissions Delegations Entitlements **Configuration**

## CONFIGURATION

[Refresh](#) [Reset](#) [Update](#)

► **SEARCH OPTIONS**

► **USER OPTIONS**

▼ **GROUP OPTIONS**

Group search fields:

Default group objectclasses:

<input type="text" value="top"/>	<a href="#">Delete</a>
<input type="text" value="groupofnames"/>	<a href="#">Delete</a>
<input type="text" value="nestedgroup"/>	<a href="#">Delete</a>
<input type="text" value="ipausergroup"/>	<a href="#">Delete</a>
<input type="text" value="ipaobject"/>	<a href="#">Delete</a>

[Add](#)

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.

#### 10.10.3.2.3.2. From the Command Line

To change the search attributes, use the **--groupsearch** options to set the attributes for group searches.

```
[bjensen@server ~]$ 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.

This can be done by specifying each attribute with a **--groupsearch** argument or by listing all of the attributes in a comma-separated list inside curly braces, such as `{attr1,attr2,attr3}`. For long lists, it can be easier to use the curly braces than multiple options.

**10.10.3.2.4. 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.

**10.10.3.3. Searching for Groups Based on Type**

Group definitions are simple, but because it is possible to create automember rules which automatically assign entries to groups, nested groups which include members implicitly, and groups based on member attributes such as POSIX, the reality of the group definitions can be very complex.

There are numerous different options with the **group-find** command which allow groups to be searched based on who the members are and are not and other attributes of the group definition.

For example, user private groups are never displayed in the IdM UI and are not returned in a regular search. Using the **--private** option, however, limits the search results to only private groups.

```
[root@server ~]# ipa group-find --private
-----
1 group matched
-----
Group name: jsmith
Description: User private group for jsmith
GID: 1084600001
-----
Number of entries returned 1
-----
```

Group searches can also be based on who does or does not belong to a group. This can mean single users, other groups, or even other configuration entries like roles and host-based access control definitions. For example, the first search shows what groups the user **jsmith** belongs to:

```
[root@server ~]# ipa group-find --user=jsmith
-----
1 group matched
-----
Group name: ipausers
```

```

Description: Default group for all users
Member users: jsmith
-----
Number of entries returned 1
-----

```

The other search shows all the groups that **jsmith** does *not* belong to:

```

[root@server ~]# ipa group-find --no-user=jsmith
-----
3 groups matched
-----
Group name: admins
Description: Account administrators group
GID: 1084600000
Member users: admin

Group name: editors
Description: Limited admins who can edit other users
GID: 1084600002

Group name: trust admins
Description: Trusts administrators group
Member users: admin
-----
Number of entries returned 3
-----

```

Some useful group search options are listed in [Table 10.6, “Common Group Search Options”](#).

**Table 10.6. Common Group Search Options**

Option	Criteria Description
--private	Displays only private groups.
--gid	Displays only the group which matches the complete, specified GID.
--group-name	Displays only groups with that name or part of their name.
--users, --no-users	Displays only groups which have the given users as members (or which do not include the given user).
--in-hbacrules, --not-in-hbac-rules	Displays only groups which belong to a given host-based access control rule (or which do not belong to the rule, for the <b>--not-in</b> option). There are similar options to display (or not) groups which belong to a specified sudo rule and role.
--in-groups, --not-in-groups	Displays only groups which belong to another, specified group (or which do not belong to the group, for the <b>--not-in</b> option). There are similar options to display (or not) groups which belong to a specified netgroup.

## 10.11. Issuing User Certificates with the IdM CA

Identity Management enables the administrator to issue certificates to individual users. In addition, users can request certificates for themselves when permitted by the Certificate Authority access control lists (CA ACLs).

The following procedures use IdM's certificate profiles and CA ACLs, which are described separately in [Section 26.9, “Certificate Profiles”](#) and [Section 26.10, “Certificate Authority ACL Rules”](#). For more details about using certificate profiles and CA ACLs, see these sections.

### Issuing Certificates to Users from the Command Line

1. Create or import a new custom certificate profile for handling requests for user certificates. For example:

```
$ ipa certprofile-import certificate_profile --
file=certificate_profile.cfg --store=True
```

2. Add a new Certificate Authority (CA) ACL that will be used to permit requesting certificates for user entries. For example:

```
$ ipa caacl-add users_certificate_profile --usercat=all
```

3. Add the custom certificate profile to the CA ACL.

```
$ ipa caacl-add-profile users_certificate_profile --
certprofiles=certificate_profile
```

4. Generate a certificate request for the user. For example, using OpenSSL:

```
$ openssl req -new -newkey rsa:2048 -days 365 -nodes -keyout
private.key -out cert.csr -subj '/CN=user'
```

5. Run the **ipa cert-request** command to have the IdM CA issue a new certificate for the user.

```
$ ipa cert-request cert.csr --principal=user --profile-
id=certificate_profile
```

To make sure the newly-issued certificate is assigned to the user, you can use the **ipa user-show** command:

```
$ ipa user-show user
User login: user
...
Certificate: MIICfzCCAWcCAQA...
...
```

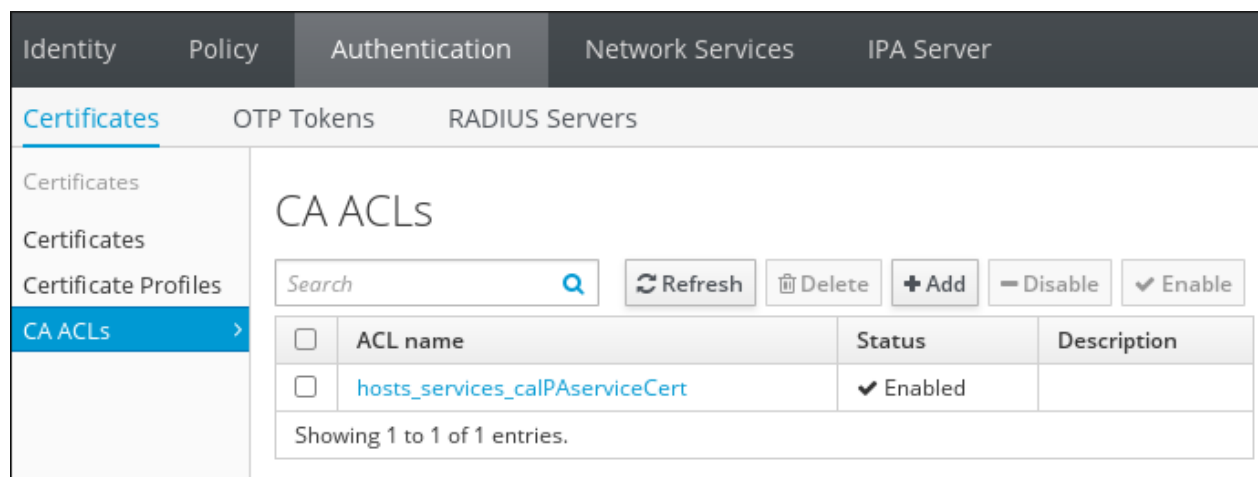
### Issuing Certificates to Users in the Web UI

1. Create or import a new custom certificate profile for handling requests for user certificates. Importing profiles is only possible from the command line, for example:

```
$ ipa certprofile-import certificate_profile --  
file=certificate_profile.txt --store=True
```

For information about certificate profiles, see [Section 26.9, “Certificate Profiles”](#).

2. In the web UI, under the **Authentication** tab, open the **CA ACLs** section.



**Figure 10.5. CA ACL Rules Management in the Web UI**

Click **Add** at the top of the list of Certificate Authority (CA) ACLs to add a new CA ACL that permits requesting certificates for user entries.

- a. In the **Add CA ACL** window that opens, fill in the required information about the new CA ACL.

The 'Add CA ACL' dialog box has a title bar with a close button (X). It contains two input fields:

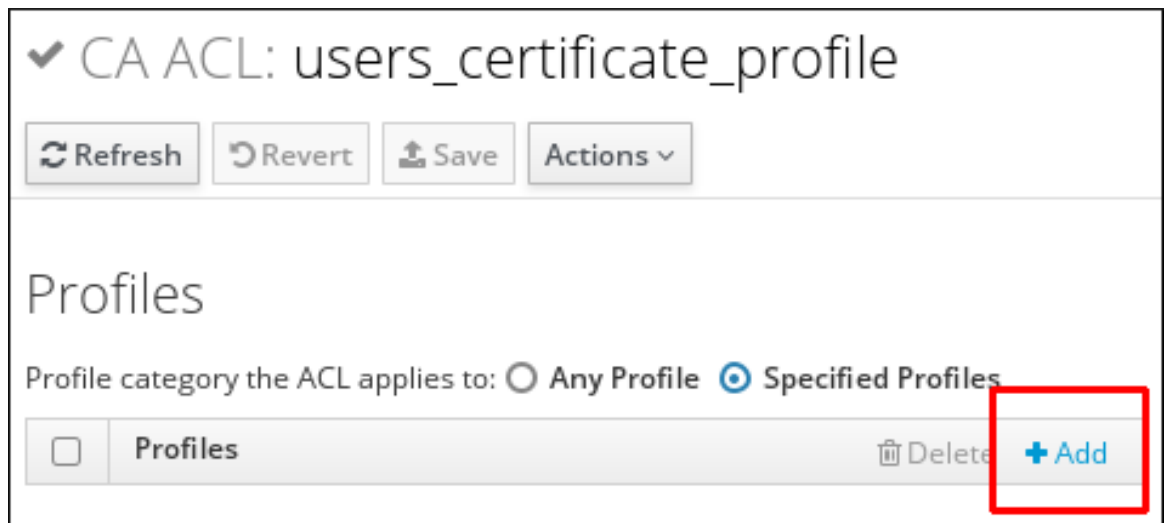
- ACL name \***: A text box containing 'users\_certificate\_profile'.
- Description**: A text area containing 'CA ACL for certificate\_profile'.

Below the input fields, there is a note: '\* Required field'. At the bottom of the dialog, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

**Figure 10.6. Adding a New CA ACL**

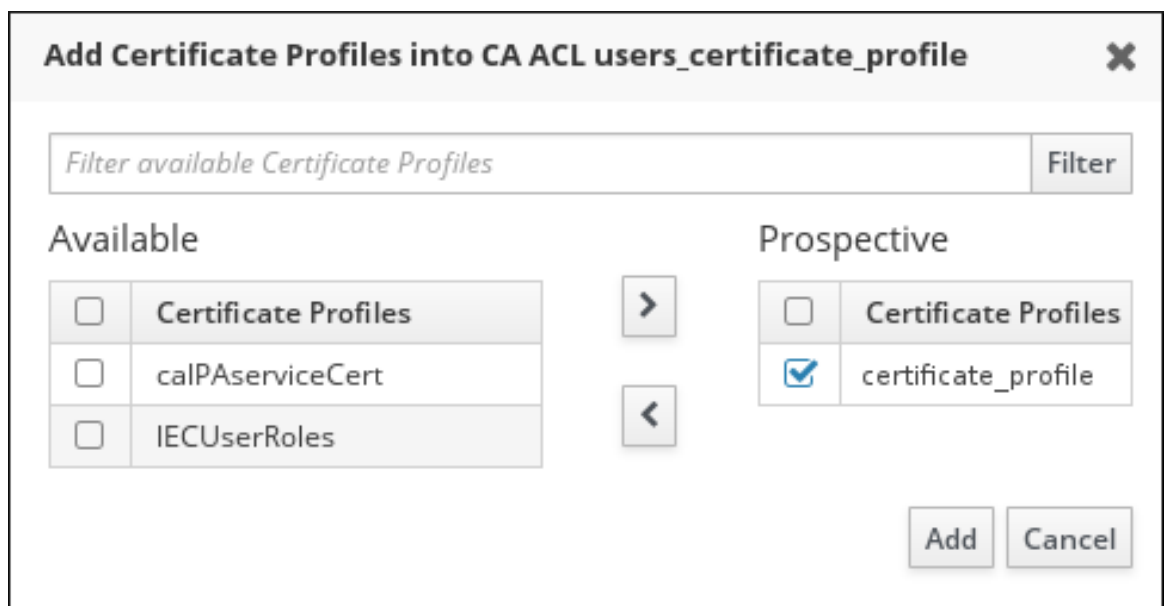
Then, click **Add and Edit** to go directly to the CA ACL configuration page.

- b. In the CA ACL configuration page, scroll to the **Profiles** section and click **Add** at the top of the profiles list.



**Figure 10.7. Adding a Certificate Profile to the CA ACL**

- c. Add the custom certificate profile to the CA ACL by selecting the profile and moving it to the **Prospective** column.

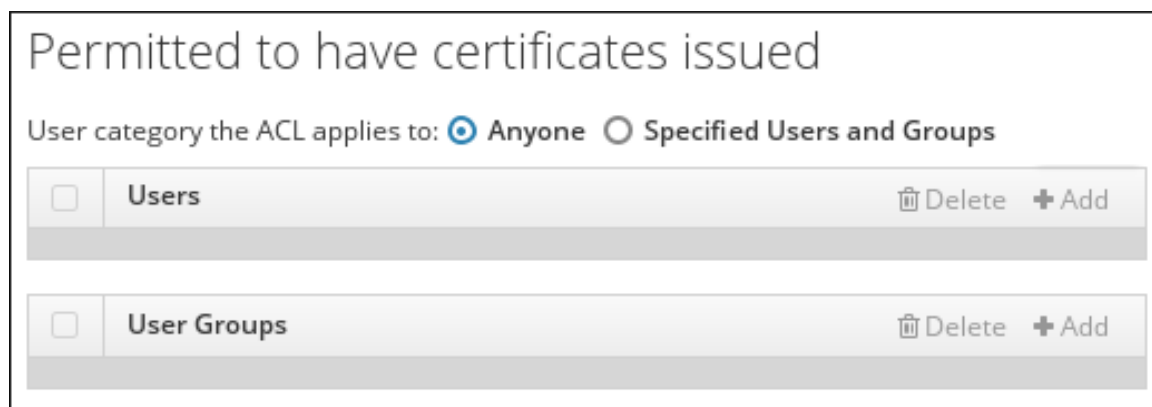


**Figure 10.8. Selecting a Certificate Profile**

Then, click **Add**.

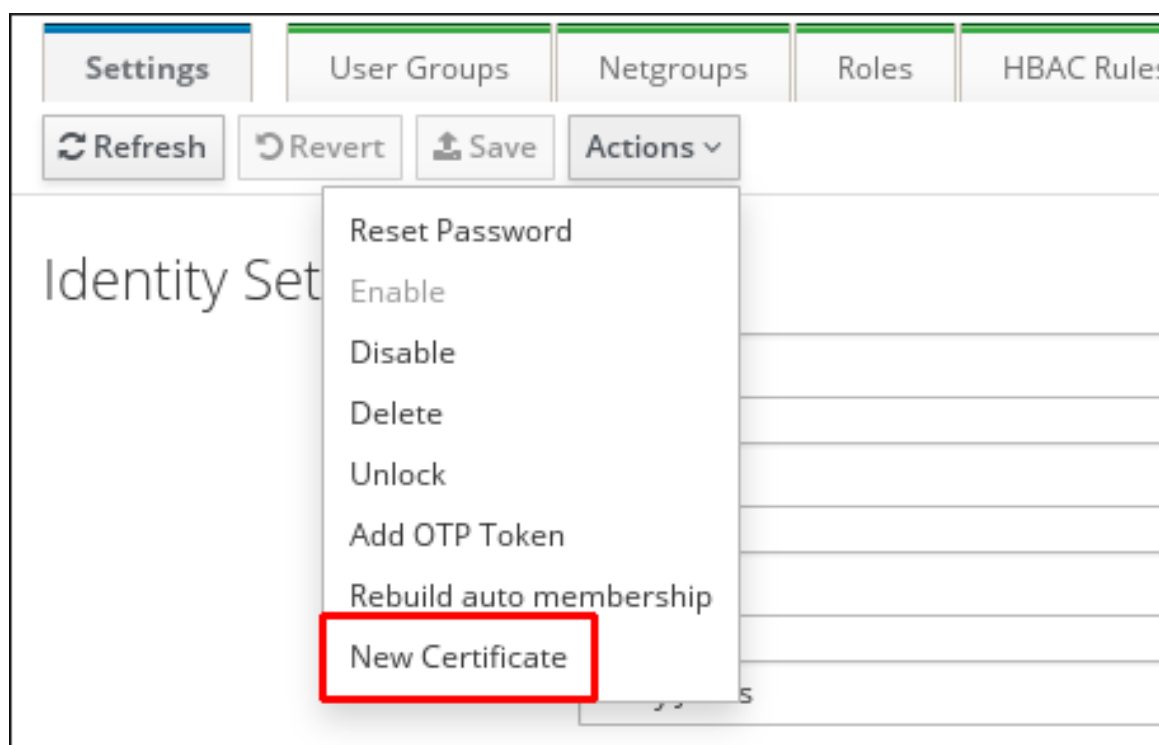
- d. Scroll to the **Permitted to have certificates issued** section to associate the CA ACL with users or user groups.

You can either add users or groups using the **Add** buttons, or select the **Anyone** option to associate the CA ACL with all users.



**Figure 10.9. Adding Users to the CA ACL**

- e. At the top of the CA ACL configuration page, click **Save** to confirm the changes to the CA ACL.
3. Request a new certificate for the user.
  - a. Under the **Identity** tab and the **Users** subtab, choose the user for whom the certificate will be requested. Click on the user's user name to open the user entry configuration page.
  - b. Click **Actions** at the top of the user configuration page, and then click **New Certificate**.



**Figure 10.10. Requesting a Certificate for a User**

- c. Fill in the required information.

**Issue New Certificate for User user**

Profile ID: certificate\_profile

1. Create a certificate database or use an existing one. To create a new database:  
`# certutil -N -d <database path>`
2. Create a CSR with subject `CN=<uid>, O=<realm>`, for example:  
`# certutil -R -d <database path> -a -g <key size> -s 'CN=alice, O=IPA.LOCAL'`
3. Copy and paste the CSR (from `-----BEGIN NEW CERTIFICATE REQUEST-----` to `-----END NEW CERTIFICATE REQUEST-----`) into the text area below:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBfDCB5gIBADAQMq4wDAYDVQQDDAVhbGljZTCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwYkCgYEA2mJhtELuvf/gY8vsha9dQKtZmM+irSotMMRz3CiSRdQSsvxJe0ID
QtPraU8z0gzBImwZY0ZGukxKvJPGG4ErjHtcIRlb5V+XEuMr1R+TmclCXqGId7FP
l+IGIIvrtw2vuHp1EHoPQpsoVUzjEp8ql5Kr9lMmUTKI9QF/4EUw2VECAwEAAAt
-----END CERTIFICATE REQUEST-----
```

Issue Cancel

**Figure 10.11. Issuing a Certificate for a User**

Then, click **Issue**.

After this, the newly issued certificate is visible in the user configuration page.

## 10.12. Managing User Certificates

In Identity Management, the administrator can add certificates issued by CAs other than the IdM CA to a user entry, as well as remove certificates from user entries. This allows the users to authenticate using smart cards: certificates issued by the smart card vendor can be added to IdM.

Note that users in IdM can have multiple certificates assigned.

### Managing User Certificates from the Command Line

To add or remove user certificates from the command line, use the following two commands:

**ipa user-add-cert**

Adds one or more certificates to a specified user entry.

**ipa user-remove-cert**

Removes one or more certificates from a specified user entry.

The commands require you to specify the following information:

- ✱ the name of the user to which the certificate is to be added or from which it is to be removed

- ✱ the Base64-encoded DER certificate to be added or removed

You can pass the user entry and the certificate directly with the command, for example:

```
$ ipa user-add-cert user --certificate=MIQTPrajQAwg...
```

If you run the commands without specifying these attributes, IdM automatically prompts you for them.

To display the certificates assigned to a user entry, use the **ipa user-show** command:

```
$ ipa user-show user
User login: user
...
Certificate: MIICfzCCAWcCAQA...
...
```

You can also save user's certificate or certificates to a file. To do this, specify the file to which to export the certificates by adding the **--out** option to **ipa user-show**. For example:

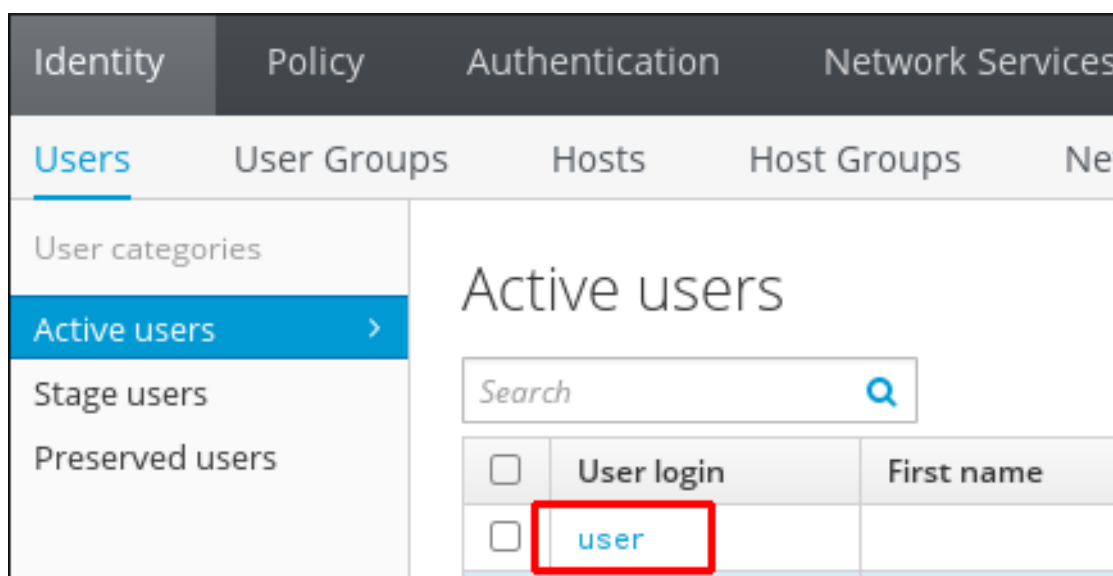
```
$ ipa user-show user --out=file_name
```

If the user has more than one certificate, the **--out** option exports all of them. The certificate or certificates are exported as PEM objects.

## User Certificates in the Web UI

The IdM web UI currently does not support adding or removing user certificates. However, it is possible to display the certificates assigned to a user entry:

1. Under the **Identity** tab, open the **Users** subtab.
2. Click on the user name to open the user entry configuration page.



**Figure 10.12. Opening the User Entry Configuration Page**



3. Scroll to the **Certificate** section to view the certificate assigned to the user entry.

Login shell	<input type="text" value="/bin/sh"/>
Home directory	<input type="text" value="/home/user"/>
SSH public keys	<input type="button" value="Add"/>
Certificate	MIIDrTCCApWgAwIBAgIBJzANBgkqhkiG9w0BAQsFADBBMR8wHQYDVQ QKDBZJUUEuTE9DQUwgMjAxNTA5MjlyMDQ3MR4wHAYDVQQDDDBVDZX J0aWZpY2F0ZSBBdXR0b3JpdHkwHhcNMTUxMTAzMDU1NTE1WhcNMTc xMTAzMDQ1NTE1WjAxMR8wHQYDVQQKDBZJUUEuTE9DQUwgMjAxNT A5MjlyMDQ3MQ4wDAYDVQQDDAVhbGljZTCBnzANBgkqhkiG9w0BAQEF AAOBjQAwgYkCgYEA2mJhtELuvf/gY8vsha9dQKtZmM+irSotMMRz3CiSRd

**Figure 10.13. Displaying the User Certificate in the Web UI**

To add or remove user certificates, use the **ipa user-add-cert** and **ipa user-remove-cert** commands, as described in [Section 10.12, “Managing User Certificates from the Command Line”](#).

[2] The key type is determined automatically from the key itself, if it is not included in the uploaded key.

[3] See [Section 10.8, “Managing Unique UID and GID Number Assignments”](#) for information on changing GID/UID assignment ranges.

## **Part III. Managing System Identities in a Linux Domain**

## Chapter 11. 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)

### 11.1. 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 11.5, “Managing Public SSH Keys for Hosts”](#).
- 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 18, 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*.

## 11.2. About Host Entry Configuration Properties

A host entry can contain information about the host that is outside its system configuration, such as its physical location, its MAC address, and keys and certificates.

This information can be set when the host entry is created if it is created manually; otherwise, most of that information needs to be added to the host entry after the host is enrolled in the domain.

**Table 11.1. Host Configuration Properties**

UI Field	Command-Line Option	Description
Description	<code>--desc=<i>description</i></code>	A description of the host.
Locality	<code>--locality=<i>locality</i></code>	The geographic location of the host.
Location	<code>--location=<i>location</i></code>	The physical location of the host, such as its data center rack.

UI Field	Command-Line Option	Description
Platform	<code>--platform=<i>string</i></code>	The host hardware or architecture.
Operating system	<code>--os=<i>string</i></code>	The operating system and version for the host.
MAC address	<code>--macaddress=<i>address</i></code>	The MAC address for the host. This is a multi-valued attribute. The MAC address is used by the NIS plug-in to create a NIS ethers map for the host.
SSH public keys	<code>--sshpubkey=<i>string</i></code>	The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.
Principal name (not editable)	<code>--principalname=<i>principal</i></code>	The Kerberos principal name for the host. This defaults to the hostname during the client installation, unless a different principal is explicitly set in the <b>-p</b> . This can be changed using the command-line tools, but cannot be changed in the UI.
Set One-Time Password	<code>--password=<i>string</i></code>	Sets a password for the host which can be used in bulk enrollment.
-	<code>--random</code>	Generates a random password to be used in bulk enrollment.
-	<code>--certificate=<i>string</i></code>	A certificate blob for the host.
-	<code>--updatedns</code>	This sets whether the host can dynamically update its DNS entries if its IP address changes.

## 11.3. 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.

### 11.3.1. 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.

### 11.3.2. 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.

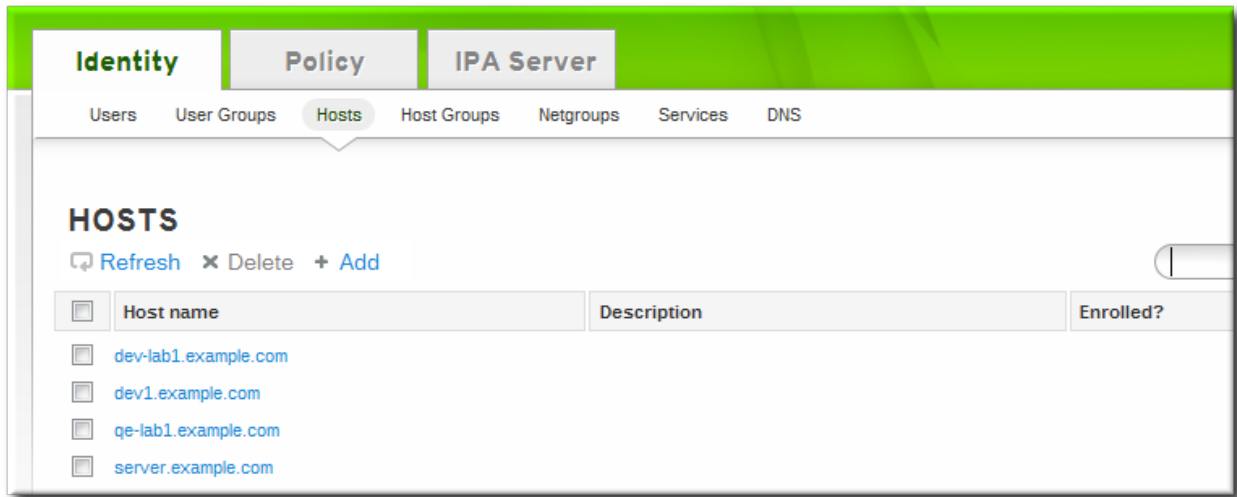
## 11.4. Creating Certificates for Hosts

By default, the IdM server has an integrated certificate authority. This CA can be used to create, revoke, and issue certificates for hosts in the IdM domain.

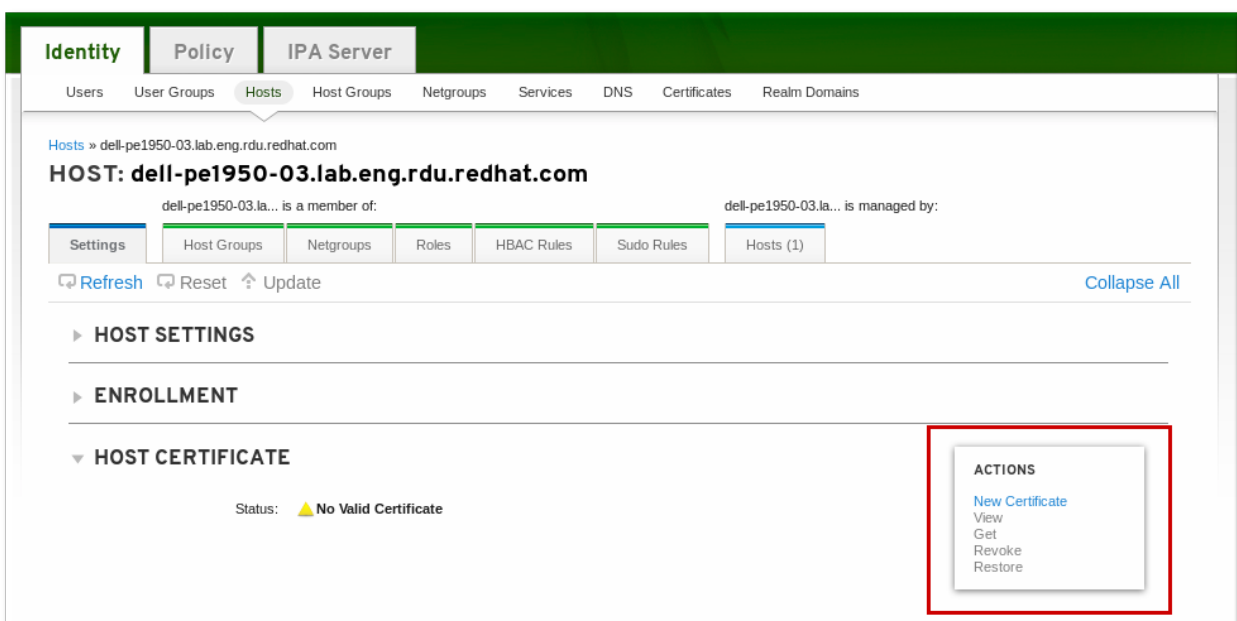
### 11.4.1. Showing Certificates

#### 11.4.1.1. In the Host Entry in the UI

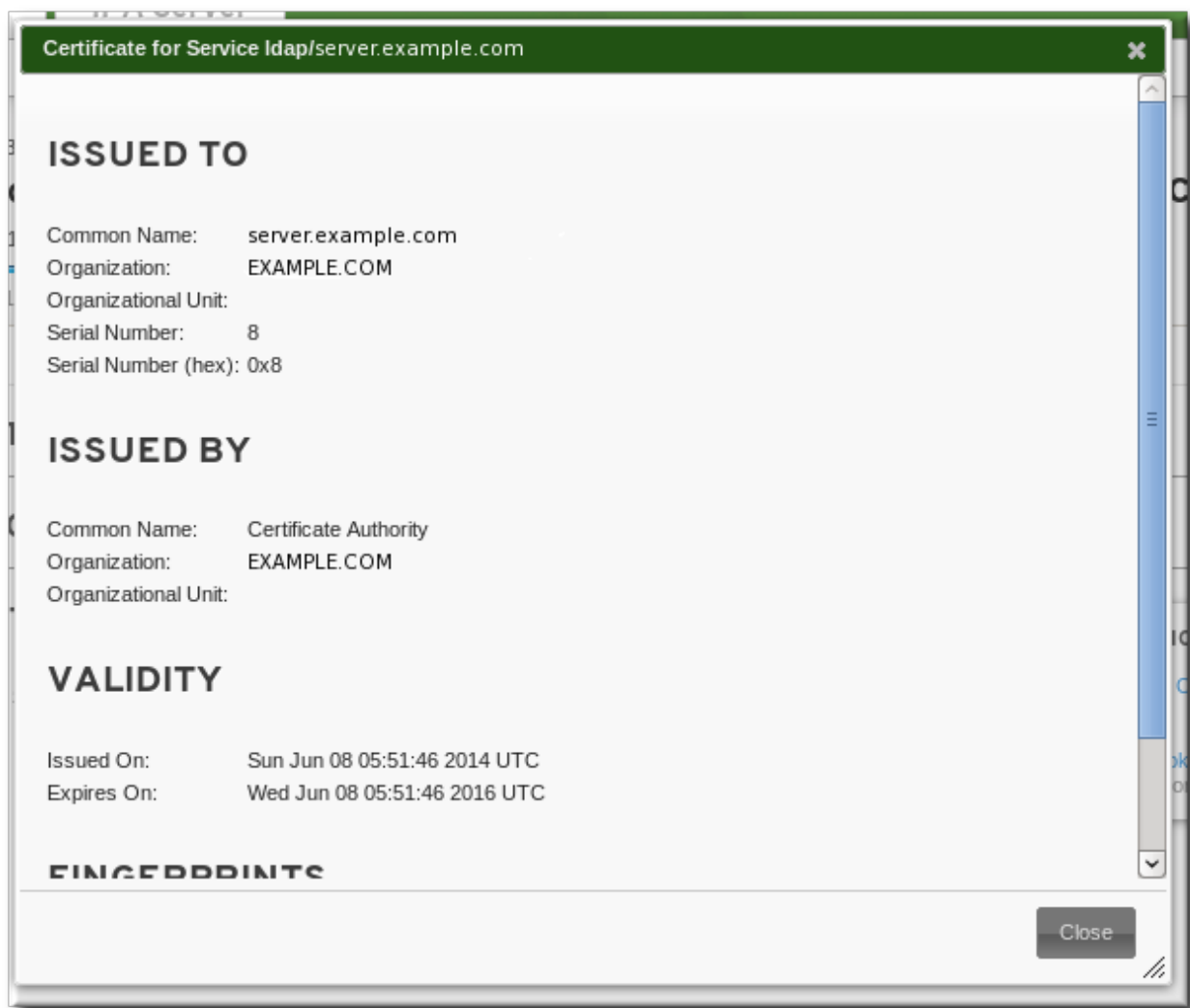
1. Open the **Identity** tab, and select the **Hosts** subtab.
2. Click the name of the host.



3. In the **Settings** tab, scroll to the **Host Certificate** tab at the bottom.

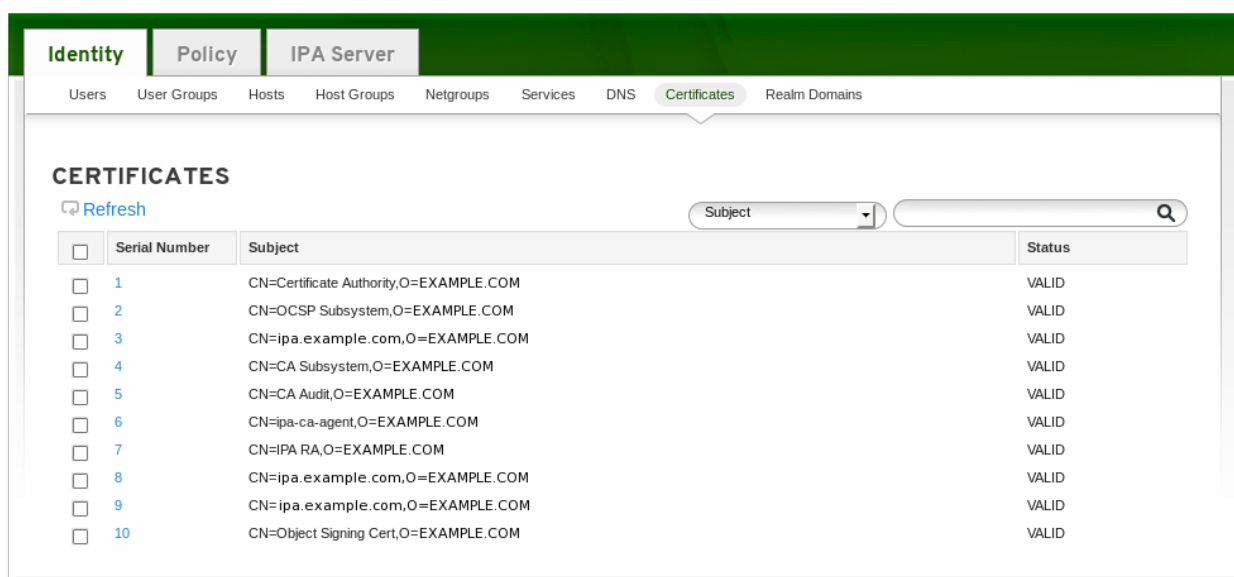


4. If a certificate has been issued, click the **View** link to display the details about the certificate. To retrieve the full certificate, click the **Get** link.



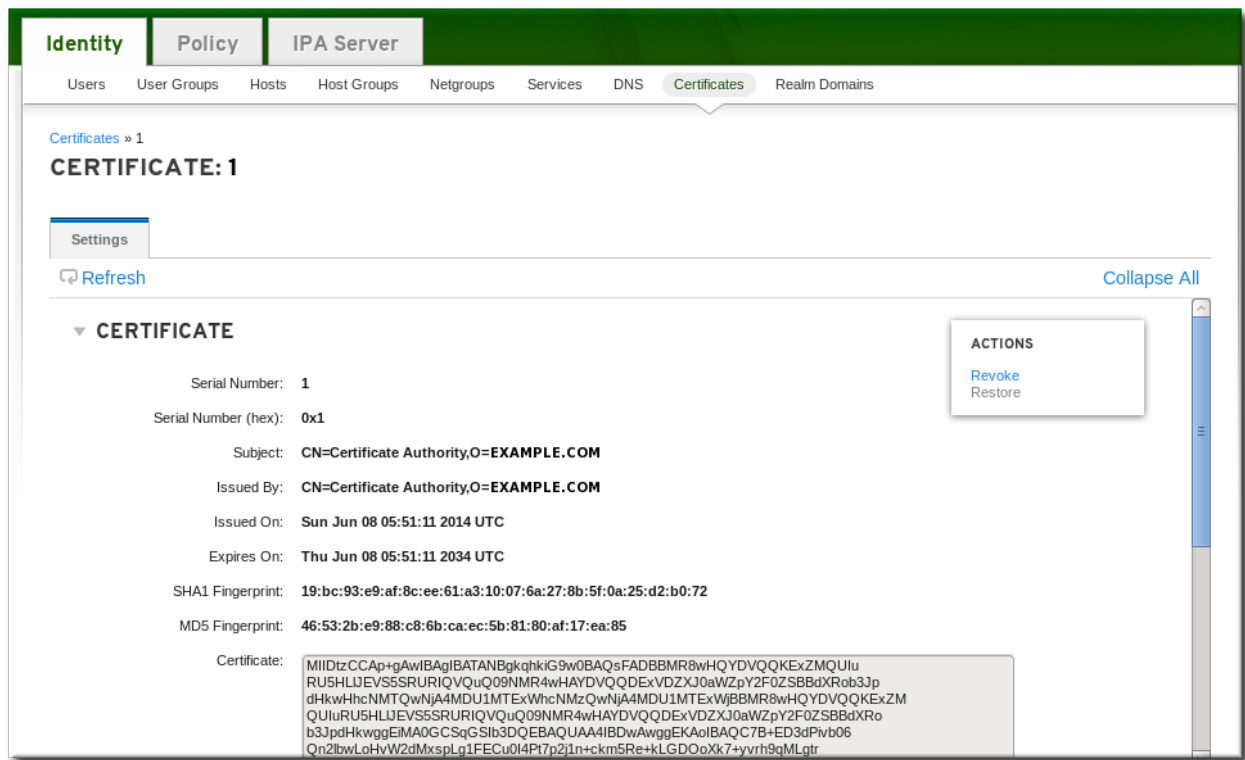
#### 11.4.1.2. In the Certificate List in the UI

1. Open the **Identity** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to view.



3. The top of the certificate entry shows the details of the certificate, such as its CN. The full certificate blob is available at the bottom of the page.





### 11.4.1.3. In the Command Line

All of the certificates which have been issued by the IdM CA are listed with the **ipa cert-find** command.

```
[root@server ~]# kinit admin
[root@server ~]# ipa cert-find
-----
10 certificates matched
-----
Serial number (hex): 0x1
Serial number: 1
Status: VALID
Subject: CN=Certificate Authority,O=EXAMPLE.COM
...
-----
Number of entries returned 10
-----
```

With a large number of certificates, it can be easier to search for a specific certificate by serial number or by an issue date. To search by a serial number, simply include it with the **cert-show** command.

```
[root@server ~]# ipa cert-show 132
Serial number: 132
Certificate:
MIIDtzCCAp+gAwIBAgIBATANBgkqhkiG9w0BAQsFADBBMR8wHQYDVQQKEZXMQUIu
...
LxIQjrEFtJmoBGB/TWrlwGEWylayr4iTEflayZ+RGNylLalEAtk9RLjEjg==
Subject: CN=Certificate Authority,O=EXAMPLE.COM
Issuer: CN=Certificate Authority,O=EXAMPLE.COM
Not Before: Sun Jun 08 05:51:11 2014 UTC
Not After: Thu Jun 08 05:51:11 2034 UTC
```

```
Fingerprint (MD5): 46:53:2b:e9:88:c8:6b:ca:ec:5b:81:80:af:17:ea:85
Fingerprint (SHA1):
19:bc:93:e9:af:8c:ee:61:a3:10:07:6a:27:8b:5f:0a:25:d2:b0:72
Serial number (hex): 0x132
Serial number: 132
```

The **--issuedon-from** and **--issuedon-to** options can set start/end points or a period of time to use to search for certificates.

```
ipa cert-find --issuedon-from=2013-02-01 --issuedon-to=2015-02-07
```

### 11.4.2. Revoking and Restoring Certificates

Every certificate has a specified expiration date, but there can be times when it is necessary to terminate (revoke) a certificate before that expiration. Revoking a certificate makes it invalid, so the host cannot use it for authentication.

When a certificate is revoked, there has to be a reason given. There are several different reasons — it was compromised, the entity has changed, the host is being pulled from service, or it has been replaced by a different certificate. The possible reasons are listed in [Table 11.2, “Revocation Reasons”](#).

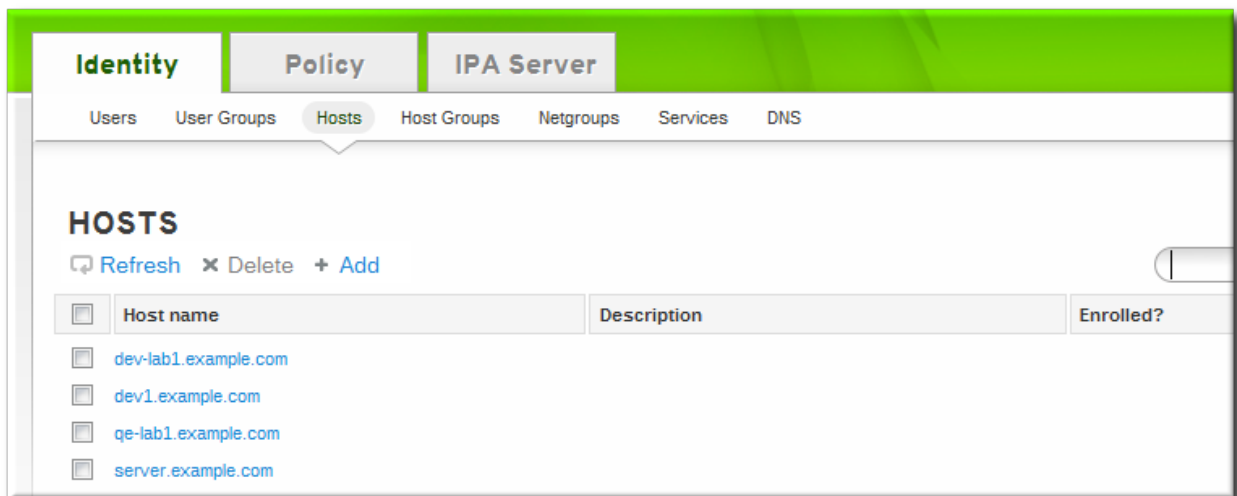
**Table 11.2. Revocation Reasons**

ID	Reason	Description
0	Unspecified	
1	Key Compromised	The underlying key was compromised. This could mean a token was lost or file was improperly accessed.
2	CA Compromised	The CA which issued the certificate was compromised.
3	Affiliation Changed	The person or host to which the certificate was issued is changing affiliations. This could mean that the person has left the company (or the host is being retired) or that it has moved departments, if the affiliation is tied to an organizational structure.
4	Superseded	The certificate has been replaced by a newer certificate.
5	Cessation of Operation	The host is being decommissioned.
6	Certificate Hold	The certificate is temporarily revoked. This is the only revocation reason that allows the certificate to be restored.

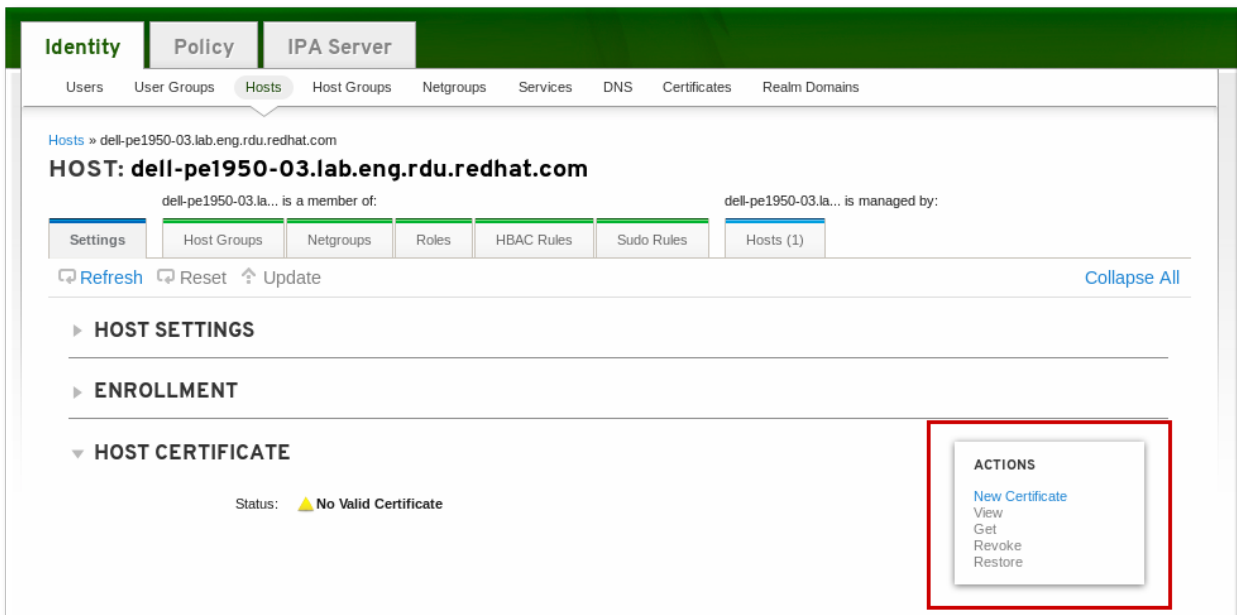
ID	Reason	Description
8	Remove from CRL	The certificate is not included in the certificate revocation list.
9	Privilege Withdrawn	The host should no longer be issued the certificate.
10	A Authority Compromise	The AA was compromised.

### 11.4.2.1. In the Host Entry in the UI

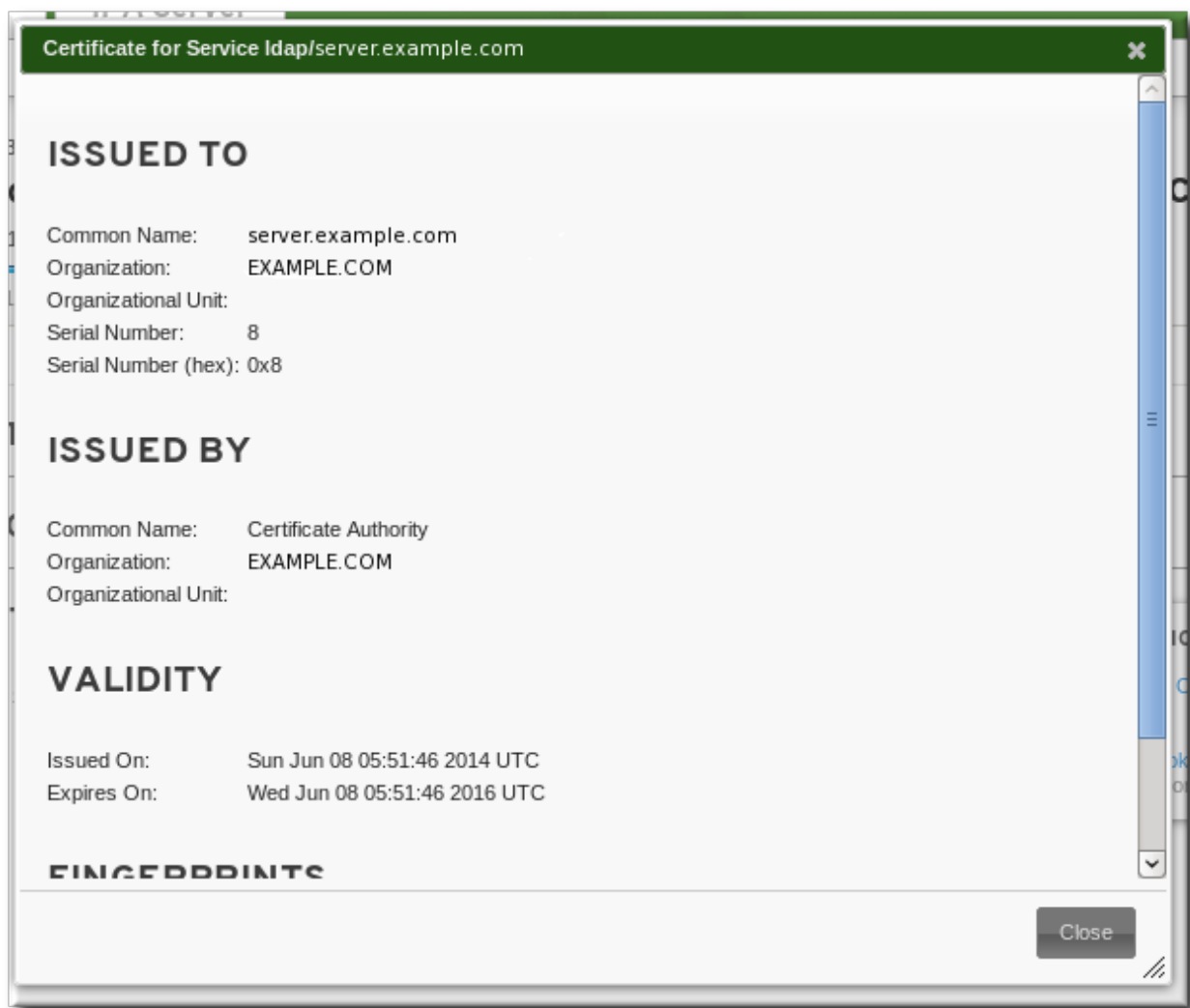
1. Open the **Identity** tab, and select the **Hosts** subtab.
2. Click the name of the host.



3. In the **Settings** tab, scroll to the **Host Certificate** tab at the bottom.



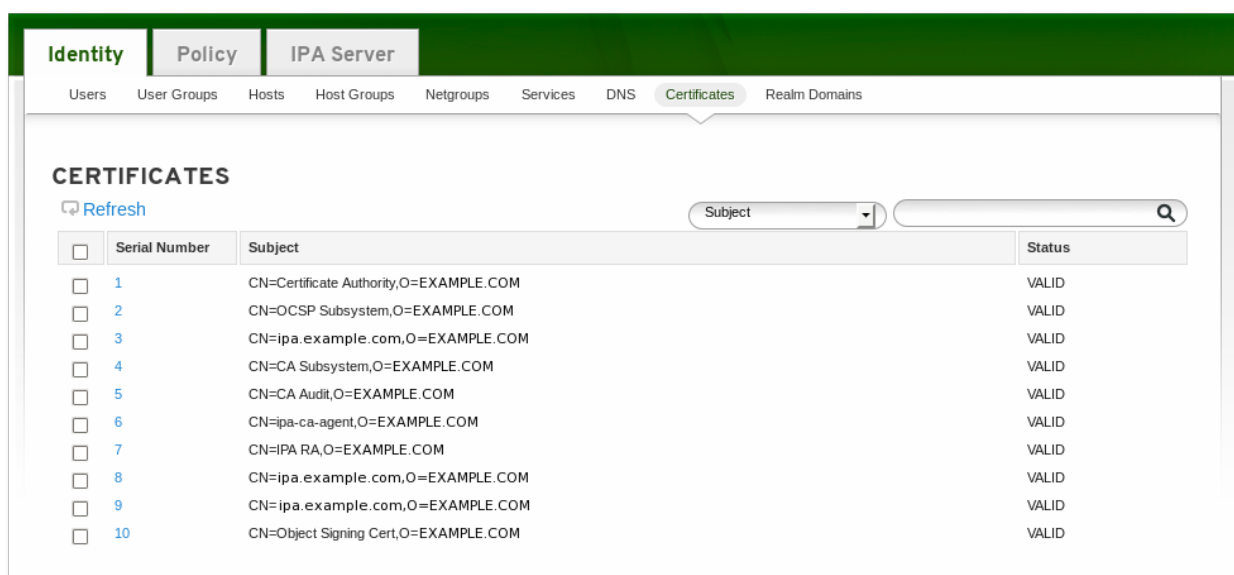
4. In the **Actions** area, click the **Revoke** link.
5. Select the reason for the revocation from the drop-down menu, and click the **Revoke** link. [Table 11.2, “Revocation Reasons”](#) describes the different options for revoking a certificate.



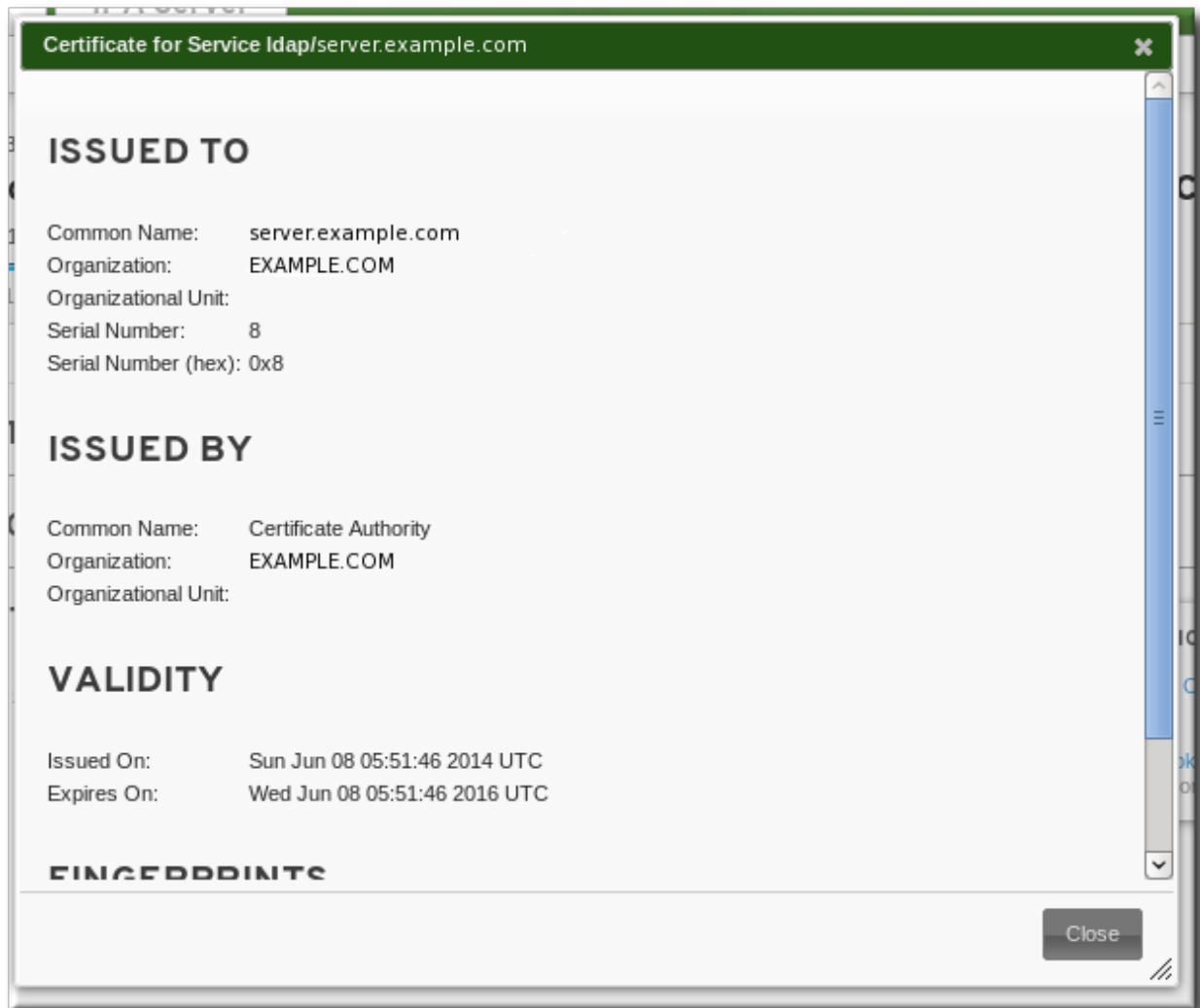
If the reason for the revocation is a certificate hold, then the certificate can be restored later by clicking the **Restore** link in the certificate actions menu.

#### 11.4.2.2. In the Certificate List in the UI

1. Open the **Identity** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to view.



3. In the **Actions** area, click the **Revoke** link.
4. Select the reason for the revocation from the drop-down menu, and click the **Revoke** link. [Table 11.2, “Revocation Reasons”](#) describes the different options for revoking a certificate.



If the reason for the revocation is a certificate hold, then the certificate can be restored later by clicking the **Restore** link in the certificate actions menu.

### 11.4.2.3. In the Command Line

To revoke a certificate from the command line, specify the certificate serial number and give the reason for the revocation in the **--revocation-reason** option.

```
[root@server ~]# kinit admin
[root@server ~]# ipa cert-revoke --revocation-reason=6 1032
```

If the reason for the revocation is a certificate hold (6), then the certificate can be restored with the **cert-remove-hold** command.

```
[root@server ~]# ipa cert-remove-hold 1032
```

### 11.4.3. Requesting New Host Certificates

The certificate request must be generated with a third-party tool such as **certutil**. The resulting certificate request can be submitted through the IdM web UI or command-line tools.

The host must already exist for a certificate to be requested. A certificate cannot be requested for a new host before it is created.

### 11.4.3.1. In the UI

1. Generate a certificate request for the host. For example:

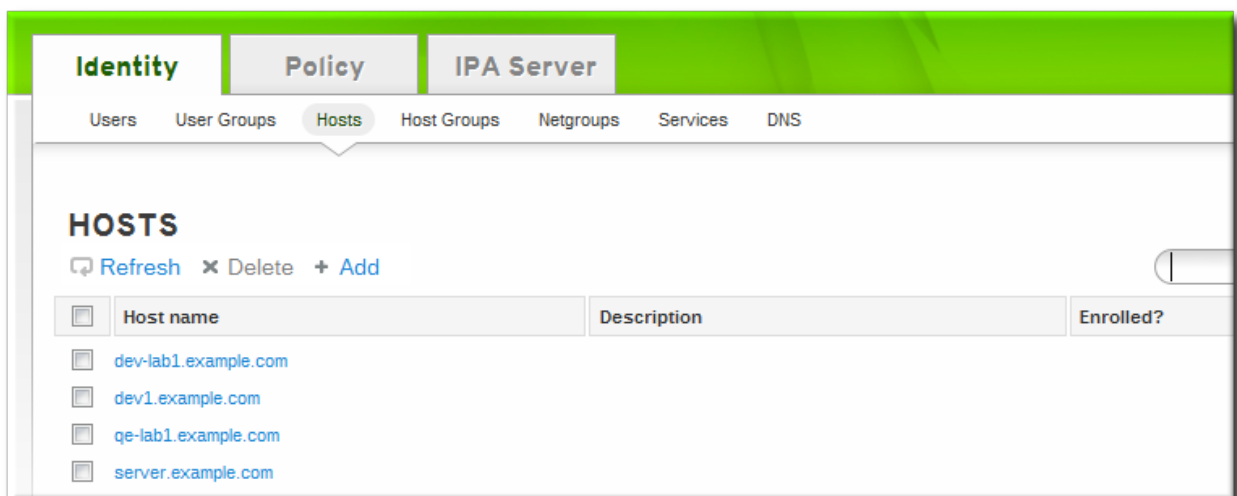
First, create a set of certificate databases that can be used to create and store the certificate locally.

```
[root@server ~]# certutil -N -d ~/test-certs/
```

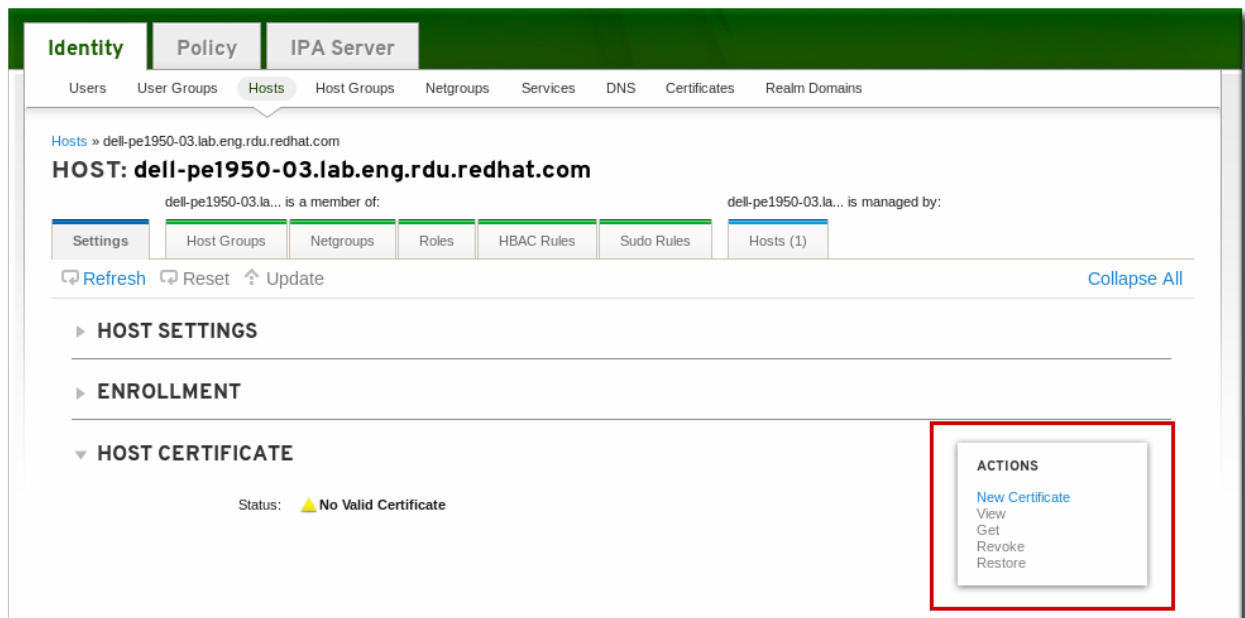
Then, create the certificate request.

```
[root@server ~]# certutil -R -d ~/test-certs -R -a -g 256 -s  
"CN=server.example.com, O=EXAMPLE.COM" -o ~/test-certs/host.csr
```

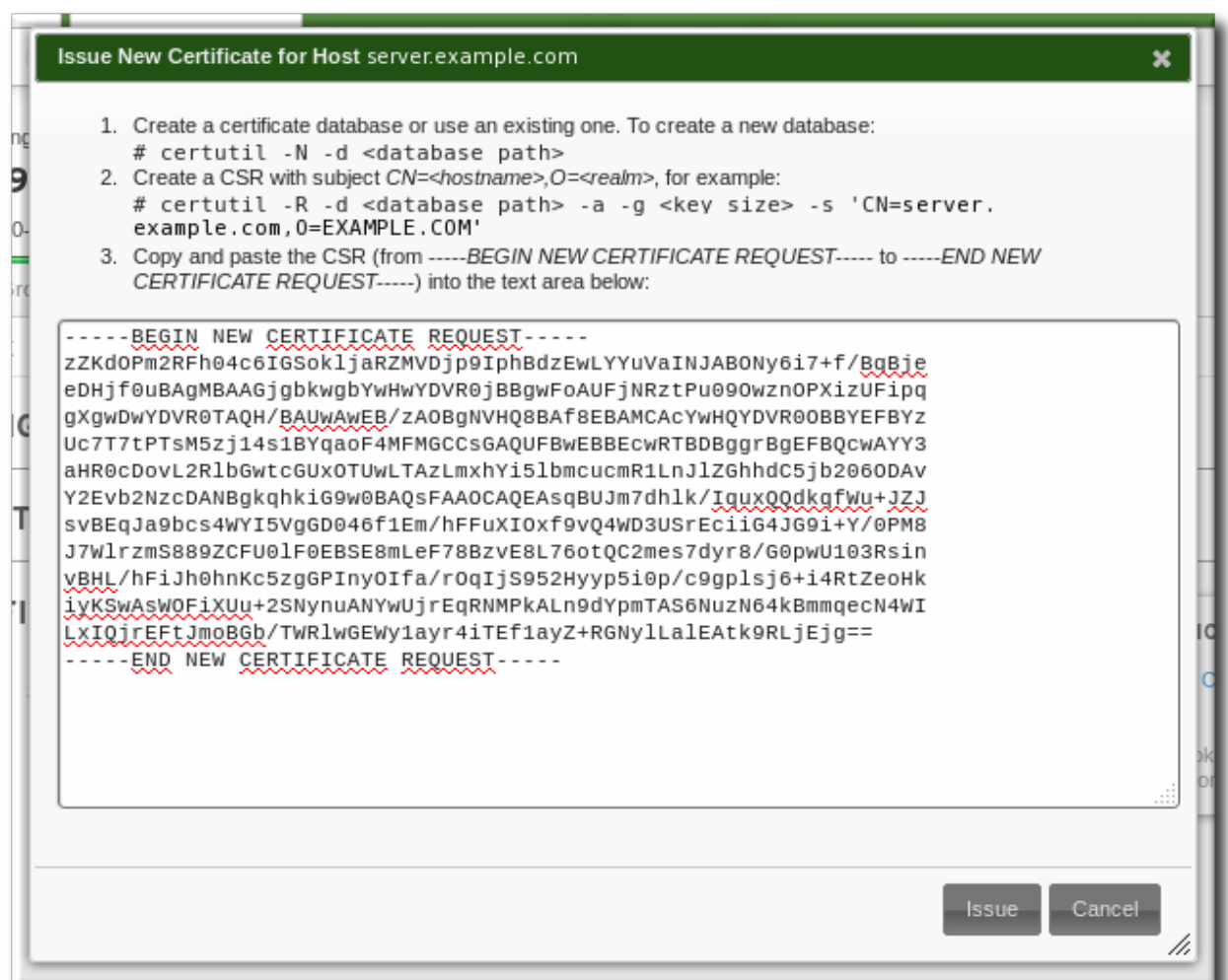
2. Copy the text of the new certificate request.
3. Open the **Identity** tab, and select the **Hosts** subtab.
4. Click the name of the host.



5. In the **Settings** tab, scroll to the **Host Certificate** tab at the bottom.



6. In the **Actions** area, click the **Request** link.
7. Paste in the body of the certificate request, including the **BEGIN NEW CERTIFICATE REQUEST** and **END NEW CERTIFICATE REQUEST** lines.



8. Click the **Issue** button.

### 11.4.3.2. In the Command Line

1. Generate a certificate request for the host. For example:

First, create a set of certificate databases that can be used to create and store the certificate locally.

```
[root@server ~]# certutil -N -d ~/test-certs/
```

Then, create the certificate request.

```
[root@server ~]# certutil -R -d ~/test-certs -R -a -g 256 -s  
"CN=server.example.com,O=EXAMPLE.COM" -o ~/test-certs/host.csr
```

2. Submit the PEM file of the certificate request to the IdM server. Along with the request itself, specify the Kerberos principal to create and associate with the newly-issued certificate.

```
[root@server ~]# ipa cert-request --  
principal=host/server.example.com host.csr
```

Note that you can use the **--profile-id** option with the **ipa cert-request** command to select a custom certificate profile to be used for the certificate. By default, IdM uses the **caIPAServiceCert** profile. For more information about certificate profiles, see [Section 26.9, "Certificate Profiles"](#).

## 11.5. Managing Public SSH Keys for Hosts

OpenSSH uses *public keys* 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.

### 11.5.1. 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.

### 11.5.2. 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.

If the SSH service is enabled with the client installation (which is the default), then an RSA key is created when the **ssh** service is first started.



#### Note

When the machine is added as an IdM client using **ipa-client-install**, the client is created with two SSH keys, RSA and DSS.

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 key fingerprints 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.

### 11.5.3. 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
AAAAB3NzaC1yc2EAAAABIwAAAQEApxjBvSFskTU0WQW4e0weeo0DZZ08F9Ud21xLLy
6F0hzwpXFGIyxvXZ52+siHBHbbqGL5+14N7UvElruyslIHx9LYUR/pPKSMXCGyboLy
5aTNl50Q5EHwrhVnFDIKXkvp45945R7SKYCUtRumm0Iw6wq0XD4o+ILeVbV3wmcB1b
Xs36ZvC/M6riefn9PcJmh6vNCvIsbMY6S+FhkwUTTi0XJjUDYRLlwM273FfWhzHK+S
SQXeBp/zInlgFvJhSZMRi9HZpDoqxLbBB9QIdIw6U4MIjNmKsSI/ASpkFm2GuQ7ZK9
KuMiTY2AoCuIRmRADf8iYNHBTXNfFurGogXwRDjQ==
```

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 |
|           . . . . |
+-----+

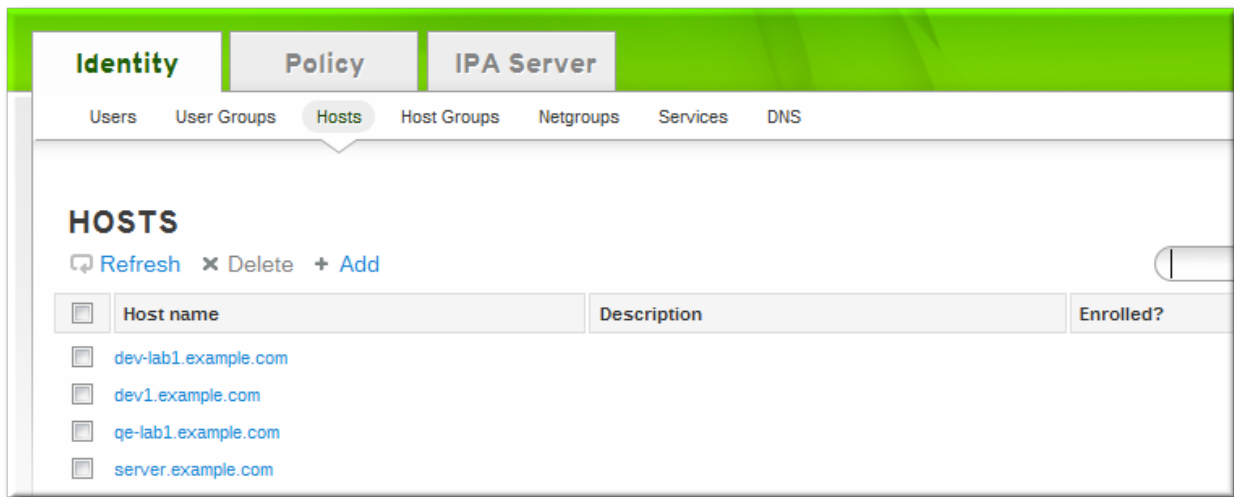
```

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]*

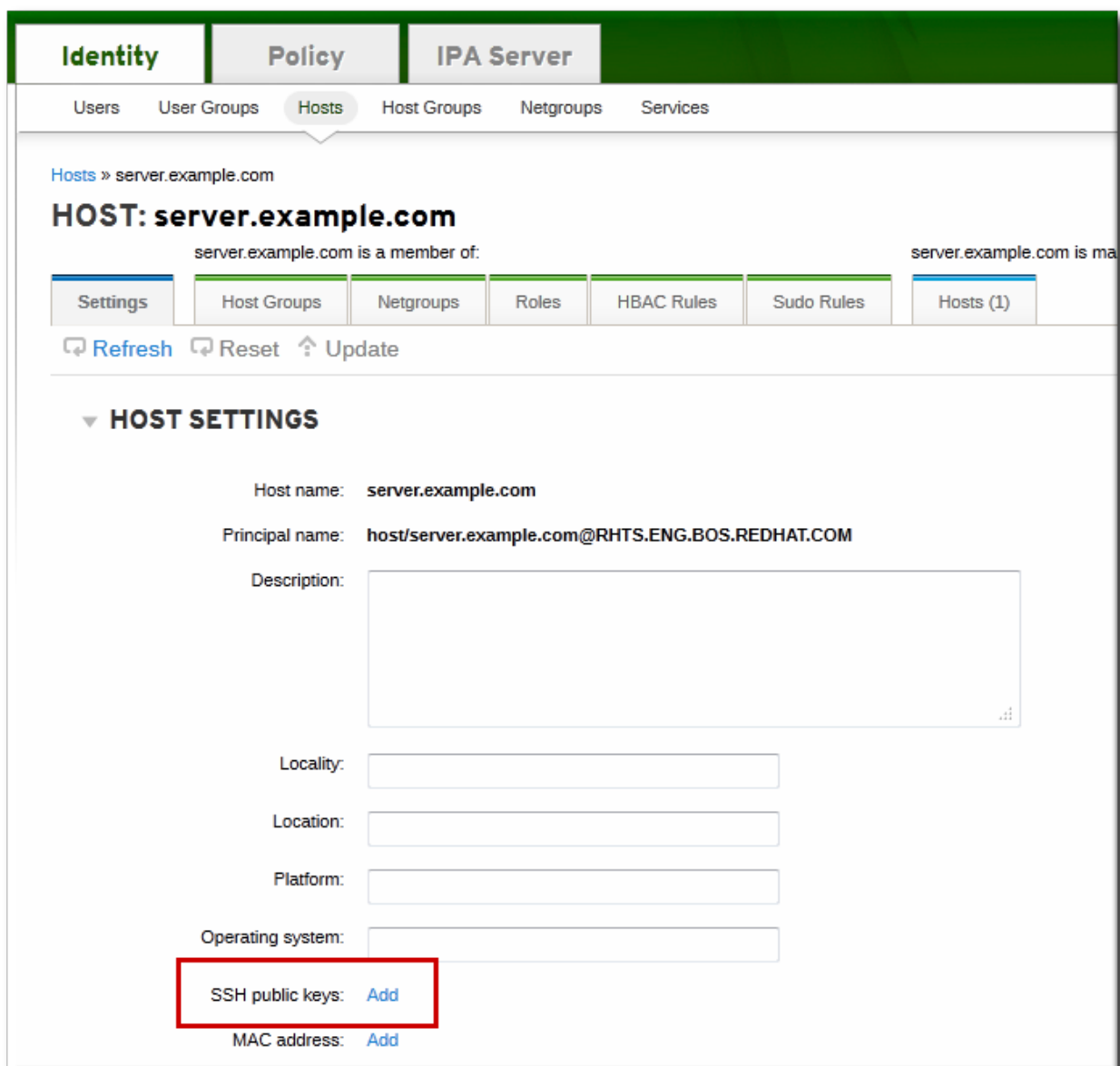
```
[jsmith@server ~]$ cat /home/jsmith/.ssh/host_keys.pub

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.



6. 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.

Netcrunch Roles HRAC Rules Sudo Rules Hosts (1)

**Set SSH key** ✕

SSH public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA1+RpdblY7UNTShs8xH2IrvF1vtse5ort4ziqay8i
9vH7+p1fyKJ6x5fZ0YAXgAbR/Q3nW4P0TQ0UsY3d5hNDCsIueIqBr461gJ97rv7FJ4jo
a
/bdLxV4ImHkLaz5PEd5JJGkJ5ukA4sugvUwzr7UOnhzma9E8H+7EiIM6JX2CqhajK0YT
2I9T9dYfRS
/VJ5dzZxkG1ZE+Syu3m4D5kJAQEjgKuaPgKYP3LSPdGT1KnSZwOolfav+buFMmwd6Smr
ThFGhz7/0F/HX5sjhk2kFOr5cgdDjuaF0d
/Ve3+ZhNIfb2txBE7T5HqUXekTbfusKcsUUbGrjtOkCPCyz4JCn+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. It includes fields for Host name (server.example.com), Principal name (host/server.example.com@RHTS.ENG.BOS.REDHAT.COM), Description (empty text area), Locality, Location, Platform, and Operating system (all empty text boxes). Below these is the 'SSH public keys' section, which is highlighted with a red box. It displays the key fingerprint 'BC:BD:BF:81:51:A5:74:07:C2:D5:EE:11:8C:95:48:3C', the host name 'server.example.com', the key type '(ssh-rsa)', and links for 'Show/Set key' and 'Delete'. Below the key list is an 'Add' link. At the bottom, there is a 'MAC address' field with an 'Add' link.

**Figure 11.1. 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 "Configuring Services: OpenSSH and Cached Keys" in the System-Level Authentication Guide](#).

#### 11.5.4. 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

RSA and DSS host keys are created by the **ipa-client-install** command, unless the SSH service is explicitly disabled in the installation script.

1. Run the **host-mod** command with the **--sshpubkey** option to upload the base64-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 RjlzYQo==" --updatedns host1.example.com
```

A real key also usually ends with an equal sign (=) but is longer.

To upload more than one key, enter multiple **--sshpubkey** command-line parameters:

```
--sshpubkey="RjlzYQo==" --sshpubkey="ZEt0TAo=="
```



### Note

A host can have multiple public keys.

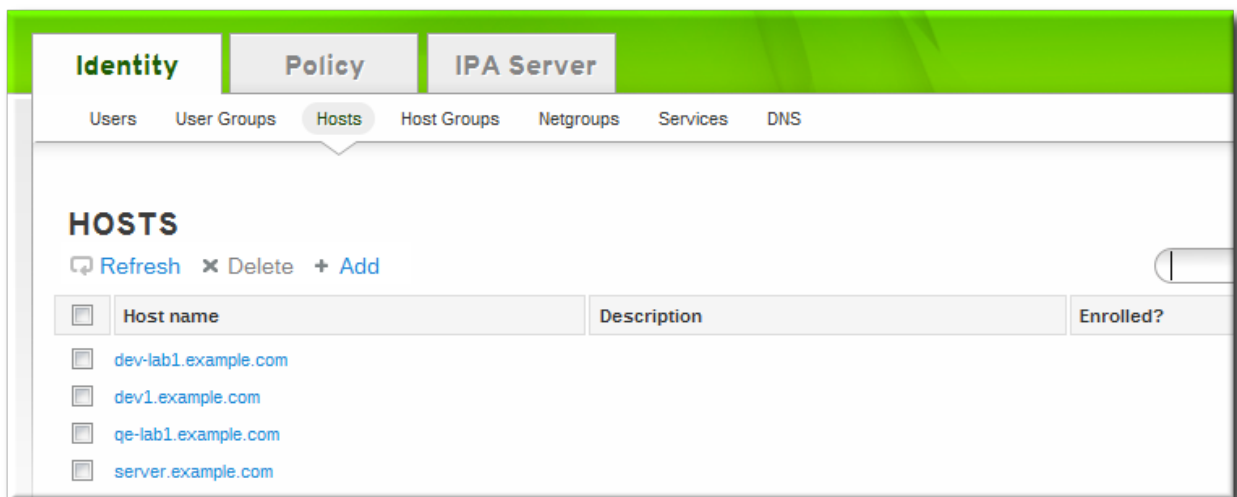
2. After uploading the host keys, configure SSSD to use Identity Management as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing host keys. This is covered in [the "Configuring Services: OpenSSH and Cached Keys" in the System-Level Authentication Guide](#).

## 11.5.5. 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.

- 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](#)

- 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
```

## 11.6. Setting Ethers Information for a Host

NIS can host an ethers table which can be used manage DHCP configuration files for systems based on their platform, operating system, DNS domain, and MAC address — all information stored in host entries in IdM.

In Identity Management, each system is created with a corresponding ethers entry in the directory, in the **ou=ethers** subtree.

```
cn=server,ou=ethers,dc=example,dc=com
```

This entry is used to create a NIS map for the ethers service which can be managed by the NIS compatibility plug-in in IdM.

To configure NIS maps for ethers entries:

- Add the MAC address attribute to a host entry. For example:

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa host-mod --macaddress=12:34:56:78:9A:BC
server.example.com
```

2. Open the **nsswitch.conf** file.
3. Add a line for the ethers service, and set it to use LDAP for its lookup.

```
ethers: ldap
```

4. Check that the ethers information is available for the client.

```
[root@server ~]# getent ethers server.example.com
```

## 11.7. Managing Host Groups

Host groups are a way of centralizing control over important management tasks, particularly access control.

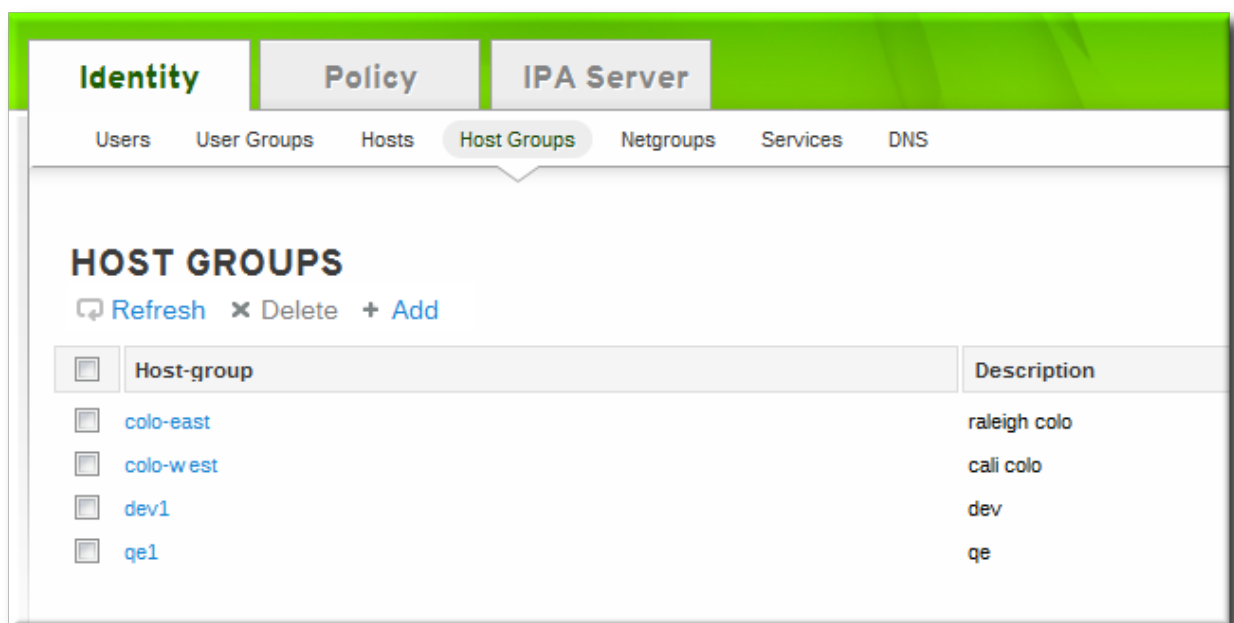
All groups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, IdM allows *nested groups*, where a group is a member of another group. In that case, all of the group members of the member group automatically 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.

### 11.7.1. Creating Host Groups

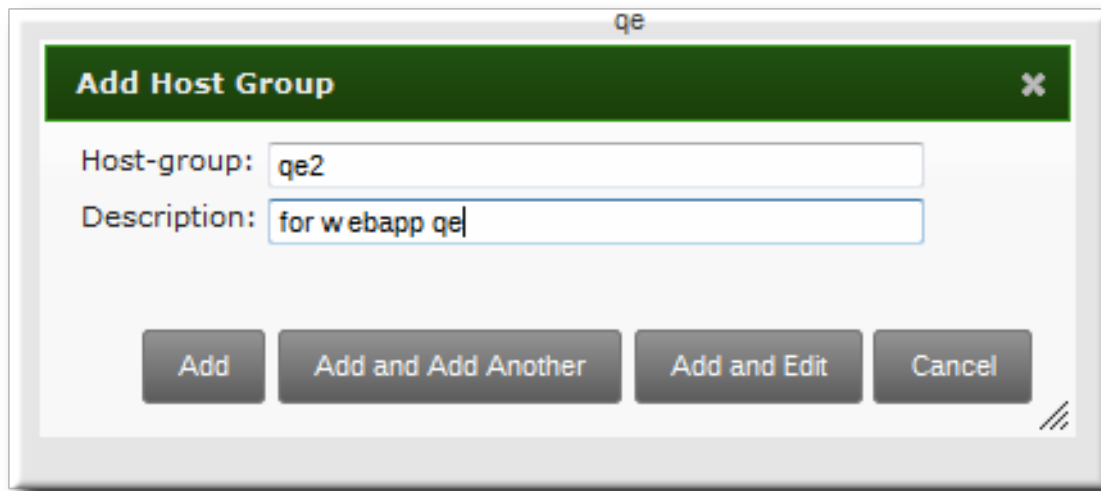
#### 11.7.1.1. Creating Host Groups from the Web UI

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 11.7.2.2, “Adding Host Group Members from the Web UI”](#).

### 11.7.1.2. 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.

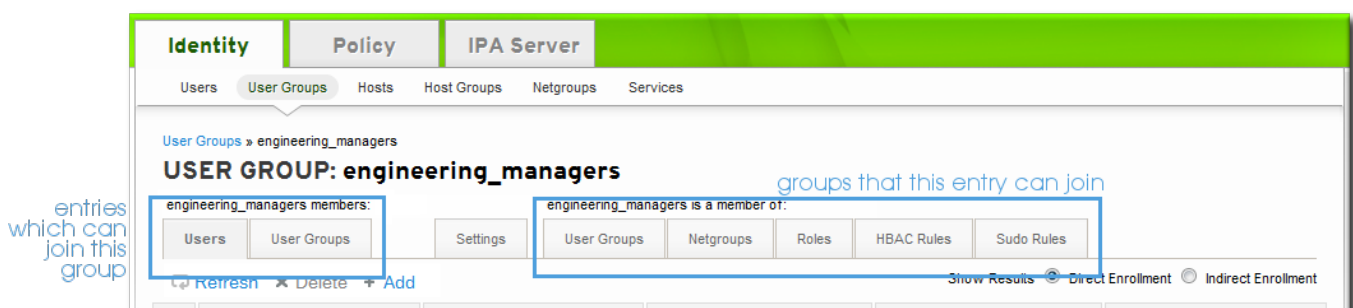
```
$ ipa hostgroup-add groupName --desc="description"
```

## 11.7.2. Adding Host Group Members

### 11.7.2.1. 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 matching entries and adds them as members.

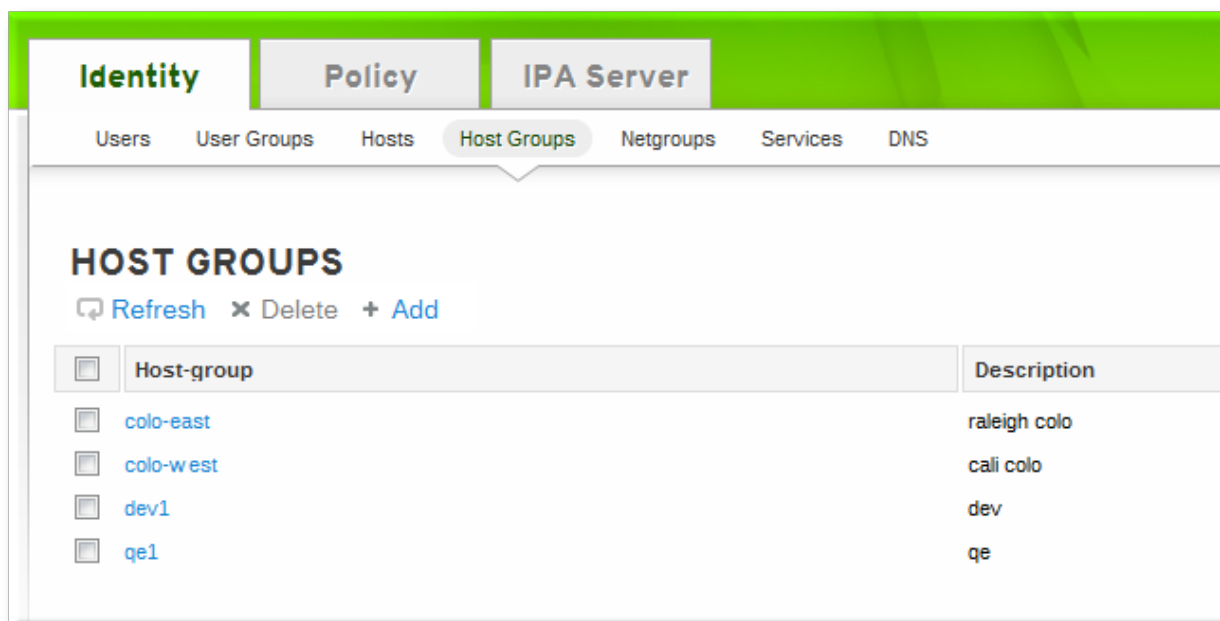
However, it is also possible for an entity to be added to a group through its own configuration. Each entry has a list of tabs that displays group types that the entry can join. The list of all groups of that type is displayed, and the entity can be added to multiple groups at the same time.



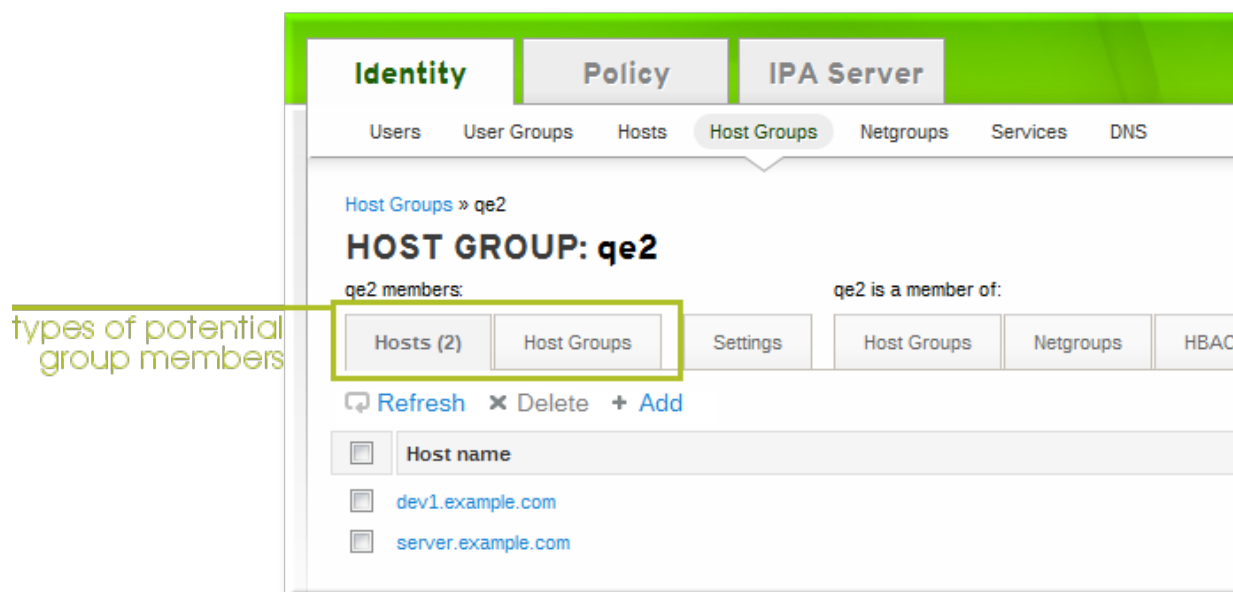
**Figure 11.2. Member Of ...**

### 11.7.2.2. Adding Host Group Members from the Web UI

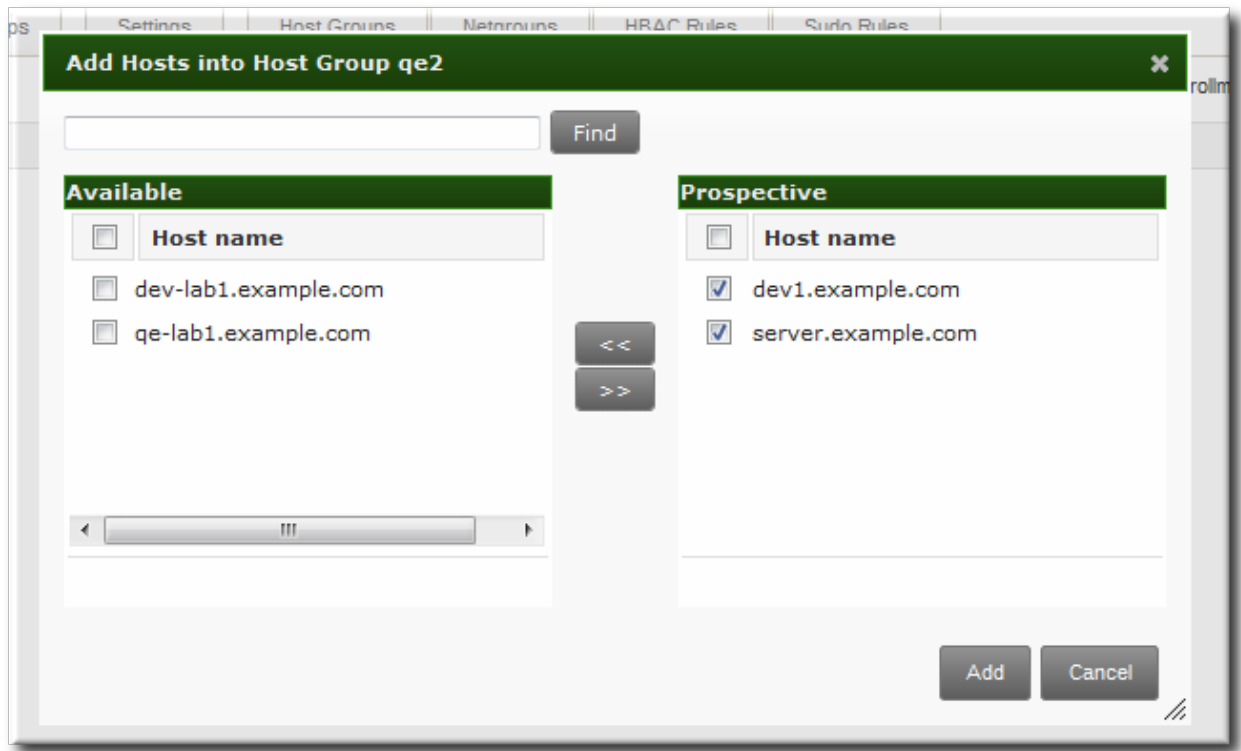
1. Open the **Identity** tab, and select the **Host Groups** subtab.
2. Click the name of the group to which to add members.



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.

### 11.7.2.3. Adding Host 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 the hosts to add. Lists of entries can be set by using the option multiple times with the same command or by listing the options in a comma-separated list inside curly braces, such as `--option={val1,val2,val3}`.

```
$ ipa hostgroup-add-member groupName [--hosts=host1 ...] [--
hostgroups=hostGroup1 ...]
```

For example, this adds three hosts to the **caligroup** group:

```
$ ipa hostgroup-add-member caligroup --hosts=ipaserver.example.com --
hosts=client1.example.com --hosts=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 --
groups=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 12. 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

### 12.1. 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`.



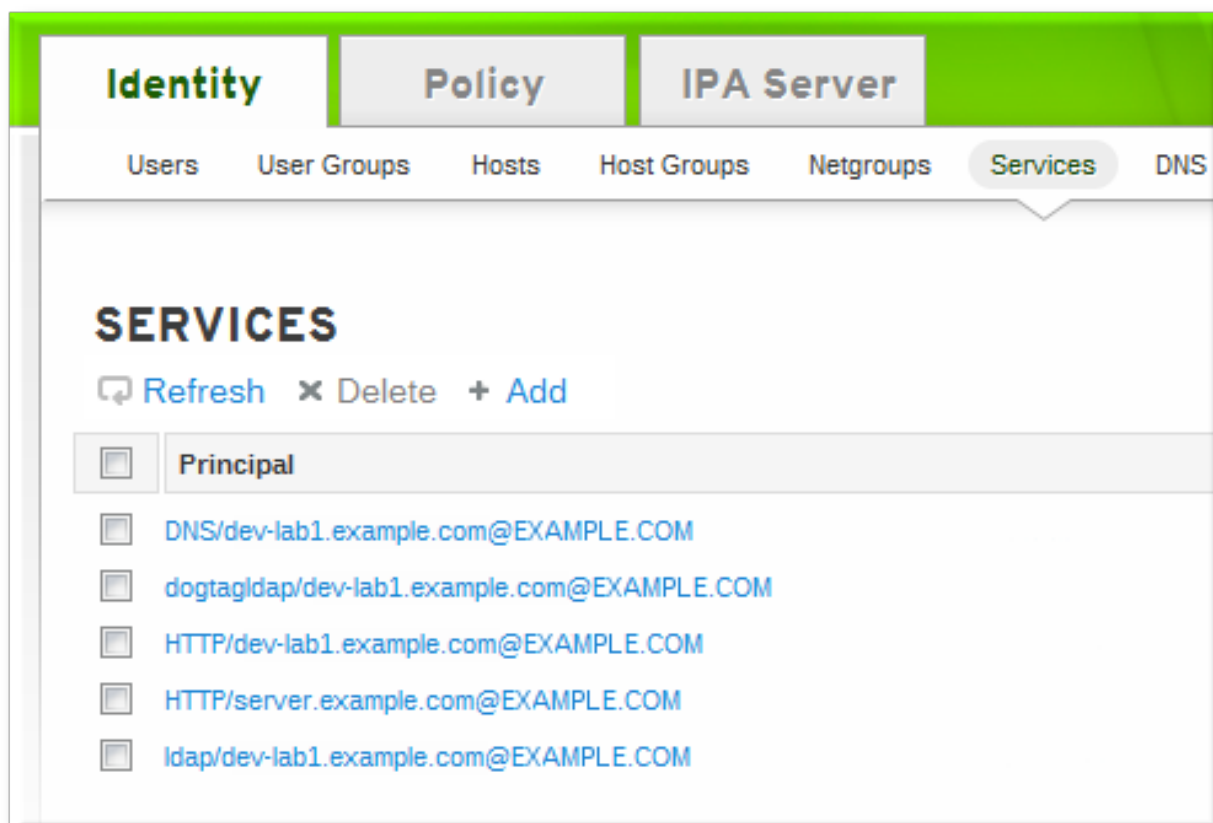
#### Note

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.

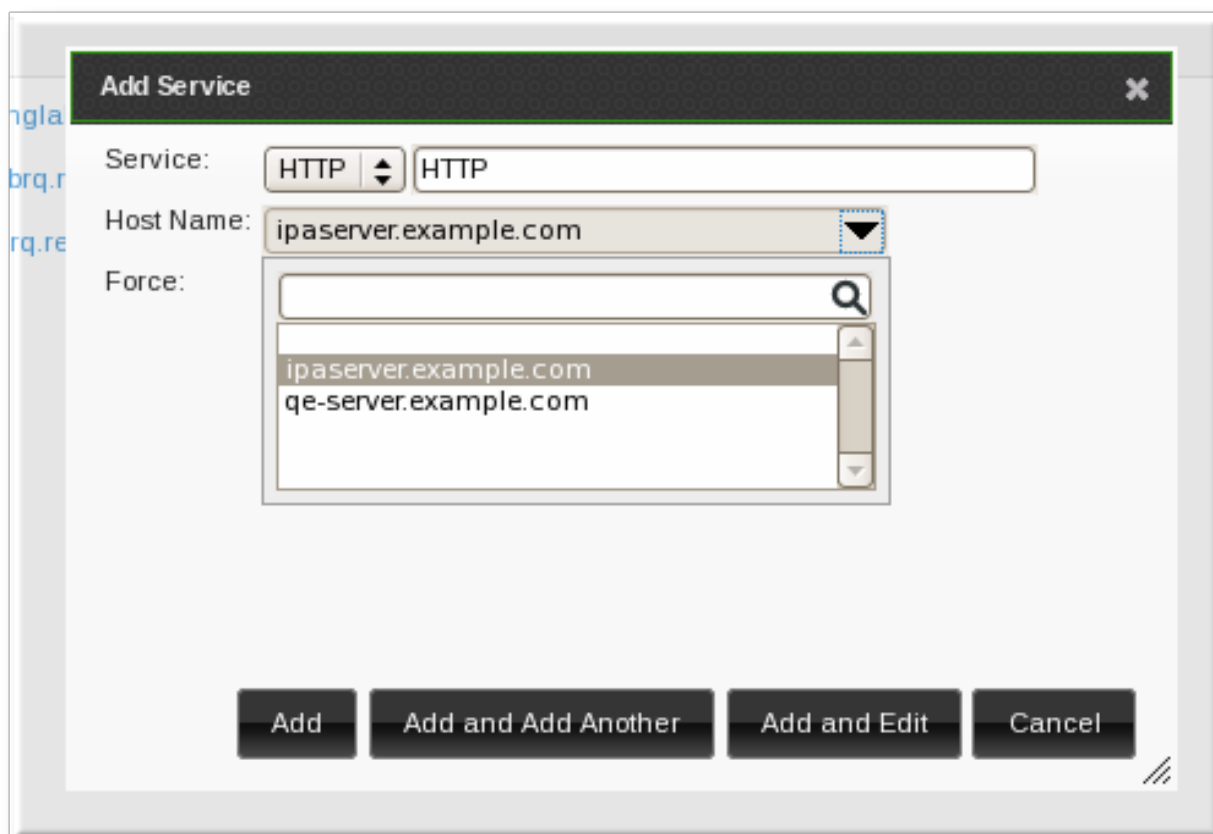
Similar locations can be specified for each service that needs to be made Kerberos aware. There is no specific location that must be used, but, when using `ipa-getkeytab`, you should avoid using `/etc/krb5.keytab`. This file should not contain service-specific keytabs; each service should have its keytab 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.

#### 12.1.1. 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.

- 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 aes256-
cts
```

- ✱ 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 list of encryption types to include in the keytab. This supersedes any default encryption type. Lists of entries can be set by using the option multiple times with the same command invocation or by listing the options in a comma-separated list inside curly braces, such as **--option={val1,val2,val3}**.



### Warning

Creating a new key resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

## 12.1.2. Adding Services and Keytabs from the Command Line

- 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
```

- 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 aes256-cts
```

- ✧ 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. Lists of entries can be set by using the option multiple times with the same command invocation or by listing the options in a comma-separated list inside curly braces, such as **--option={val1,val2,val3}**.



### Warning

The **ipa-getkeytab** command resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

## 12.2. Creating Certificates for Services

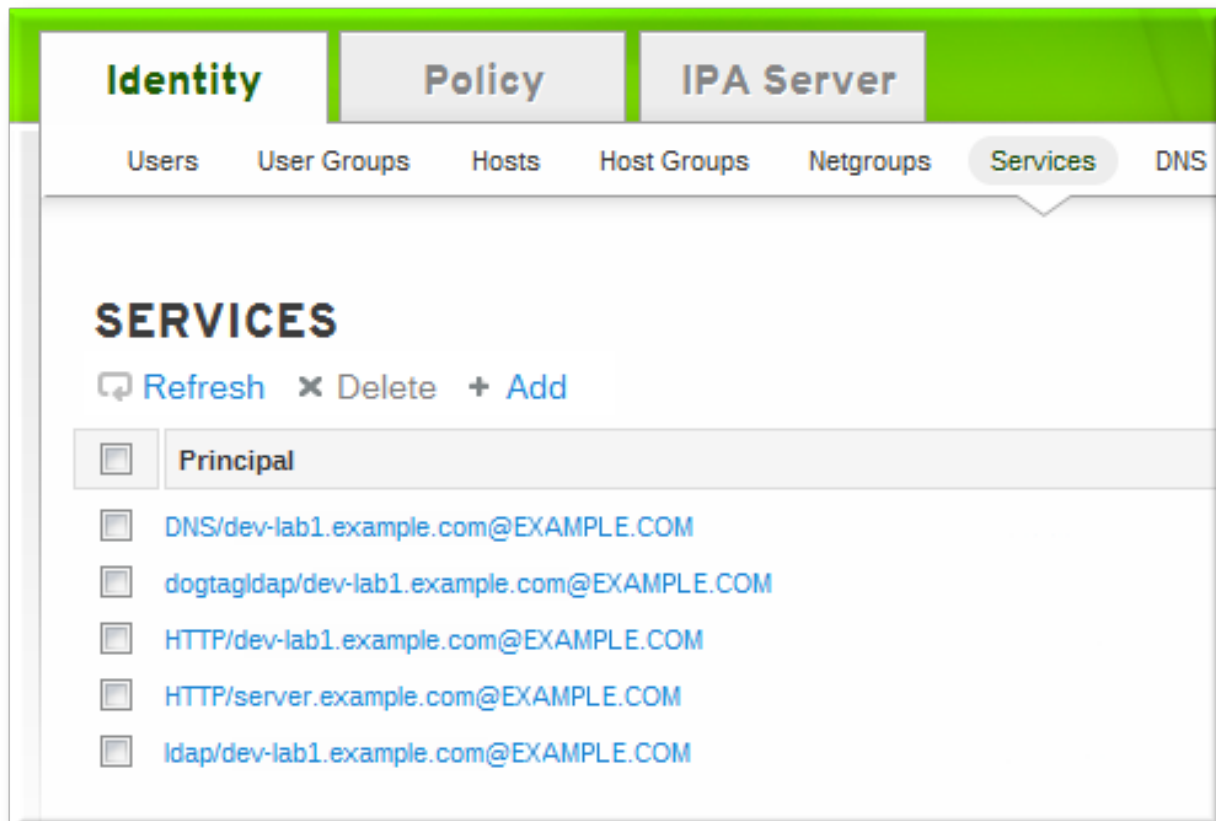
By default, the IdM server has an integrated certificate authority. This CA can be used to create, revoke, and issue certificates for services in the IdM domain.

### 12.2.1. Showing Certificates

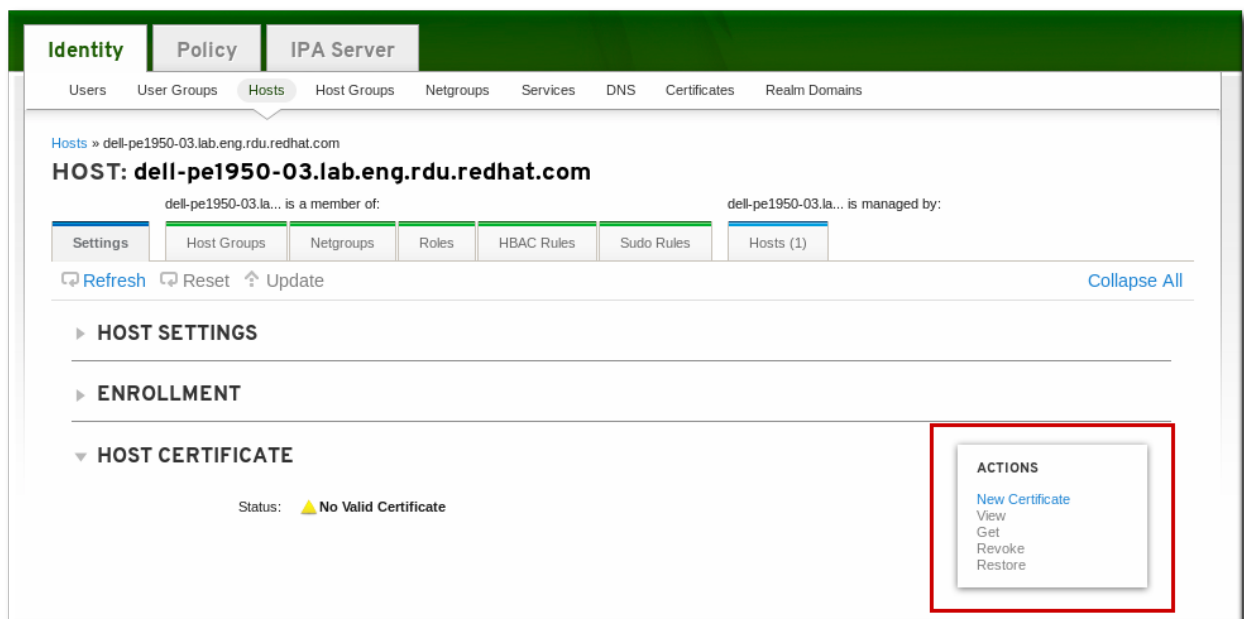
#### 12.2.1.1. In the Service Entry in the UI

1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the name of the service.

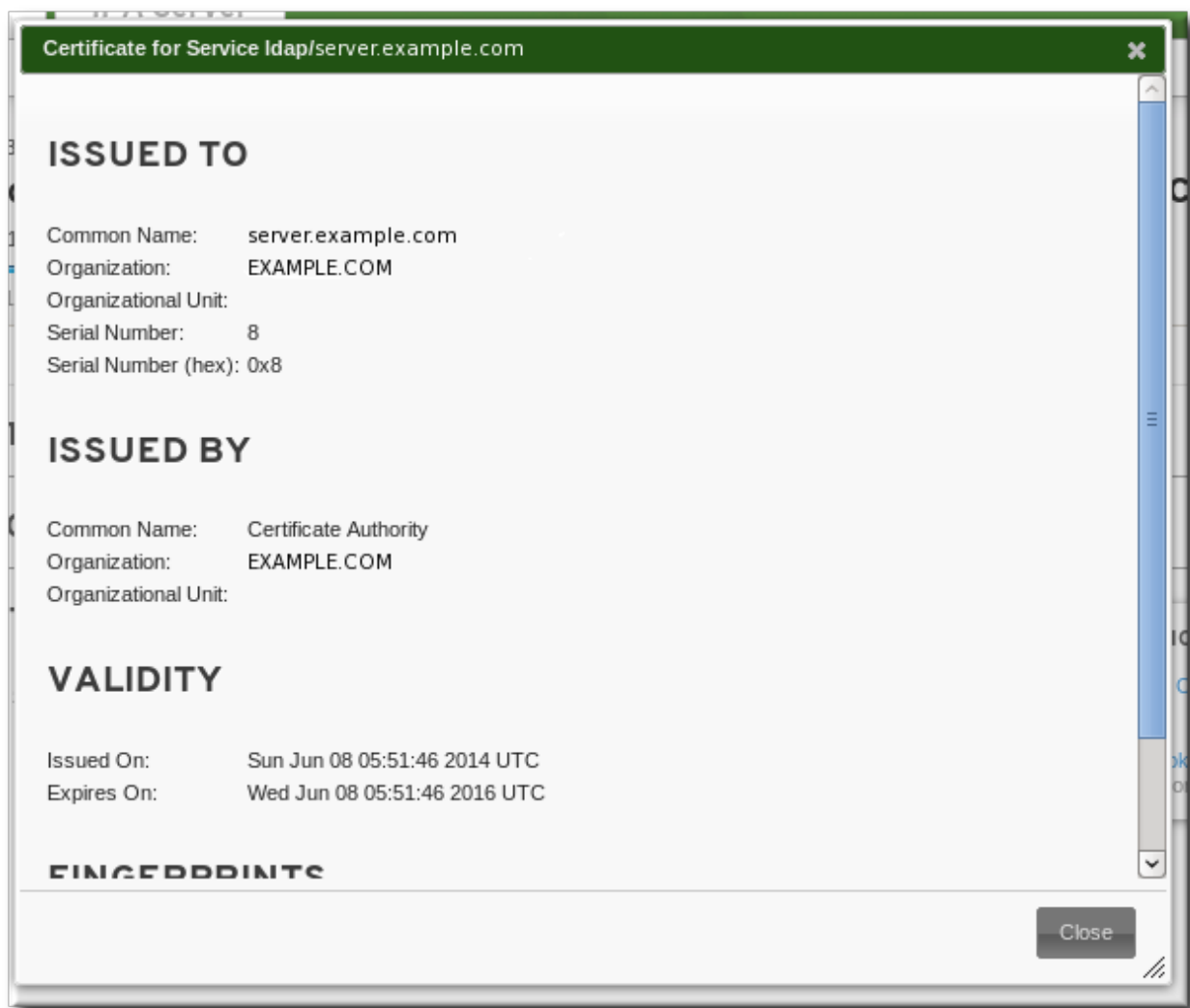




3. In the **Settings** tab, scroll to the **Service Certificate** tab at the bottom.

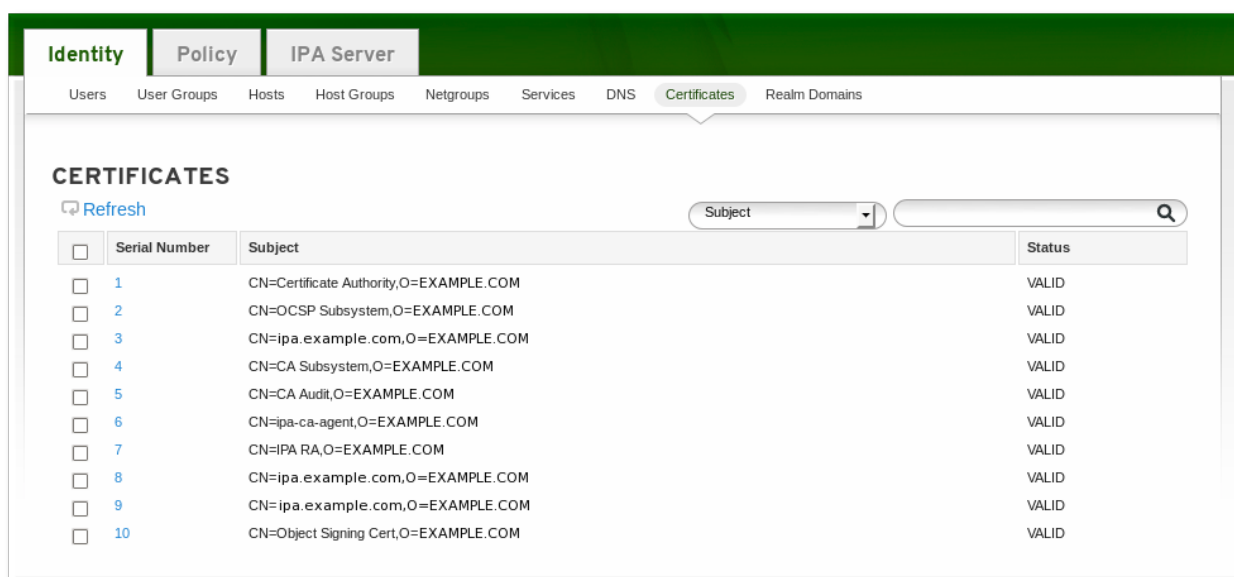


4. If a certificate has been issued, click the **View** link to display the details about the certificate. To retrieve the full certificate, click the **Get** link.

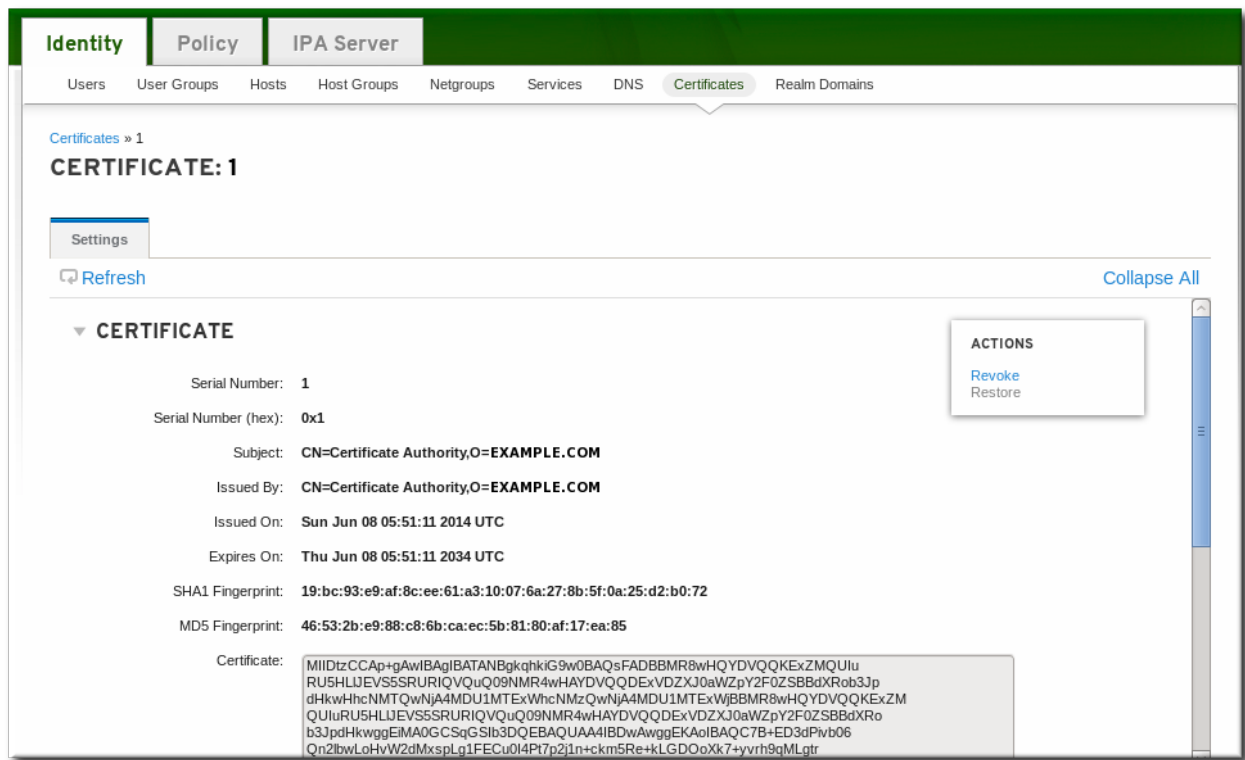


### 12.2.1.2. In the Certificate List in the UI

1. Open the **Identity** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to view.



3. The top of the certificate entry shows the details of the certificate, such as its CN. The full certificate blob is available at the bottom of the page.



### 12.2.1.3. In the Command Line

All of the certificates which have been issued by the IdM CA are listed with the **ipa cert-find** command.

```
[root@server ~]# kinit admin
[root@server ~]# ipa cert-find
-----
10 certificates matched
-----
Serial number (hex): 0x1
Serial number: 1
Status: VALID
Subject: CN=Certificate Authority,O=EXAMPLE.COM
...
-----
Number of entries returned 10
-----
```

With a large number of certificates, it can be easier to search for a specific certificate by serial number or by an issue date. To search by a serial number, simply include it with the **cert-show** command.

```
[root@server ~]# ipa cert-show 132
Serial number: 132
Certificate:
MIIDtzCCAp+gAwIBAgIBATANBgkqhkiG9w0BAQsFADBBMR8wHQYDVQQKEZXMQUIu
...
LxIQjrEFtJmoBGB/TWrlwGEWylayr4iTEflayZ+RGNylLalEAtk9RLjEjg==
Subject: CN=Certificate Authority,O=EXAMPLE.COM
Issuer: CN=Certificate Authority,O=EXAMPLE.COM
Not Before: Sun Jun 08 05:51:11 2014 UTC
Not After: Thu Jun 08 05:51:11 2034 UTC
```

```

Fingerprint (MD5): 46:53:2b:e9:88:c8:6b:ca:ec:5b:81:80:af:17:ea:85
Fingerprint (SHA1):
19:bc:93:e9:af:8c:ee:61:a3:10:07:6a:27:8b:5f:0a:25:d2:b0:72
Serial number (hex): 0x132
Serial number: 132

```

The **--issuedon-from** and **--issuedon-to** options can set start/end points or a period of time to use to search for certificates.

```
ipa cert-find --issuedon-from=2013-02-01 --issuedon-to=2015-02-07
```

### 12.2.2. Revoking and Restoring Certificates

Every certificate has a specified expiration date, but there can be times when it is necessary to terminate (revoke) a certificate before that expiration. Revoking a certificate makes it invalid, so the service cannot use it for authentication.

When a certificate is revoked, there has to be a reason given. There are several different reasons — it was compromised, the entity has changed, the service is being pulled from service, or it has been replaced by a different certificate. The possible reasons are listed in [Table 12.1, “Revocation Reasons”](#).

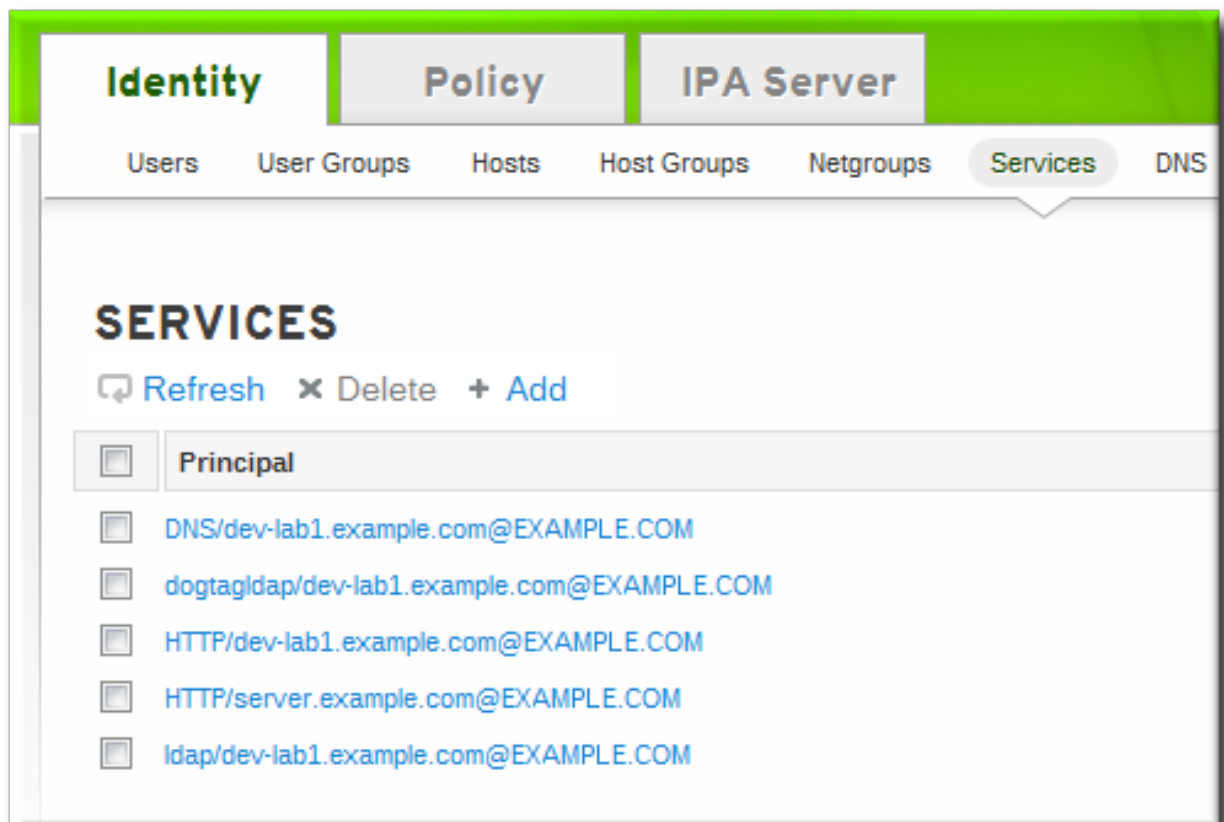
**Table 12.1. Revocation Reasons**

ID	Reason	Description
0	Unspecified	
1	Key Compromised	The underlying key was compromised. This could mean a token was lost or file was improperly accessed.
2	CA Compromised	The CA which issued the certificate was compromised.
3	Affiliation Changed	The person or service to which the certificate was issued is changing affiliations. This could mean that the person has left the company (or the service is being retired) or that it has moved departments, if the affiliation is tied to an organizational structure.
4	Superseded	The certificate has been replaced by a newer certificate.
5	Cessation of Operation	The service is being decommissioned.
6	Certificate Hold	The certificate is temporarily revoked. This is the only revocation reason that allows the certificate to be restored.

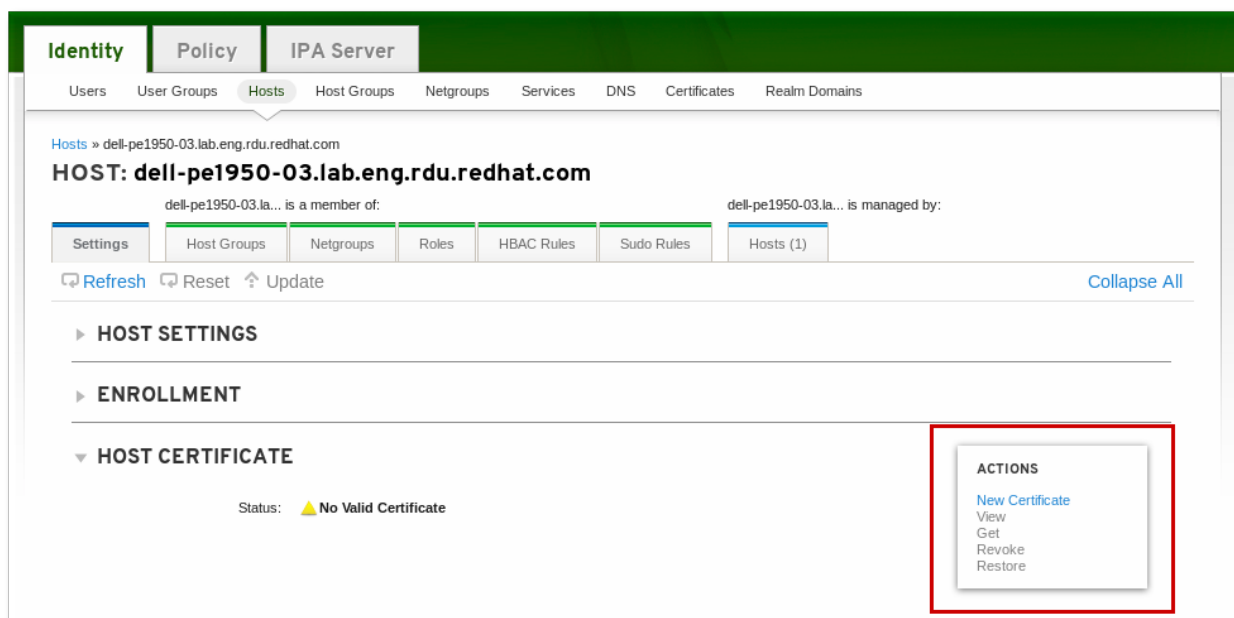
ID	Reason	Description
8	Remove from CRL	The certificate is not included in the certificate revocation list.
9	Privilege Withdrawn	The service should no longer be issued the certificate.
10	A Authority Compromise	The AA was compromised.

### 12.2.2.1. In the Service Entry in the UI

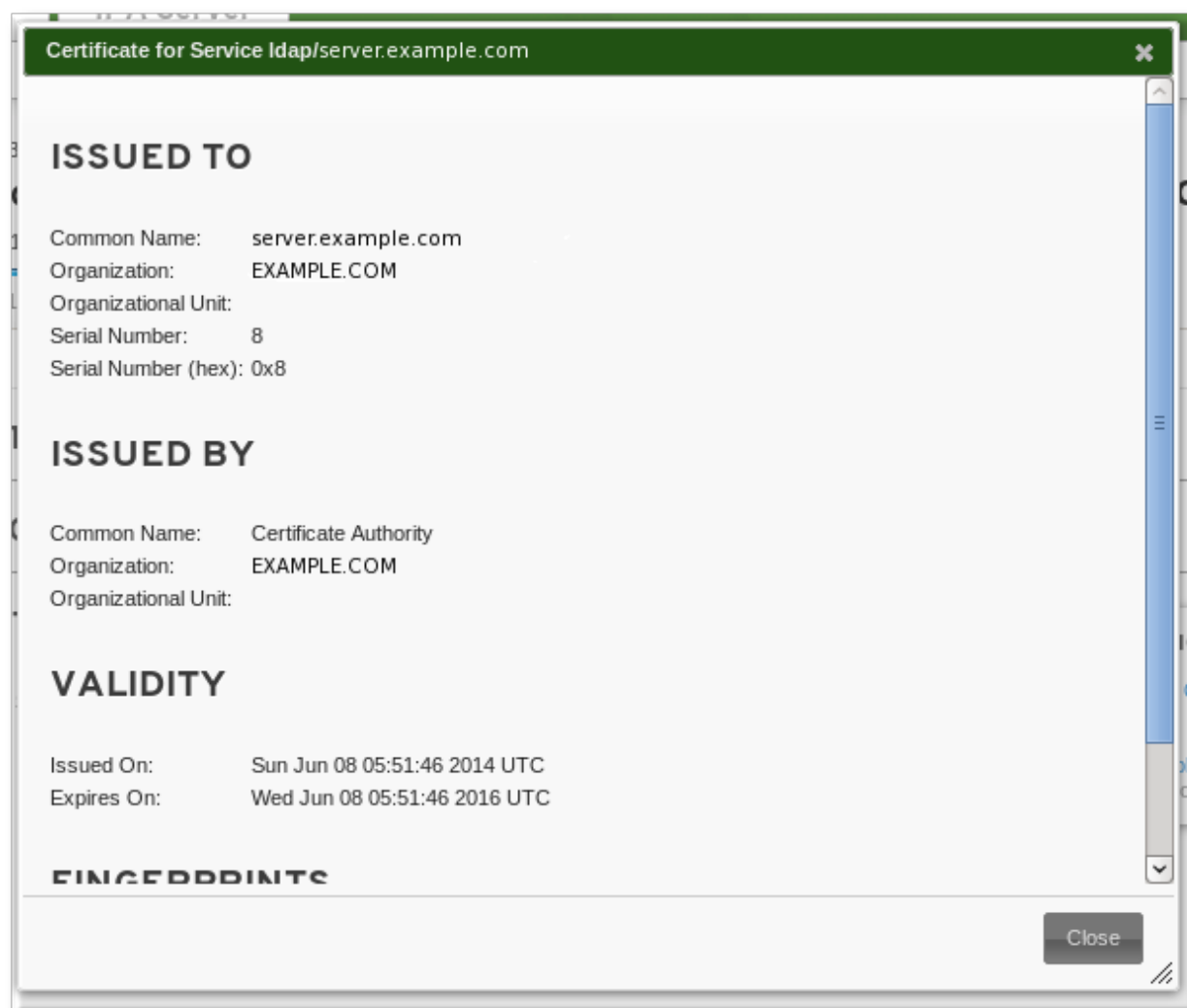
1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the name of the service.



3. In the **Settings** tab, scroll to the **Service Certificate** tab at the bottom.



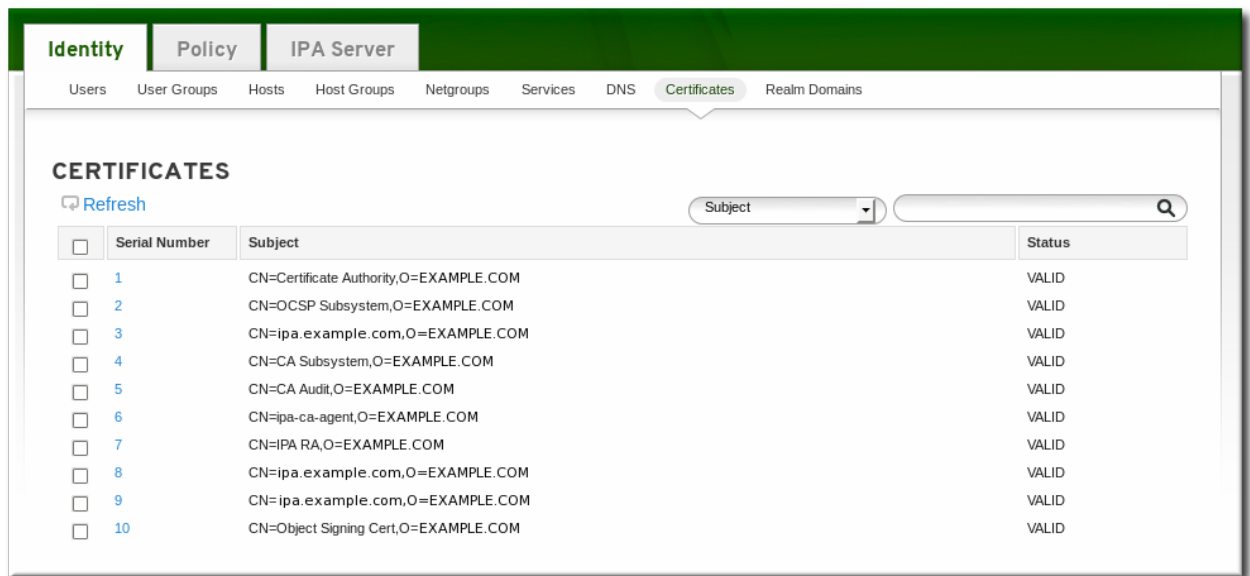
4. In the **Actions** area, click the **Revoke** link.
5. Select the reason for the revocation from the drop-down menu, and click the **Revoke** link. [Table 12.1, “Revocation Reasons”](#) describes the different options for revoking a certificate.



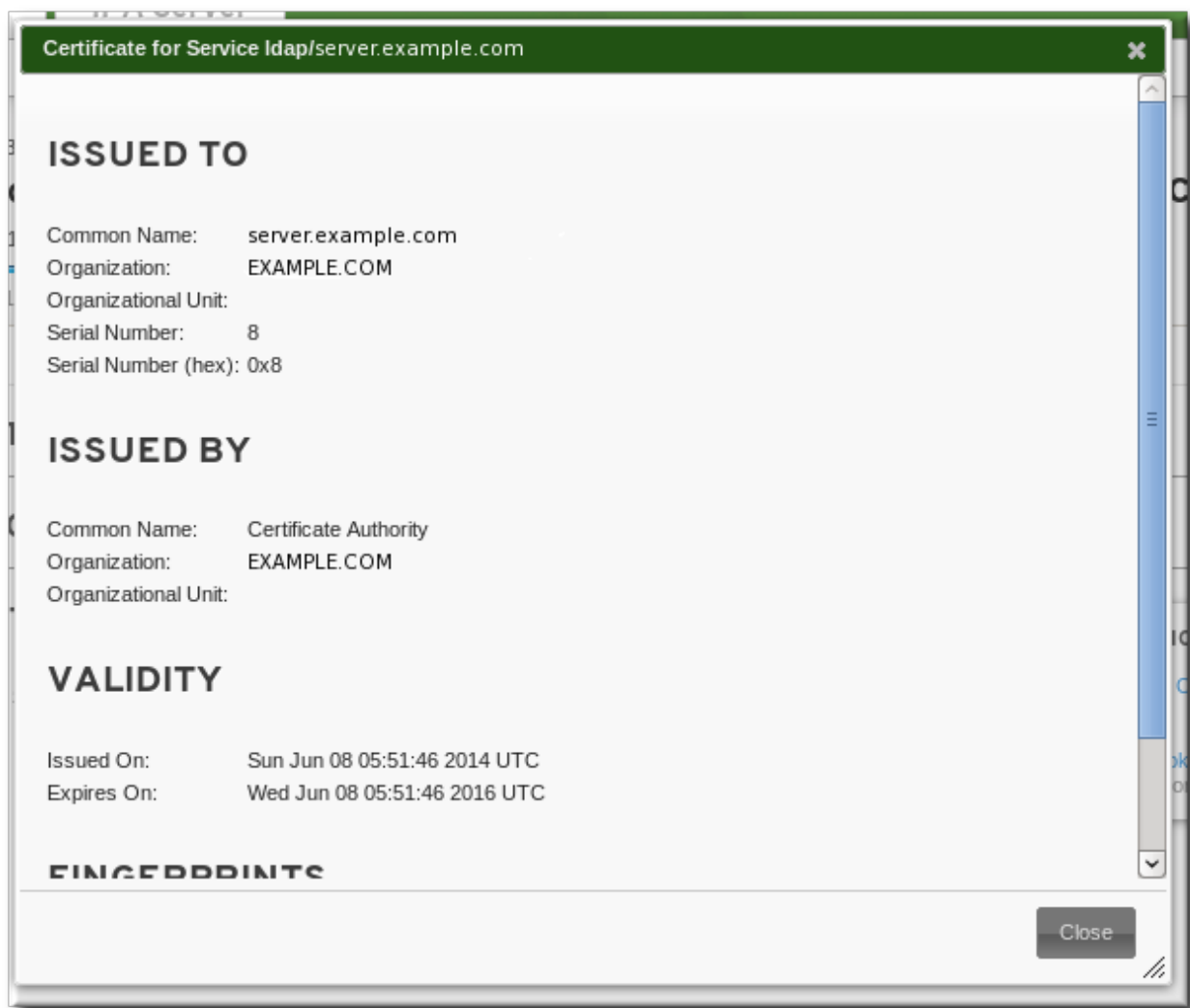
If the reason for the revocation is a certificate hold, then the certificate can be restored later by clicking the **Restore** link in the certificate actions menu.

### 12.2.2.2. In the Certificate List in the UI

1. Open the **Identity** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to view.



3. In the **Actions** area, click the **Revoke** link.
4. Select the reason for the revocation from the drop-down menu, and click the **Revoke** link. [Table 12.1, “Revocation Reasons”](#) describes the different options for revoking a certificate.



If the reason for the revocation is a certificate hold, then the certificate can be restored later by clicking the **Restore** link in the certificate actions menu.

### 12.2.2.3. In the Command Line

To revoke a certificate from the command line, specify the certificate serial number and give the reason for the revocation in the **--revocation-reason** option.

```
[root@server ~]# kinit admin
[root@server ~]# ipa cert-revoke --revocation-reason=6 1032
```

If the reason for the revocation is a certificate hold (6), then the certificate can be restored with the **cert-remove-hold** command.

```
[root@server ~]# ipa cert-remove-hold 1032
```

### 12.2.3. Requesting New Service Certificates

The certificate request must be generated with a third-party tool such as **certutil**. The resulting certificate request can be submitted through the IdM web UI or command-line tools.



The service must already exist for a certificate to be requested. If the service does not yet exist, then with the command line, there is an option to create the service as part of requesting the certificate.

### 12.2.3.1. In the UI

1. Generate a certificate request for the service. For example:

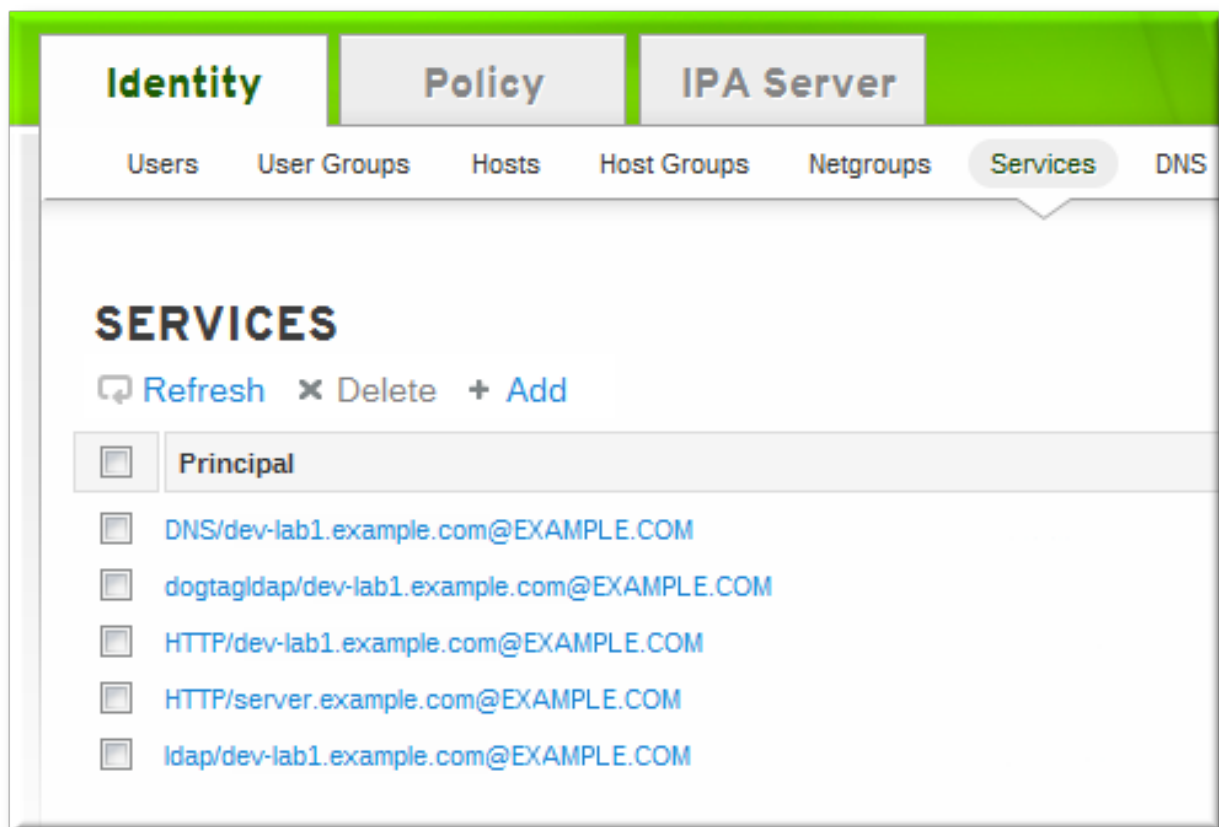
First, create a set of certificate databases that can be used to create and store the certificate locally.

```
[root@server ~]# certutil -N -d ~/test-certs/
```

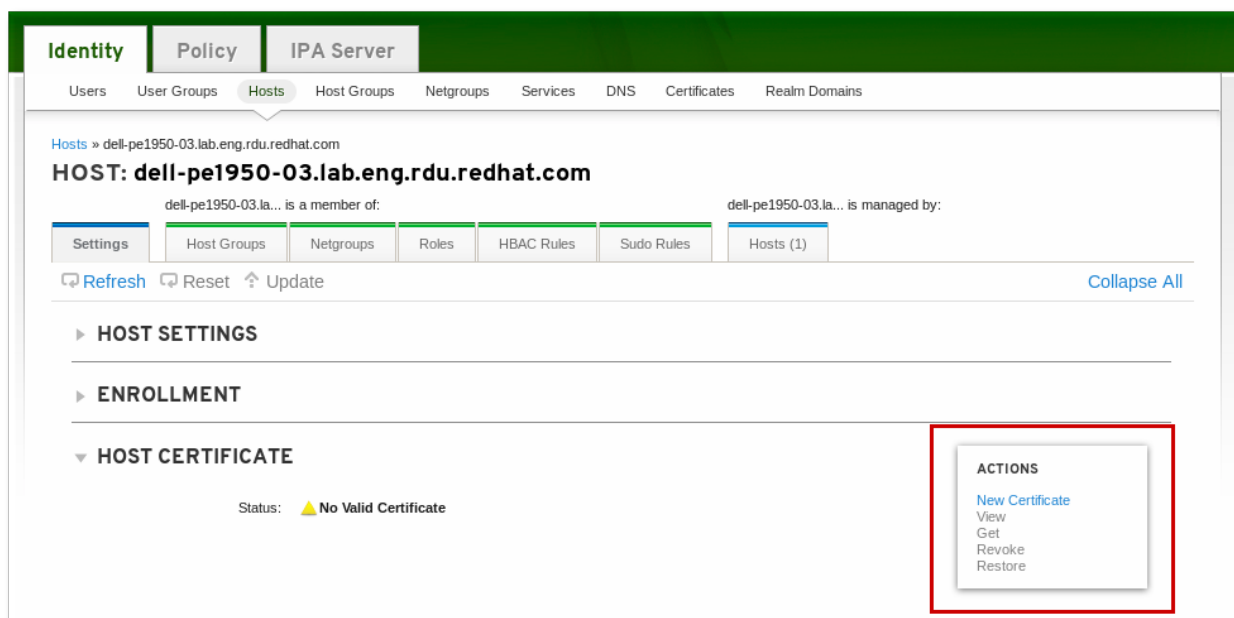
Then, create the certificate request.

```
[root@server ~]# certutil -R -d ~/test-certs -R -a -g 256 -s  
"CN=server.example.com,O=EXAMPLE.COM" -o ~/test-certs/service.csr
```

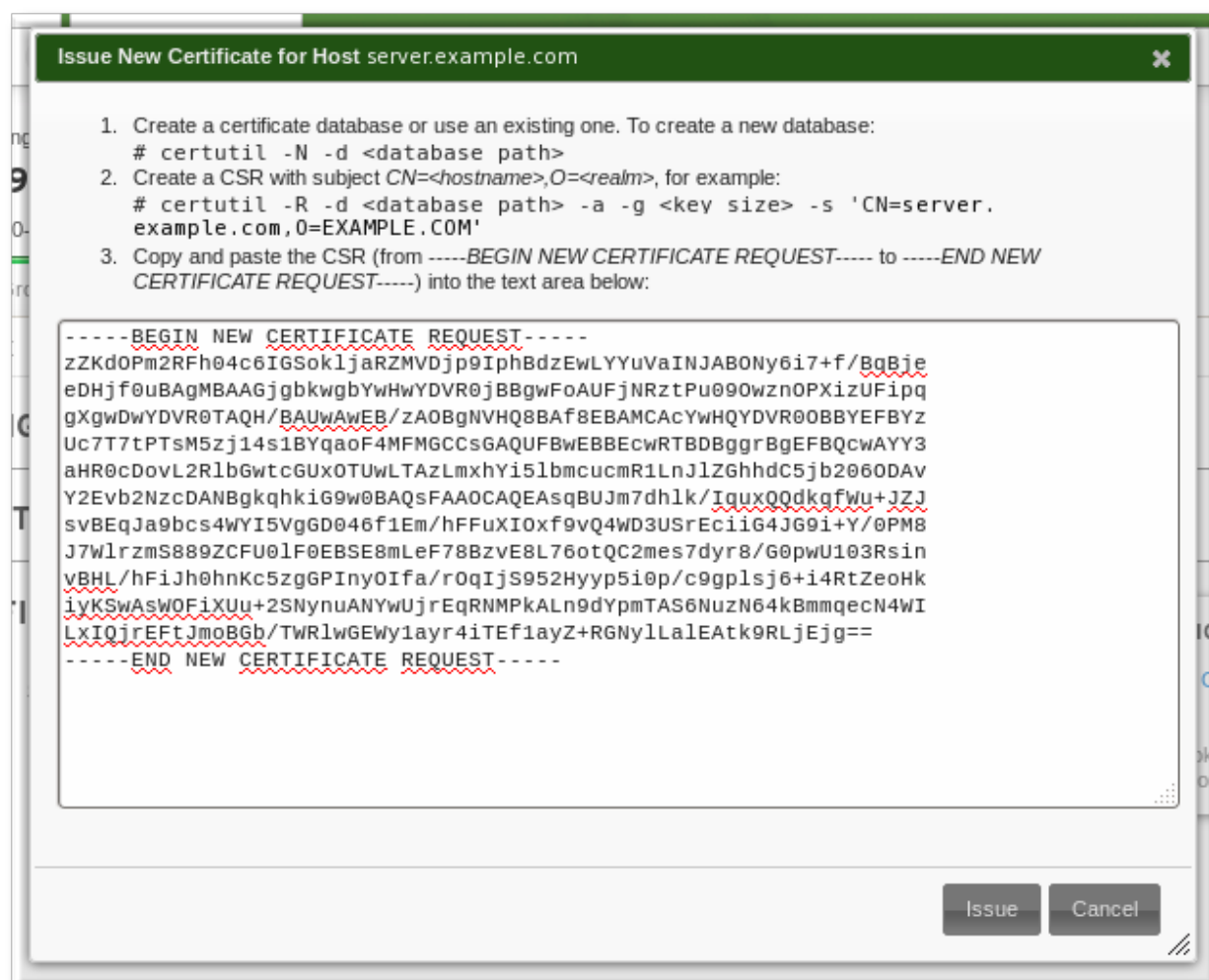
2. Copy the text of the new certificate request.
3. Open the **Identity** tab, and select the **Services** subtab.
4. Click the name of the service.



5. In the **Settings** tab, scroll to the **Service Certificate** tab at the bottom.



6. In the **Actions** area, click the **Request** link.
7. Paste in the body of the certificate request, including the **BEGIN NEW CERTIFICATE REQUEST** and **END NEW CERTIFICATE REQUEST** lines.



8. Click the **Issue** button.

### 12.2.3.2. In the Command Line

1. Generate a certificate request for the service. For example:

First, create a set of certificate databases that can be used to create and store the certificate locally.

```
[root@server ~]# certutil -N -d ~/test-certs/
```

Then, create the certificate request.

```
[root@server ~]# certutil -R -d ~/test-certs -R -a -g 256 -s  
"CN=server.example.com,O=EXAMPLE.COM" -o ~/test-certs/service.csr
```

2. Submit the PEM file of the certificate request to the IdM server. Along with the request itself, specify the Kerberos principal to create and associate with the newly-issued certificate.

If the service does not already exist, then use the **--add** option to create the service, and then issue the certificate.

```
[root@server ~]# ipa cert-request -add --  
principal=ldap/server.example.com service.csr
```

Note that you can use the **--profile-id** option with the **ipa cert-request** command to select a custom certificate profile to be used for the certificate. By default, IdM uses the **caIPAServiceCert** profile. For more information about certificate profiles, see [Section 26.9, "Certificate Profiles"](#).

## 12.3. 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.

## 12.4. 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.

## 12.5. 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 aes256-cts
```

2. Either direct multiple servers or services to use the same file, or copy the file to individual servers as required.

## 12.6. 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.

### 12.6.1. 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.

### 12.6.2. 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:

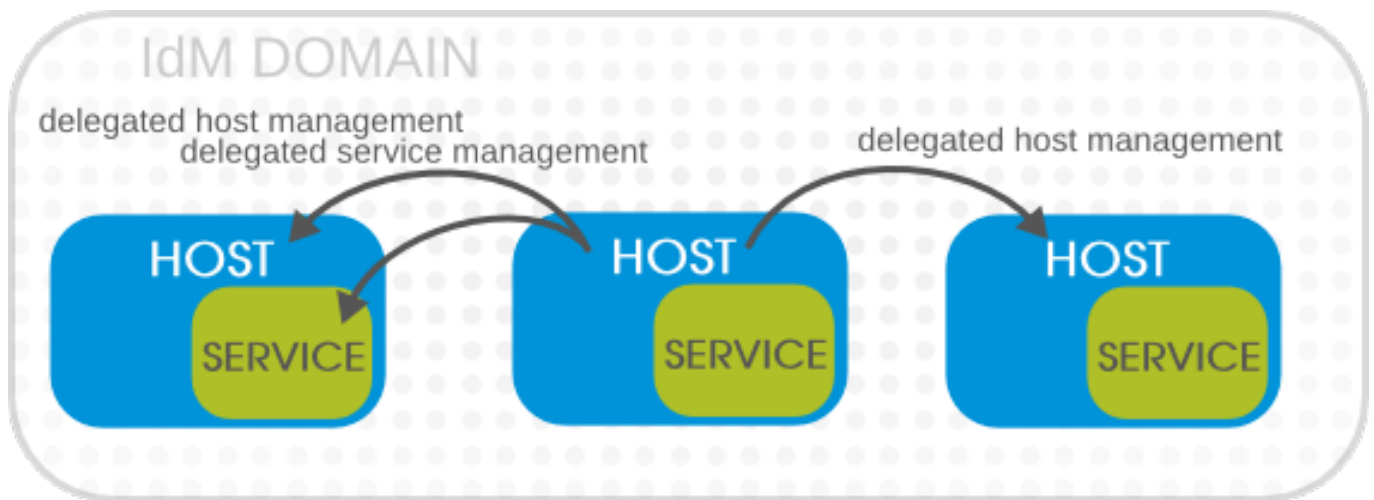
```
[root@ipaserver ~]# ipa-getkeytab -s ipaserver.example.com -p
HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e aes256-cts
```

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 13. Delegating User 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 13.1. 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.

### 13.1. 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 with the control:

```
# ipa service-add-host principal --hosts=hostnames
```

For example:

```
[root@server]# 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:

```
[root@server ]# 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:

```
[root@server ]# 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
```

Note that you can use the **--profile-id** option with the **ipa cert-request** command to select a custom certificate profile to be used for the certificate. By default, IdM uses the **caIPAServiceCert** profile. For more information about certificate profiles, see [Section 26.9, “Certificate Profiles”](#).

## 13.2. 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.

```
[root@server ]# kinit admin
```

2. Add the **managedby** entry. For example, this delegates authority over client2 to client1.

```
[root@server ]# ipa host-add-managedby client2.example.com --
hosts=client1.example.com
```

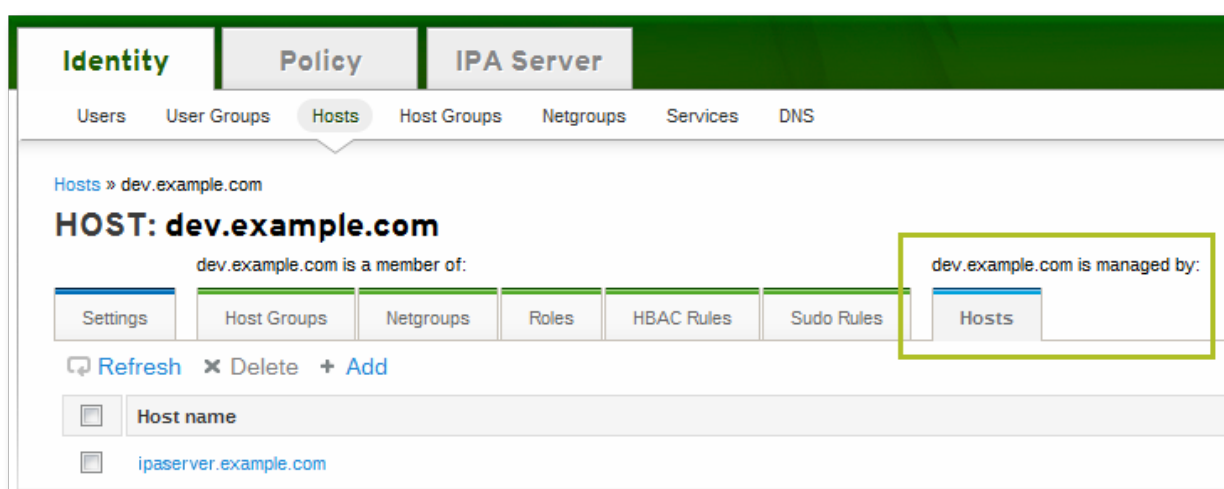
3. Obtain a ticket as the host **client1** and then retrieve a keytab for **client2**:

```
[root@server ]# kinit -kt /etc/krb5.keytab host/`hostname`
[root@server ~]# ipa-getkeytab -s `hostname` -k
/tmp/client2.keytab -p host/client2.example.com
Keytab successfully retrieved and stored in: /tmp/client2.keytab
```

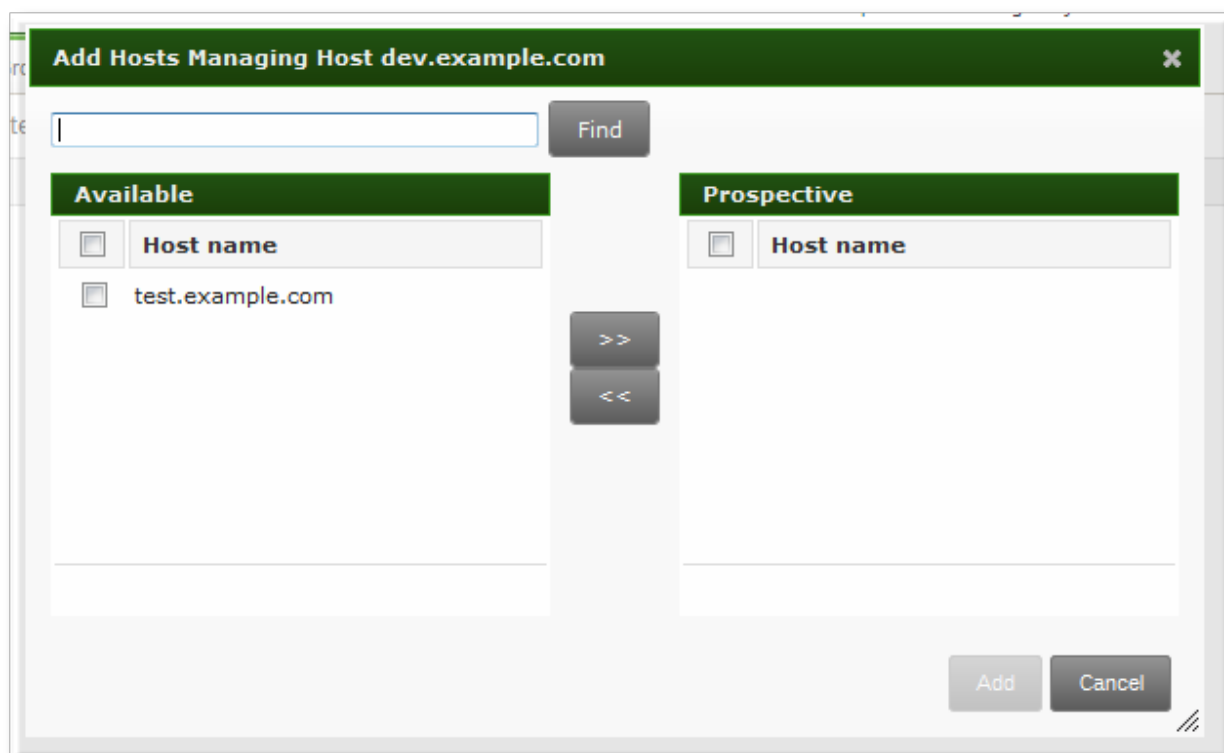
## 13.3. Delegating Host or Service Management in the Web UI

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.



## 13.4. 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:

```
[root@server ]# kinit -kt /etc/krb5.keytab  
host/ipa.example.com@EXAMPLE.COM
```

To access a service:

```
[root@server ]# kinit -kt /etc/httpd/conf/krb5.keytab  
http/ipa.example.com@EXAMPLE.COM
```

## Chapter 14. 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.

### 14.1. 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:

```
[root@server ~]# 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.

## 14.2. 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. Enable the NIS listener and compatibility plug-ins:

```
[root@ipaserver ~]# ipa-nis-manage enable
[root@ipaserver ~]# ipa-compat-manage enable
```

2. 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"
```

3. Restart the Directory Server to load the new plug-in configuration.

```
[root@ipaserver ~]# systemctl restart dirsrv.target
```

## 14.3. Creating Netgroups

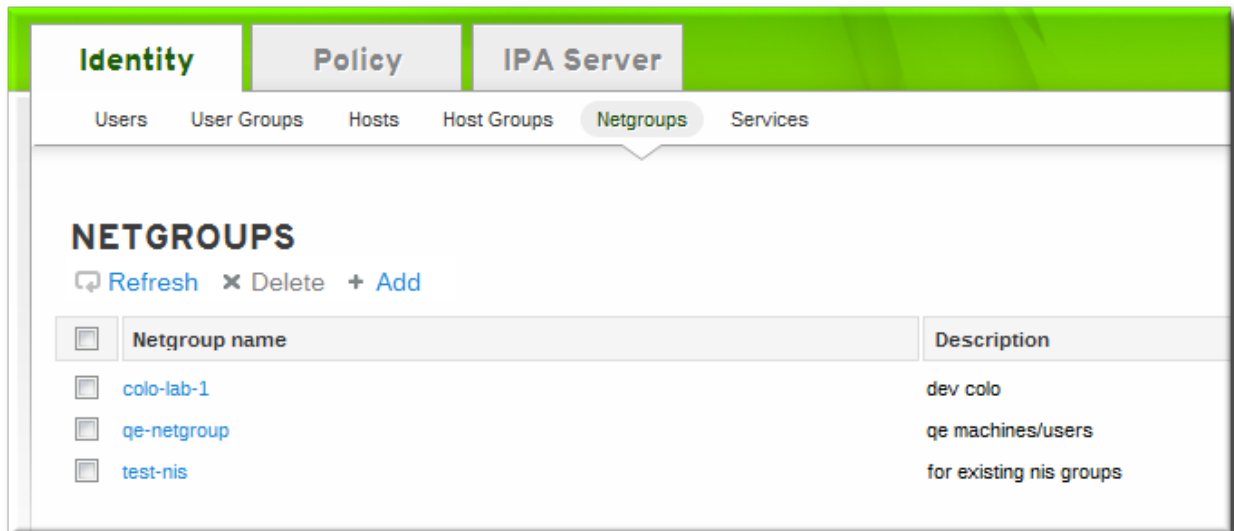
All netgroups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, IdM allows *nested groups*, where a group is a member of another group. In that case, all of the group members of the member group 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.

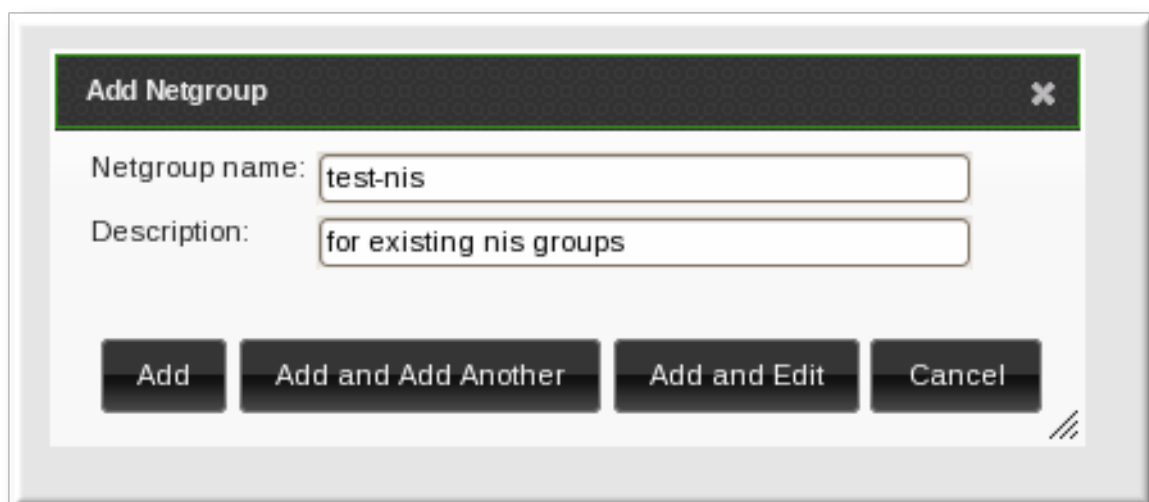
### 14.3.1. Adding Netgroups

#### 14.3.1.1. With the Web UI

1. Open the **Identity** tab, and select the **Netgroups** subtab.
2. 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.

- Click the **Add and Edit** button to go immediately to the netgroup's edit pages.
- Optionally, set the NIS domain for the netgroup. This defaults to the IdM domain, but it can be changed.
  - Click the **Settings** tab.
  - Enter the name of the alternate NIS domain in the **NIS domain name** field.

The screenshot shows the 'Netgroups' tab in the Identity Management web UI. The main heading is 'NETGROUP: qe-netgroup'. Below it, there are tabs for 'Hosts', 'Host Groups', 'Users', 'User Groups', 'Netgroups', 'Settings', and 'Netgroups'. The 'Netgroups' tab is selected. The 'NETGROUP SETTINGS' section is expanded, showing the following fields:

- Netgroup name:** qe-netgroup
- Description:** qe machines/users
- NIS domain name:** example.com (with an 'undo' button next to it)

At the top of the settings section, there are links for 'Reset' and 'Update'.

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 14.3.2.1, “With the Web UI”](#).

#### 14.3.1.2. 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.

```
[jsmith@server ~]$ ipa netgroup-add --desc="description" [--  
nisdomain=domainName] groupName
```

For example:

```
[root@server ~][root@server ~]# ipa netgroup-add --desc="my new  
netgroup" example-netgroup  
[root@server ~]# ipa netgroup-add-member --hosts=ipa.example.com  
example-netgroup  
[root@server ~]# 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.

### 14.3.2. Adding Netgroup Members



## Note

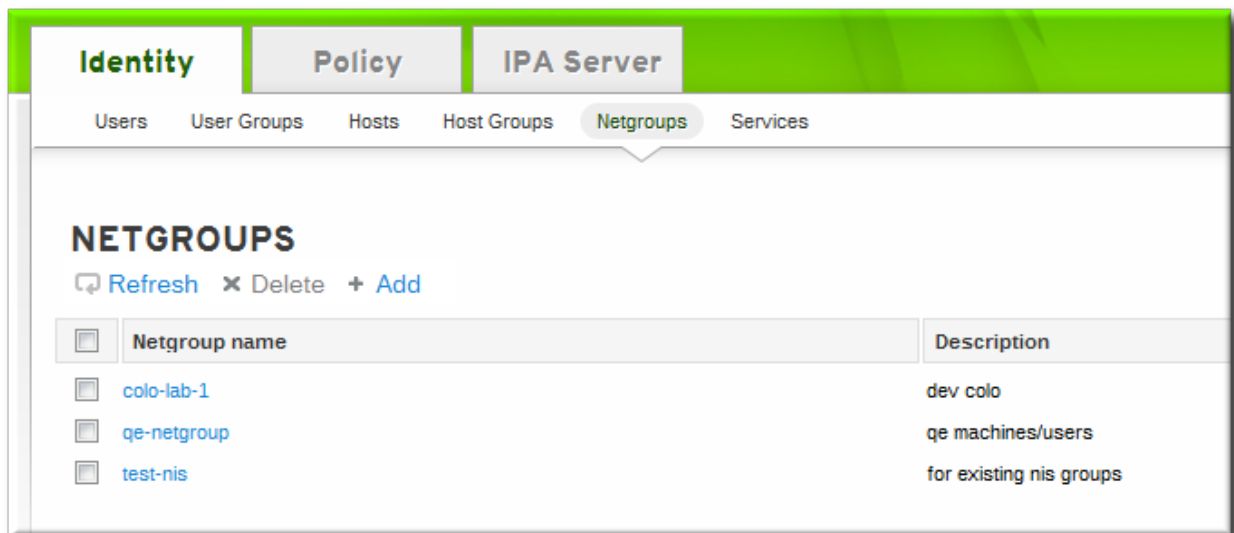
Netgroups can contain user groups, host groups, and other netgroups as their members. These are *nested* groups.

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.

### 14.3.2.1. 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.

Netgroups » test

## NETGROUP: test

test members: test is a member of:

Settings Netgroups Netgroups

Refresh Reset Update Collapse

### GENERAL

Netgroup name: test

Description: \* test

NIS domain name: server.example.com

### USER

User category the rule applies to: ☐ Anyone ☒ Specified Users and Groups

<input type="checkbox"/>	Users	<input type="button" value="Delete"/>	<input checked="" type="button" value="Add"/>
<input type="checkbox"/>	User Groups	<input type="button" value="Delete"/>	<input type="button" value="Add"/>

- 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.

### Add Users into Netgroup net-stage

Find

Available	Prospective
<input type="checkbox"/> User login	<input type="checkbox"/> User login
<input type="checkbox"/> admin	<input checked="" type="checkbox"/> bguster
<input type="checkbox"/> jdoe	<input checked="" type="checkbox"/> bjensen
<input type="checkbox"/> jsmith	<input checked="" type="checkbox"/> jjones
<input type="checkbox"/> jtyler	<input checked="" type="checkbox"/> jrockford
<input type="checkbox"/> ljenkins	
<input type="checkbox"/> mloy	

>> <<

Add Cancel

- Click the **Add** button.

### 14.3.2.2. 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, either use the option multiple times or use a comma-separated list inside a set of curly braces (for example, `--option={val1,val2,val3}`). For example, this sets two users and two hosts with the other configuration:

```
[root@server ~]# ipa netgroup-add-member --users=jsmith --users=bjensen
--groups=ITAdmin --hosts=host1.example.com --hosts=host2.example.com --
hostgroups=EngDev --netgroups=nisgroup2 example-group
```

## 14.4. 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 in the 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**:

```
[root@server ~]# 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.

## 14.5. 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.

### 14.5.1. 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 14.3, “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, add the users and hosts into IdM groups or other netgroups, and then add those groups as members to the netgroup.

### 14.5.2. 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 port mapper and Directory Server service:

```
[root@server ~]# systemctl start rpcbind.service
[root@server ~]# systemctl restart dirsrv.target
```

### 14.5.3. 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.

#### 14.5.3.1. 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 passw0rd1|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.

### 14.5.3.2. 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
```

### 14.5.3.3. 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.

### 14.5.3.4. 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 14.1, “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
```

### 14.5.3.5. 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
```

#### 14.5.4. 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 salted SHA (SSHA) or SSHA-256. If the 389 Directory Server password hash is not changed, then NIS users cannot authenticate to the IdM domain. The **kinit** command is not affected by the server's password hashing configuration.

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 15. Managing DNS

An Identity Management server can be installed without integrated DNS services so that it uses an external DNS service, as described in [Section 3.2.1, “Basic Interactive Installation”](#), or with DNS configured, as described in [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#). DNS services can also be installed into an already existing IdM server that does not have DNS configured, as described in [Section 15.1, “Installing DNS Services Into an Existing Server”](#).

If the DNS service is configured within the domain, IdM offers the administrator a significant amount of flexibility and control over DNS settings. For example, DNS entries for the domain, such as host entries, locations, or records, can be managed using native IdM tools, and clients can update their own DNS records dynamically.

Most documentation material and tutorials available for BIND version 9.9 are also applicable to IdM DNS, because majority of configuration options work in the same way in BIND and IdM. This chapter mostly focuses on notable differences between BIND and IdM.

### 15.1. Installing DNS Services Into an Existing Server

It is possible to install DNS services into an IdM server that was originally installed without them. To do this, make sure the *ipa-server-dns* package is installed, and then use the **ipa-dns-install** utility.

Configuring DNS services using **ipa-dns-install** follows the same principles as installing DNS using the **ipa-server-install --setup-dns** command, as described in [Section 3.2.4, “Configuring DNS Services within the IdM Domain”](#).

For more information about **ipa-dns-install**, see the `ipa-dns-install(1)` man page.

### 15.2. BIND in Identity Management

IdM integrates BIND DNS server version 9.9 with an LDAP database used for data replication and with Kerberos for DNS update signing using the GSS-TSIG protocol [5]. This enables convenient DNS management using IdM tools and at the same time increases resiliency because IdM-integrated DNS servers support multi-master operations, allowing all IdM-integrated DNS servers to accept DNS updates from clients without single point of failure.

The default IdM DNS configuration is suitable for internal networks that are not accessible from the public Internet. If the IdM DNS server is accessible from the public Internet, Red Hat recommends to apply the usual hardening applicable to the BIND service, described in the [Red Hat Enterprise Linux Networking Guide](#).



#### Note

It is not possible to run BIND integrated with IdM inside **chroot** environment.

BIND integrated with IdM communicates with the Directory Server using the **bind-dyndb-ldap** plug-in. IdM creates a **dynamic-db** configuration section in the `/etc/named.conf` file for the BIND service, which configures the **bind-dyndb-ldap** plug-in for the BIND **named-pkcs11** service.

The most notable difference between standard BIND and IdM DNS is that IdM stores all DNS information as LDAP entries. Every domain name is represented as LDAP entry, and every resource record is stored as an LDAP attribute of the LDAP entry. For example, the following **client1.example.com.** domain name contains three A records and one AAAA record:

```
dn:
idnsname=client1,idnsname=example.com.,cn=dns,dc=idm,dc=example,dc=com
objectclass: top
objectclass: idnsrecord
idnsname: client1
Arecord: 192.0.2.1
Arecord: 192.0.2.2
Arecord: 192.0.2.3
AAAArecord: 2001:DB8::ABCD
```



### Important

To edit DNS data or BIND configuration, always use the IdM tools described in this chapter.

## 15.3. Supported DNS Zone Types

IdM supports two DNS zone types: *master* and *forward* zones.



### Note

This guide uses the BIND terminology for zone types which is different from the terminology used for Microsoft Windows DNS. Master zones in BIND serve the same purpose as *forward lookup zones* and *reverse lookup zones* in Microsoft Windows DNS. Forward zones in BIND serve the same purpose as *conditional forwarders* in Microsoft Windows DNS.

### Master DNS zones

Master DNS zones contain authoritative DNS data and can accept dynamic DNS updates. This behavior is equivalent to the **type master** setting in standard BIND configuration. Master zones are managed using the **ipa dnszone-\*** commands.

In compliance with standard DNS rules, every master zone must contain SOA and NS records. IdM generates these records automatically when the DNS zone is created, but the NS records must be manually copied to the parent zone to create proper delegation.

In accordance with standard BIND behavior, forwarding configuration specified for master zones only affects queries for names for which the server is not authoritative.

### Example 15.1. Example Scenario for DNS Forwarding

The IdM server contains the **test.example.** master zone. This zone contains an NS delegation record for the **sub.test.example.** name. In addition, the **test.example.** zone is configured with the **192.0.2.254** forwarder IP address.

A client querying the name **nonexistent.test.example.** receives the **NXDomain** answer, and no forwarding occurs because the IdM server is authoritative for this name.

On the other hand, querying for the **sub.test.example.** name is forwarded to the configured forwarder **192.0.2.254** because the IdM server is not authoritative for this name.

### Forward DNS zones

Forward DNS zones do not contain any authoritative data. All queries for names belonging to a forward DNS zone are forwarded to a specified forwarder. This behavior is equivalent to the **type forward** setting in standard BIND configuration. Forward zones are managed using the **ipa dnsforwardzone-\*** commands.

## 15.4. DNS Configuration Priorities

Many DNS configuration options can be configured on three different levels.

### Zone-specific configuration

The level of configuration specific for a particular zone defined in IdM has the highest priority. Zone-specific configuration is managed using the **ipa dnszone-\*** and **ipa dnsforwardzone-\*** commands.

### Global DNS configuration

If no zone-specific configuration is defined, IdM uses global DNS configuration stored in LDAP. Global DNS configuration is managed using the **ipa dnsconfig-\*** commands. Settings defined in global DNS configuration are applied to all IdM DNS servers.

### Configuration in `/etc/named.conf`

Configuration defined in the `/etc/named.conf` file on each IdM DNS server has the lowest priority. It is specific for each server and must be edited manually.

The `/etc/named.conf` file is usually only used to specify DNS forwarding to a local DNS cache; other options are managed using the commands for zone-specific and global DNS configuration mentioned above.

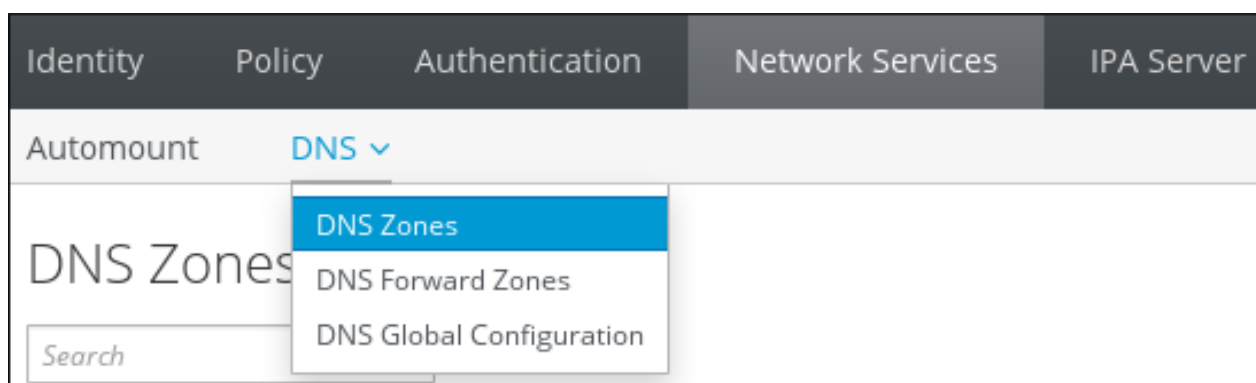
DNS options can be configured on multiple levels at once. In such cases, configuration with the highest priority takes precedence over configuration defined at lower levels.

## 15.5. Managing Master DNS Zones

### 15.5.1. Adding and Removing Master DNS Zones

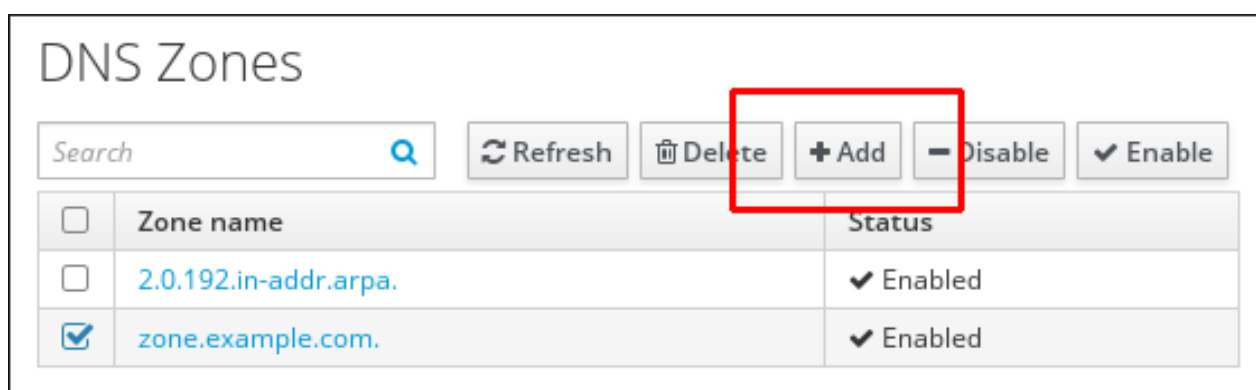
#### Adding Master DNS Zones in the Web UI

1. Open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



**Figure 15.1. Managing DNS Master Zones**

2. To add a new master zone, click **Add** at the top of the list of all zones.



**Figure 15.2. Adding a Master DNS Zone**

3. Provide the zone name, and click **Add**.

A screenshot of the 'Add DNS Zone' dialog box. It has a title bar with a close button. Inside, there are two radio buttons: 'Zone name' (selected) and 'Reverse zone'. The 'Zone name' radio button is followed by a text input field containing 'zone.example.com.'. Below the 'Reverse zone' radio button is a text input field for 'IP network'. At the bottom, there is a legend '\* Required field' and four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

**Figure 15.3. Entering a New Master Zone**

## Adding Master DNS Zones from the Command Line

The **ipa dnszone-add** command adds a new zone to the DNS domain. Adding a new zone requires you to specify the name of the new subdomain. You can pass the subdomain name directly with the command:

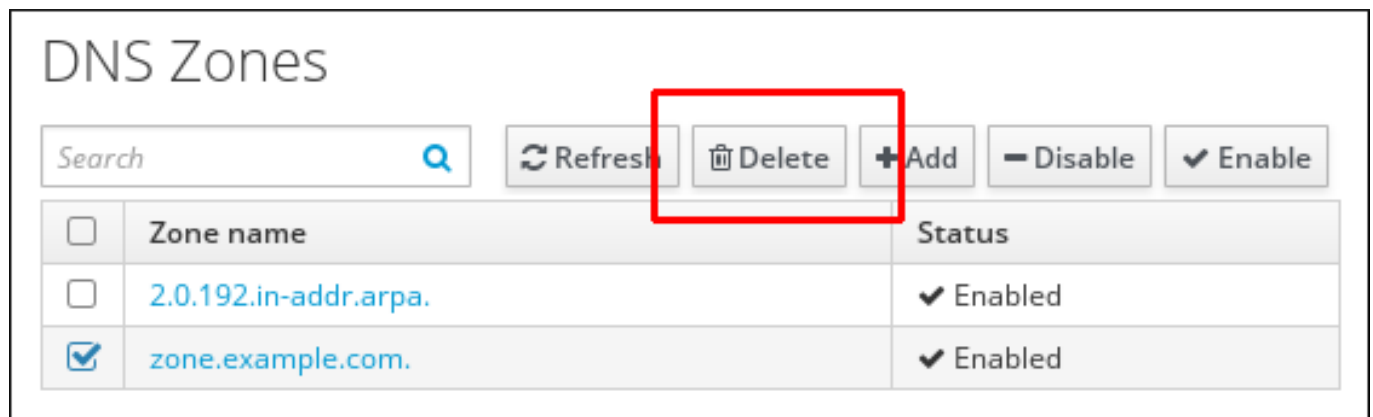
```
$ ipa dnszone-add newserver.example.com
```

If you do not pass the name to **ipa dnszone-add**, the script prompts for it automatically.

The **ipa dnszone-add** command also accepts various command-line options. For a complete list of these options, run the **ipa dnszone-add --help** command.

## Removing Master DNS Zones

To remove a master DNS zone in the web UI, in the list of all zones, select the check box by the zone name and click **Delete**.



**Figure 15.4. Removing a Master DNS Zone**

To remove a master DNS zone from the command line, use the **ipa dnszone-del** command. For example:

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

## 15.5.2. Adding Additional Configuration for Master DNS Zones

IdM creates a new zone with certain default configuration, such as the refresh periods, transfer settings, or cache settings.

### DNS Zone Configuration Attributes

The available zone settings are listed in [Table 15.1, “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.

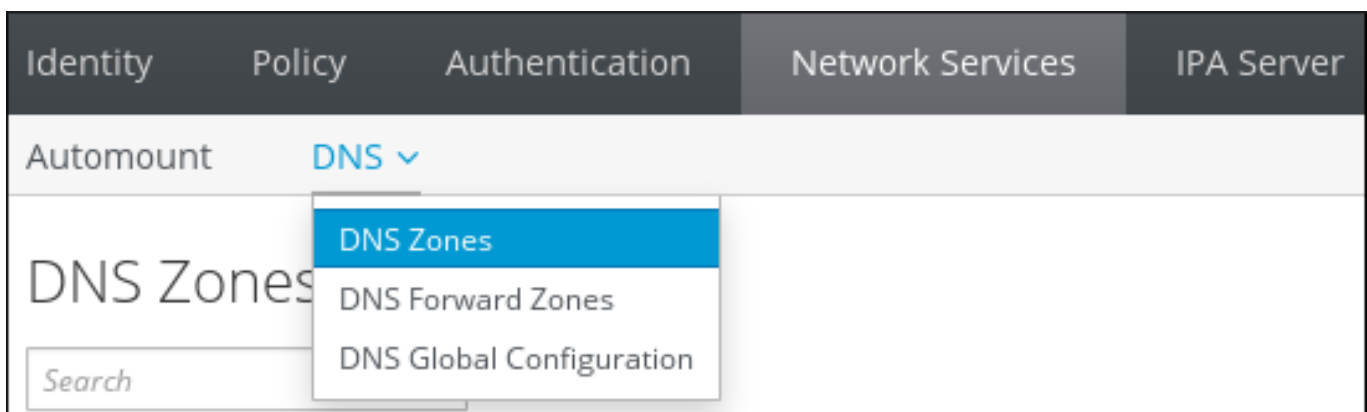
**Table 15.1. Zone Attributes**

Attribute	Command-Line Option	Description
Authoritative name server	--name-server	<p>Sets the domain name of the master DNS name server, also known as SOA MNAME.</p> <p>By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using <b>--name-server</b> is ignored.</p>
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 serial number in the SOA record. Note that IdM sets the version number automatically and users are not expected to modify it.
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 time-to-live (TTL) value in seconds for negative caching according to <a href="#">RFC 2308</a> .
SOA time to live	--ttl	Sets TTL in seconds for records at zone apex. In zone <b>example.com</b> , for instance, all records (A, NS, or SOA) under name <b>example.com</b> are configured, but no other domain names, like <b>test.example.com</b> , are affected.
BIND update policy	--update-policy	<p>Sets the permissions allowed to clients in the DNS zone.</p> <p>See <a href="#">Dynamic Update Policies in the BIND 9 Administrator Reference Manual</a> for more information on update policy syntax.</p>
Dynamic update	--dynamic-update=TRUE FALSE	Enables dynamic updates to DNS records for clients. Note that if this is set to false, IdM client machines will not be able to add or update their IP address. See <a href="#">Section 15.6.1, “Enabling Dynamic DNS Updates”</a> for more information.
Allow transfer	--allow-transfer= <i>string</i>	<p>Gives a semi-colon-separated list of IP addresses or network names which are allowed to transfer the given zone.</p> <p>Zone transfers are disabled by default. The default <b>--allow-transfer</b> value is <b>none</b>.</p>

Attribute	Command-Line Option	Description
Allow query	--allow-query	Gives a semi-colon-separated list of IP addresses or network names which are allowed to issue DNS queries.
Allow PTR sync	--allow-sync-ptr=1 0	Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.
Zone forwarders	--forwarder= <i>IP_addresses</i>	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	--forward-policy= <i>none only first</i>	Specifies the forward policy. For information about the supported policies, see <a href="#">Section 15.7, “Forward Policies”</a>

## Editing the Zone Configuration in the Web UI

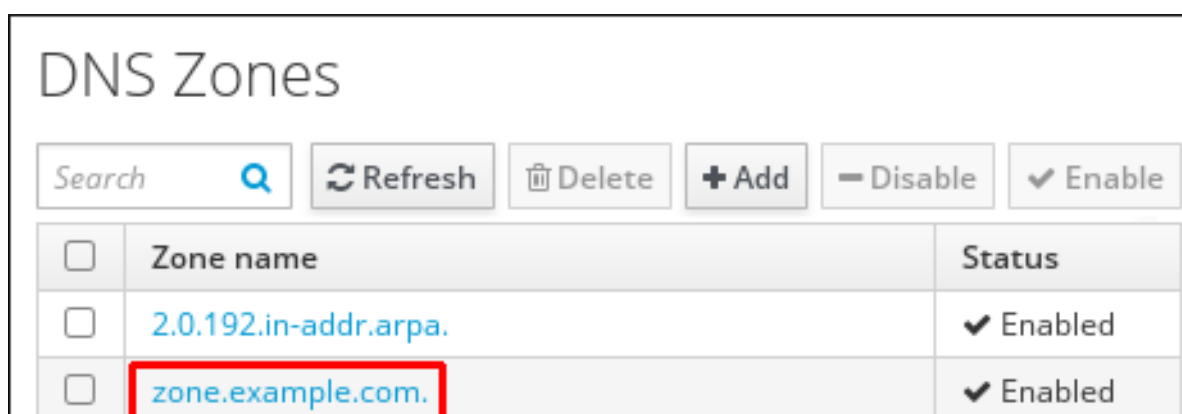
To manage DNS master zones from the web UI, open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



**Figure 15.5. DNS Master Zones Management**

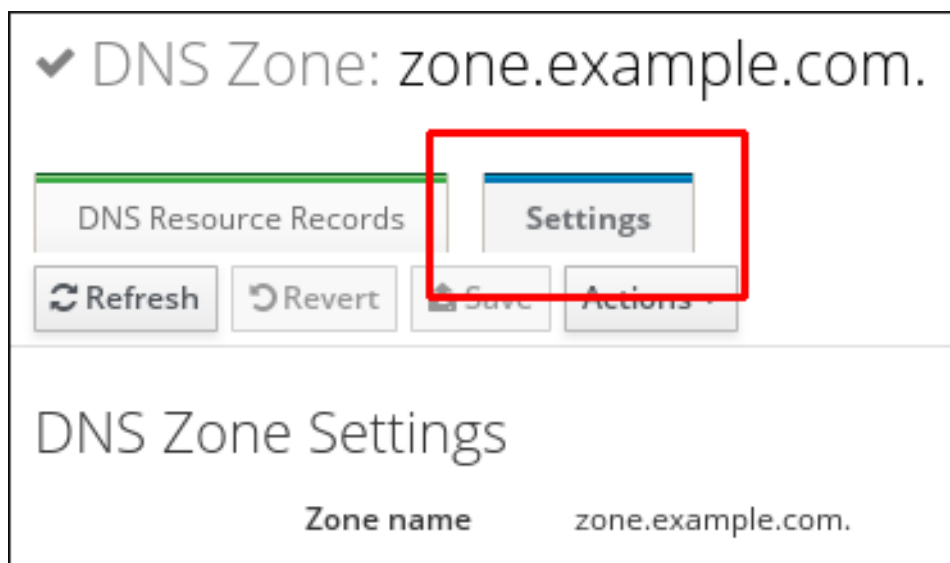
To edit an existing master zone in the **DNS Zones** section:

1. Click on the zone name in the list of all zones to open the DNS zone page.



**Figure 15.6. Editing a Master Zone**

2. Click **Settings**, and then change the zone configuration as required.

**Figure 15.7. The Settings Tab in the Master Zone Edit Page**

For information about the available settings, see [Table 15.1, “Zone Attributes”](#).

3. Click **Save** to confirm the new configuration.

### Editing the Zone Configuration from the Command Line

To modify an existing master DNS zone from the command line, use the **ipa dnszone-mod** command. For information about the available settings, see [Table 15.1, “Zone Attributes”](#).

If an attribute does not exist in the DNS zone entry, the **ipa dnszone-mod** command adds the attribute. If the attribute exists, the command overwrites the current value with the specified value.

For detailed information about **ipa dnszone-mod** and its options, run the **ipa dnszone-mod --help** command.

#### 15.5.3. Enabling Zone Transfers

Name servers maintain authoritative data for the zones; changes made to the zones must be sent to and distributed among the name servers for the DNS domain. A *zone transfer* copies all resource records from one name server to another.

IdM supports zone transfers according to the [RFC 5936](#) (AXFR) and [RFC 1995](#) (IXFR) standards.





## Important

The IdM-integrated DNS is multi-master. SOA serial numbers in IdM zones are not synchronized between IdM servers. For this reason, configure DNS slave servers to only use one IdM master server. This prevents zone transfer failures caused by non-synchronized SOA serial numbers.

### Enabling Zone Transfers in the UI

Open the DNS zone page, as described in [Section 15.5.2, “Editing the Zone Configuration in the Web UI”](#), and switch to the **Settings** tab.

Under **Allow transfer**, specify the name servers to which the zone records will be transferred.

Allow transfer	192.0.2.1	Undo
	198.51.100.1	Undo
	203.0.113.1	Undo
Add		Undo All

**Figure 15.8. Enabling Zone Transfers**

Click **Save** at the top of the DNS zone page to confirm the new configuration.

### Enabling Zone Transfers from the Command Line

To enable zone transfers from the command line, add the **--allow-transfer** option to the **ipa dnszone-mod** command. Specify the list of name servers to which the zone records will be transferred using **--allow-transfer**. For example:

```
[user@server ~]$ ipa dnszone-mod --allow-
transfer=192.0.2.1;198.51.100.1;203.0.113.1 example.com
```

Once zone transfers are enabled in the **bind** service, IdM DNS zones can be transferred, by name, by clients such as the **dig** utility:

```
[root@server ~]# dig @ipa-server zone_name AXFR
```

### 15.5.4. Adding Records to DNS Zones

IdM supports many different record types. The following four are used most frequently:

#### A

This is a basic map for a host name and an ordinary IPv4 address. The **Record Name** of an A record is a host name, such as `www`. The **IP Address** value of an A record is a standard IPv4 address, such as `192.0.2.1`.

For more information about A records, see [RFC 1035](#).

## AAAA

This is a basic map for a host name and an IPv6 address. The **Record Name** of an AAAA record is a host name, such as `www`. The **IP Address** value is a standard hexadecimal IPv6 address, such as `2001:DB8::1111`.

For more information about AAAA records, see [RFC 3596](#).

## SRV

*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 server which manages it.

The **Record Name** of an SRV record has the format `_service._protocol`, such as `_ldap._tcp`. The configuration options for SRV records include priority, weight, port number, and host name for the target service.

For more information about SRV records, see [RFC 2782](#).

## PTR

A pointer record type (PTR) record adds a reverse DNS record, which maps an IP address to a domain name.



### Note

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 network address `192.0.2.0/24`, the reverse zone is `2.0.192.in-addr.arpa`.

The **Record Name** of a PTR record must be in the standard format specified in [RFC 1035](#), extended in [RFC 2317](#), and [RFC 3596](#). The **Hostname** value must be a canonical host name of the host for which you want to create the record. For more information, see [Example 15.6, "PTR Record"](#).



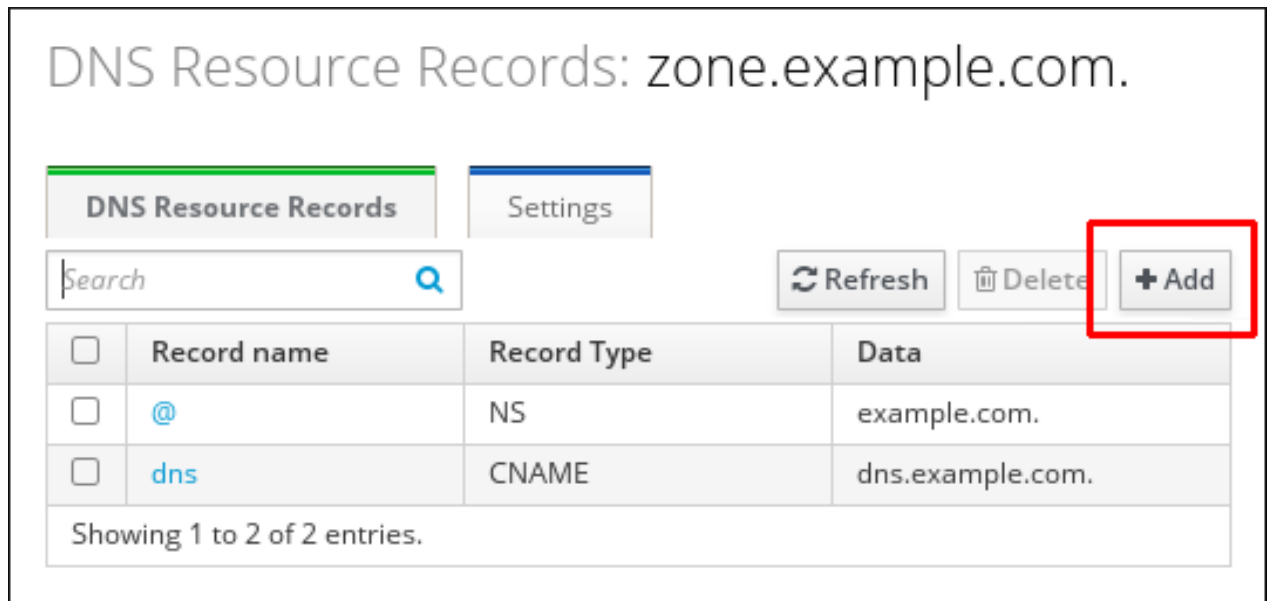
### Note

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](#).

When adding DNS resource records, note that many of the records require different data. For example, a CNAME record requires a host name, while an A record requires an IP address. In the web UI, the fields in the form for adding a new record are updated automatically to reflect what data is required for the currently selected type of record.

## Adding DNS Resource Records from the Web UI

1. Open the DNS zone page, as described in [Section 15.5.2, “Editing the Zone Configuration in the Web UI”](#).
2. In the **DNS Resource Records** section, click **Add** to add a new record.



**Figure 15.9. Adding a New DNS Resource Record**

3. Select the type of record to create and fill out the other fields as required.

Add DNS Resource Record

Record name \*

dns

Record Type

CNAME

Hostname \*

dns.example.com.

\* Required field

Add

Add and Add Another

Add and Edit

Cancel

**Figure 15.10. Defining a New DNS Resource Record**

4. Click **Add** to confirm the new record.

## Adding DNS Resource Records from the Command Line

To add a DNS resource record of any type from the command line, use the **ipa dnsrecord-add** command. The command follows this syntax:

```
$ 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 15.2, “Common ipa dnsrecord-add Options”](#) lists options for the most common resource record types: A (IPv4), AAAA (IPv6), SRV, and PTR. Lists of entries can be set by using the option multiple times with the same command invocation or, in Bash, by listing the options in a comma-separated list inside curly braces, such as **--option={val1,val2,val3}**.

For more detailed information on how to use **ipa dnsrecord-add** and which DNS record types are supported by IdM, run the **ipa dnsrecord-add --help** command.

**Table 15.2. Common ipa dnsrecord-add Options**

General Record Options	
Option	Description
--ttl= <i>number</i>	Sets the time to live for the record.
--structured	Parses the raw DNS records and returns them in a structured format.
"A" Record Options	
Option	Description
--a-rec= <i>ARECORD</i>	Passes a list of A records.
--a-ip-address= <i>string</i>	Gives the IP address for the record.
"AAAA" Record Options	
Option	Description
--aaaa-rec= <i>AAAAARECORD</i>	Passes a list of AAAA (IPv6) records.
--aaaa-ip-address= <i>string</i>	Gives the IPv6 address for the record.
"PTR" Record Options	
Option	Description
--ptr-rec= <i>PTRRECORD</i>	Passes a list of PTR records.
--ptr-hostname= <i>string</i>	Gives the hostname for the record.
"SRV" Record Options	
Option	Description
--srv-rec= <i>SRVRECORD</i>	Passes a list of SRV records.
--srv-priority= <i>number</i>	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.
--srv-weight= <i>number</i>	Sets the weight of the record. This helps determine the order of SRV records with the same priority. The set weights should add up to 100, representing the probability (in percentages) that a particular record is used.

**"SRV" Record Options**

<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.

**15.5.5. Examples of Adding or Modifying DNS Resource Records from the Command Line****Example 15.2. Adding a IPv4 Record**

The following example creates the record **www.example.com** with the IP address **192.0.2.123**.

```
$ ipa dnsrecord-add example.com www --a-rec 192.0.2.123
```

**Example 15.3. Modifying a IPv4 Record**

When creating a record, the option to specify the A record value is **--a-record**. However, when modifying an A record, the **--a-record** option is used to specify the current value for the A record. The new value is set with the **-a--ip-address** option.

```
$ ipa dnsrecord-mod example.com www --a-rec 192.0.2.123 --a-ip-address 192.0.2.1
```

**Example 15.4. Adding an IPv6 Record**

The following example creates the record **www.example.com** with the IP address **2001:db8::1231:5675**.

```
$ ipa dnsrecord-add example.com www --aaaa-rec 2001:db8::1231:5675
```

**Example 15.5. Adding an SRV Record**

The **recordName** value identifies the service type and the connection protocol, in the format **\_service.\_protocol**. The **record** information has the format "**priority weight port target**".

For example:

```
[root@server ~]# ipa dnsrecord-add server.example.com _ldap._tcp --
srv-rec="0 51 389 server1.example.com."
[root@server ~]# ipa dnsrecord-add server.example.com _ldap._tcp --
srv-rec="1 49 389 server2.example.com."
```

The weight values add up to 100, representing the probability (in percentages) that a particular record is used.

### Example 15.6. PTR Record

When adding the reverse DNS record, the zone name used with the **ipa dnsrecord-add** command is reverse, compared to the usage for adding other DNS records:

```
$ ipa dnsrecord-add reverseNetworkIpAddress hostIpAddress --ptr-rec
FQDN
```

Typically, *hostIpAddress* is the last octet of IP address in a given network.

For example, this adds a PTR record for **server4.example.com** with IPv4 address 192.0.2.4:

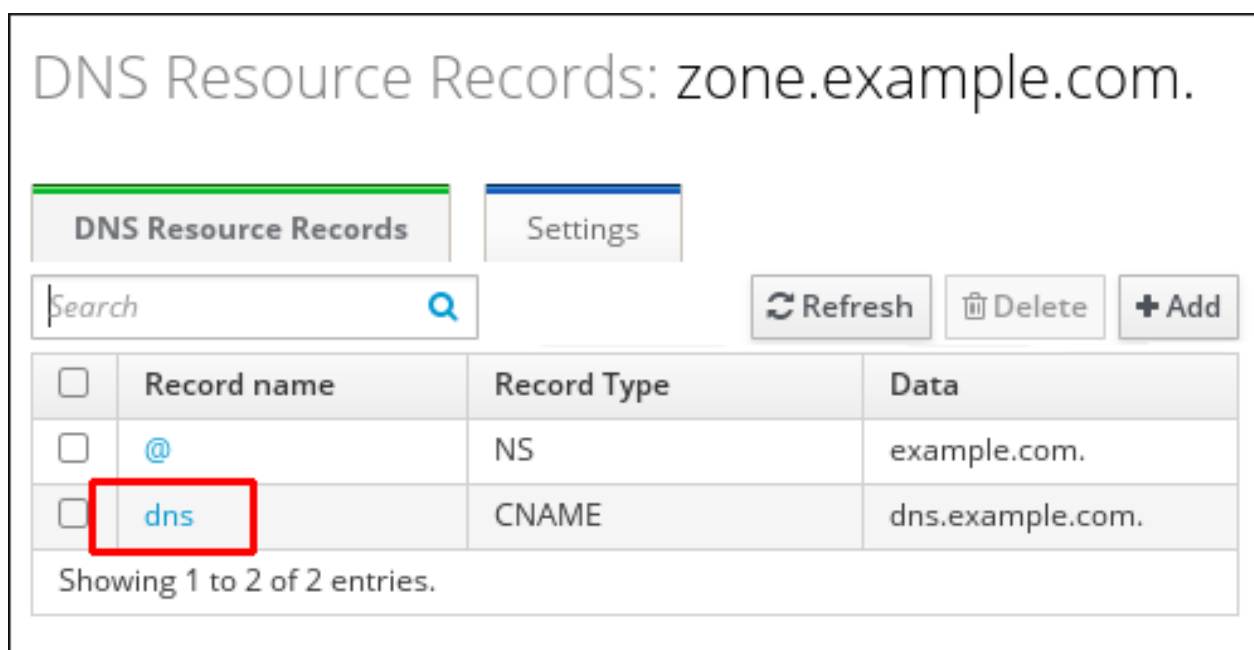
```
$ ipa dnsrecord-add 2.0.192.in-addr.arpa 4 --ptr-rec
server4.example.com.
```

## 15.5.6. Deleting Records from DNS Zones

### Deleting Records in the Web UI

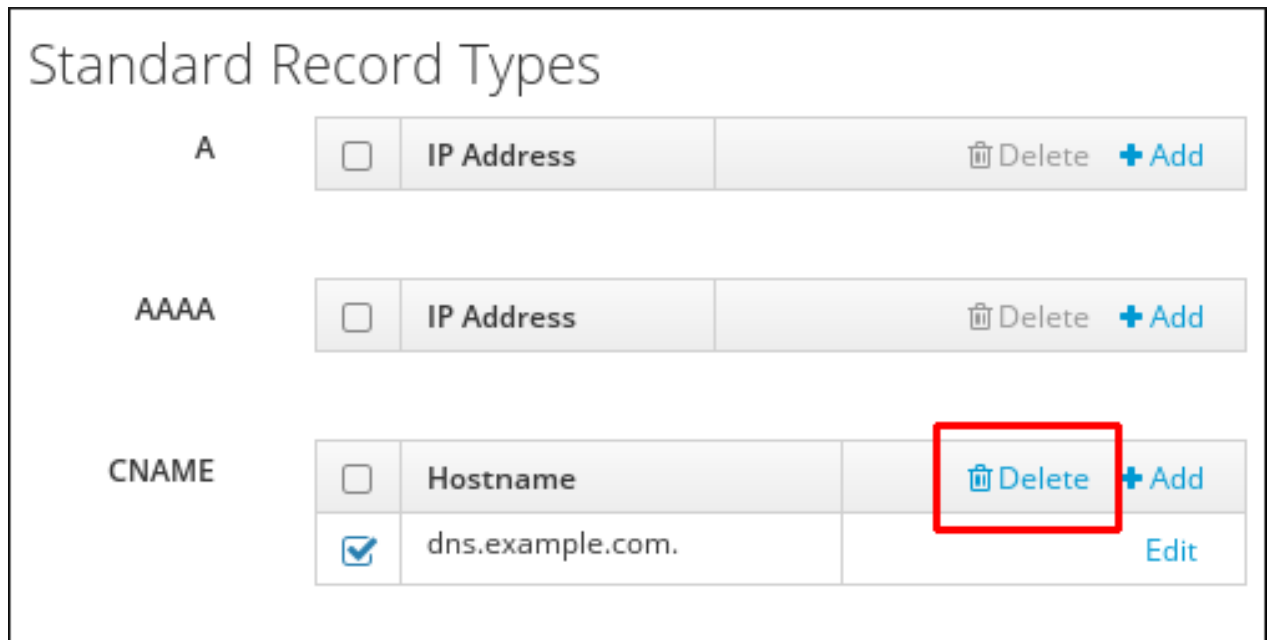
To delete only a specific record type from the resource record:

1. Open the DNS zone page, as described in [Section 15.5.2, “Editing the Zone Configuration in the Web UI”](#).
2. In the **DNS Resource Records** section, click the name of the resource record.



**Figure 15.11. Selecting a DNS Resource Record**

3. Select the check box by the name of the record type to delete.

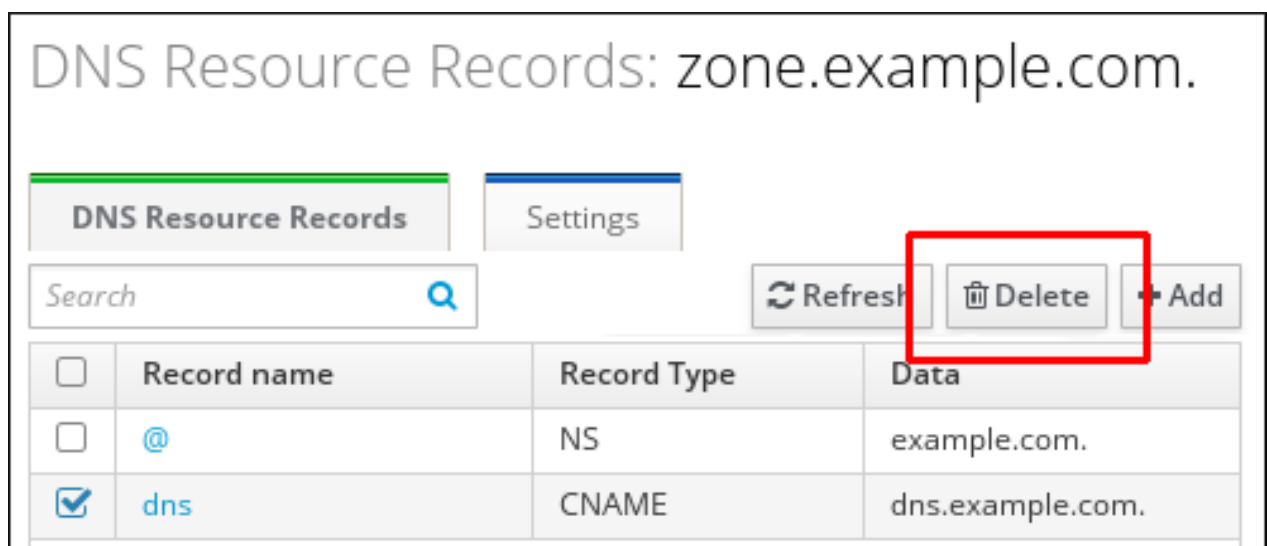


**Figure 15.12. Deleting a DNS Resource Record**

After this, only the selected record type is deleted; the other configuration is left intact.

To delete all records for the resource in the zone:

1. Open the DNS zone page, as described in [Section 15.5.2, "Editing the Zone Configuration in the Web UI"](#).
2. In the **DNS Resource Records** section, select the check box by the name of the resource record to delete, and then click **Delete** at the top of the list of zone records.



**Figure 15.13. Deleting an Entire Resource Record**

After this, the entire resource record is deleted.

## Deleting Records from the Command Line

To remove records from a zone, use the **ipa dnsrecord-del** command and add the **-recordType-rec** option together with the record value.

For example, to remove the A type record:

```
$ ipa dnsrecord-del example.com www --a-rec 192.0.2.1
```

If you run **ipa dnsrecord-del** without any options, the command prompts for information about the record to delete. Note that passing the **--del-all** option with the command removes all associated records for the zone.

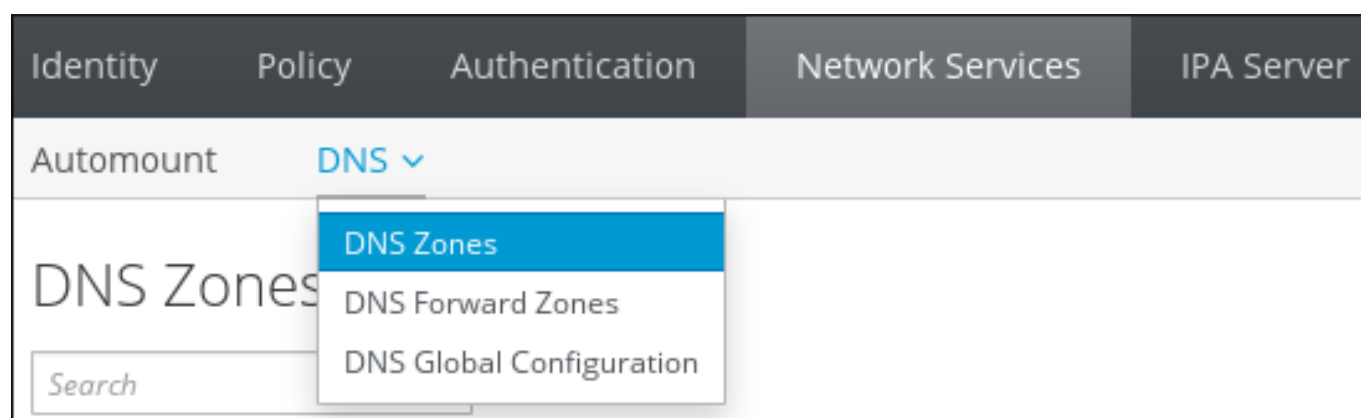
For detailed information on how to use **ipa dnsrecord-del** and a complete list of options accepted by the command, run the **ipa dnsrecord-del --help** command.

### 15.5.7. Disabling and Enabling Zones

IdM allows the administrator to disable and enable DNS zones. While deleting a DNS zone, described in [Section 15.5.1, “Removing Master DNS Zones”](#), completely removes the zone entry and all the associated configuration, disabling the zone removes it from activity without permanently removing the zone from IdM. A disabled zone can also be enabled again.

#### Disabling and Enabling Zones in the Web UI

To manage DNS zones from the Web UI, open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



**Figure 15.14. Managing DNS Zones**

To disable a zone, select the check box next to the zone name and click **Disable**.





**Figure 15.15. Disabling a DNS Zone**

Similarly, to enable a disabled zone, select the check box next to the zone name and click **Enable**.

### Disabling and Enabling DNS Zones from the Command Line

To disable a DNS zone from the command line, use the **ipa dnszone-disable** command. For example:

```
[user@server ~]$ ipa dnszone-disable zone.example.com
-----
Disabled DNS zone "example.com"
-----
```

To re-enable a disabled zone, use the **ipa dnszone-enable** command.

## 15.6. Managing Dynamic DNS Updates

### 15.6.1. Enabling Dynamic DNS Updates

Dynamic DNS updates are disabled by default for new DNS zones in IdM. With dynamic updates disabled, the **ipa-client-install** script cannot add a DNS record pointing to the new client.



#### Note

Enabling dynamic updates can potentially pose a security risk. However, if enabling dynamic updates is acceptable in your environment, you can do it to make client installations easier.

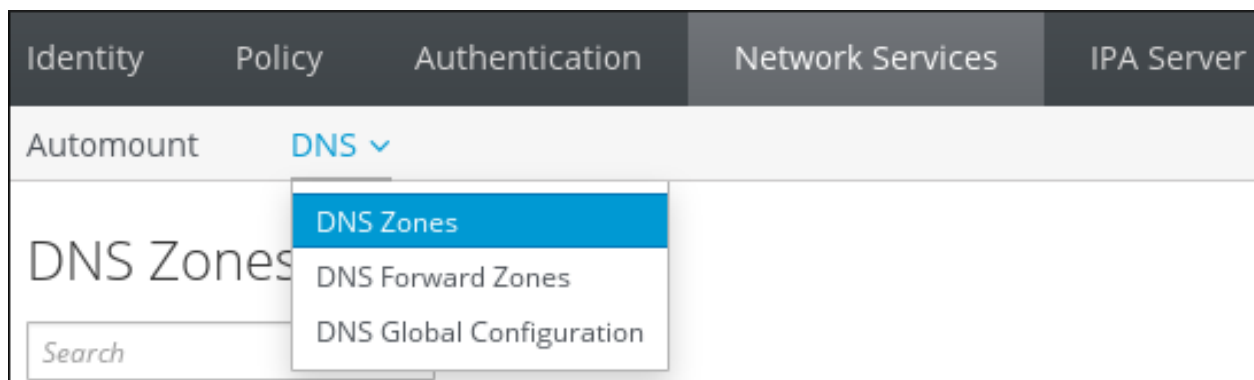
Enabling dynamic updates requires the following:

- ✧ The DNS zone must be configured to allow dynamic updates
- ✧ The local clients must be configured to send dynamic updates

#### 15.6.1.1. Configuring the DNS Zone to Allow Dynamic Updates

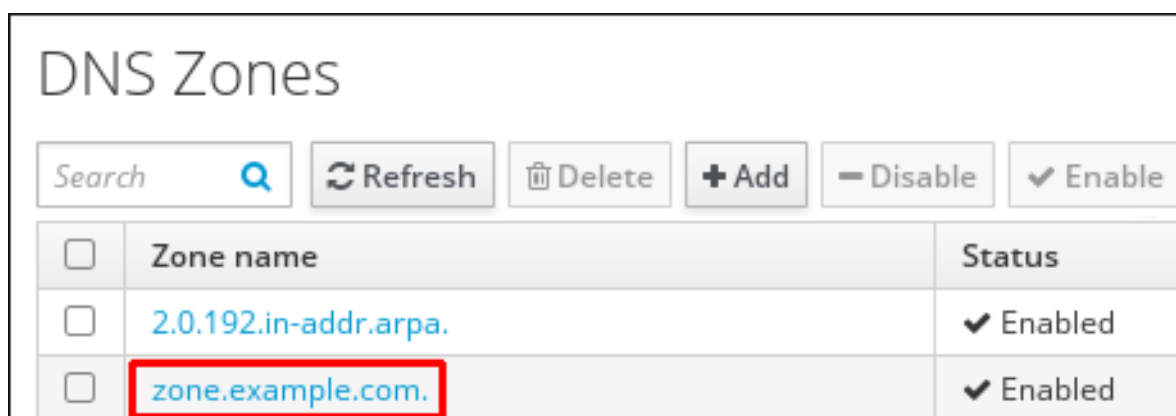
## Enabling Dynamic DNS Updates in the Web UI

1. Open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



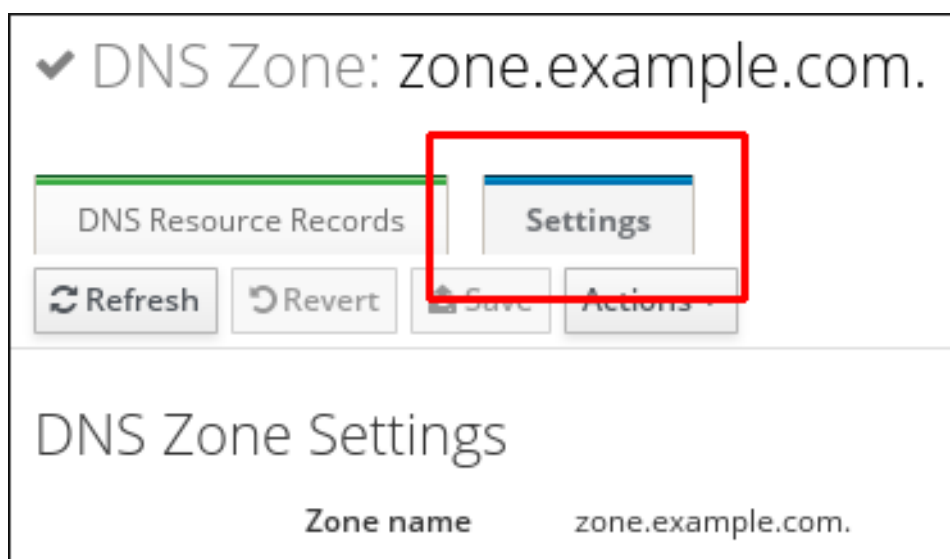
**Figure 15.16. DNS Zone Management**

2. Click on the zone name in the list of all zones to open the DNS zone page.



**Figure 15.17. Editing a Master Zone**

3. Click **Settings** to switch to the DNS zone settings tab.



**Figure 15.18. The Settings Tab in the Master Zone Edit Page**

4. Scroll down to the **Dynamic update** field, and set the value to **True**.

Time to live

Dynamic update ☒ True ☐ False

BIND update policy

grant EXAMPLE.COM krb5-self \* A; grant EXAMPLE.COM krb5-self \* AAAA; grant EXAMPLE.COM krb5-self \* SSHFP;

Undo

**Figure 15.19. Enabling Dynamic DNS Updates**

5. Click **Save** at the top of the page to confirm the new configuration.

### Enabling Dynamic DNS Updates from the Command Line

To allow dynamic updates to the DNS zones from the command line, use the **ipa dnszone-mod** command with the **--dynamic-update=TRUE** option. For example:

```
[user@server ~]$ ipa dnszone-mod server.example.com --dynamic-update=TRUE
```

#### 15.6.1.2. Configuring the Clients to Send Dynamic Updates

Clients are automatically set up to send DNS updates when they are enrolled in the domain, by using the **--enable-dns-updates** option with the **ipa-client-install** script.

```
[root@client ~]# ipa-client-install --enable-dns-updates
```

The DNS zone has a time-to-live value set for records within its SOA configuration. However, the time-to-live for the dynamic updates is managed on the local system by the System Security Service Daemon (SSSD). To change the time-to-live value for the dynamic updates, edit the SSSD file to set a value; the default is 1200 seconds.

1. Open the SSSD configuration file.

```
[root@server ~]# vim /etc/sss/sss.conf
```

2. Find the domain section for the IdM domain.

```
[domain/ipa.example.com]
```

3. If dynamic updates have not been enabled for the client, then set the **`dyndns_update`** value to true.

```
dyndns_updates = true
```

4. Add or edit the **`dyndns_ttl`** parameter to set the value, in seconds, for the update time-to-live.

```
dyndns_ttl = 2400
```

### 15.6.2. Synchronizing A/AAAA and PTR Records

A and AAAA records are configured separately from PTR records in reverse zones. Because these records are configured independently, it is possible for A/AAAA records to exist without corresponding PTR records, and vice versa.

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 15.6.1, “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.



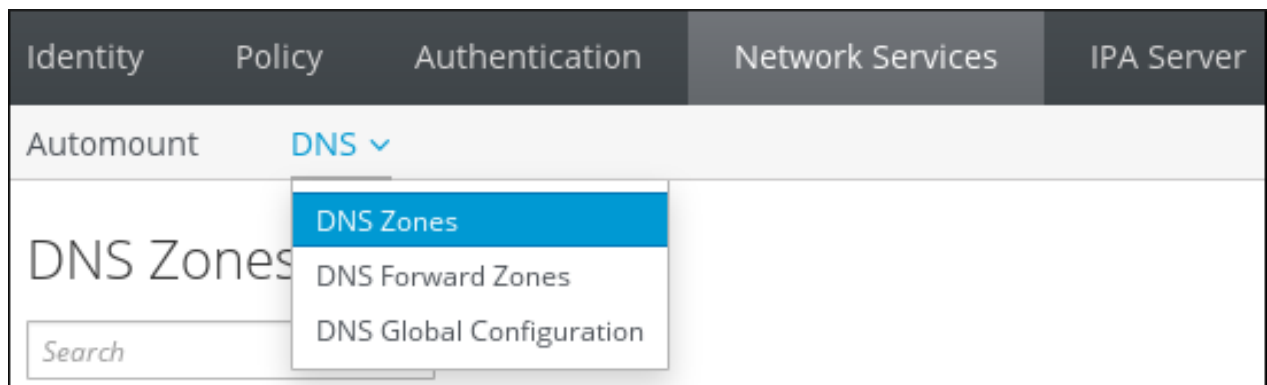
#### Warning

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.

### Configuring PTR Record Synchronization in the Web UI

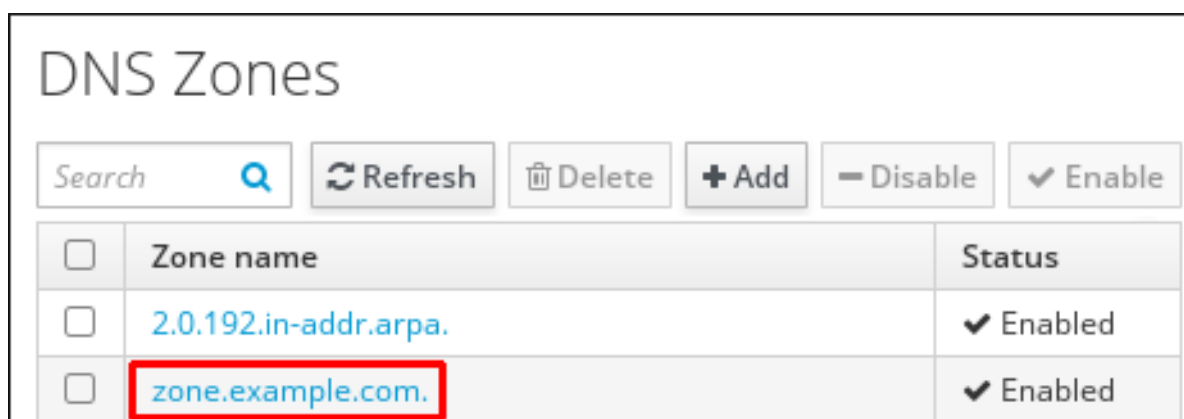
Note that PTR record synchronization must be configured on the zone where A or AAAA records are stored, not on the reverse DNS zone where PTR records are located.

1. Open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



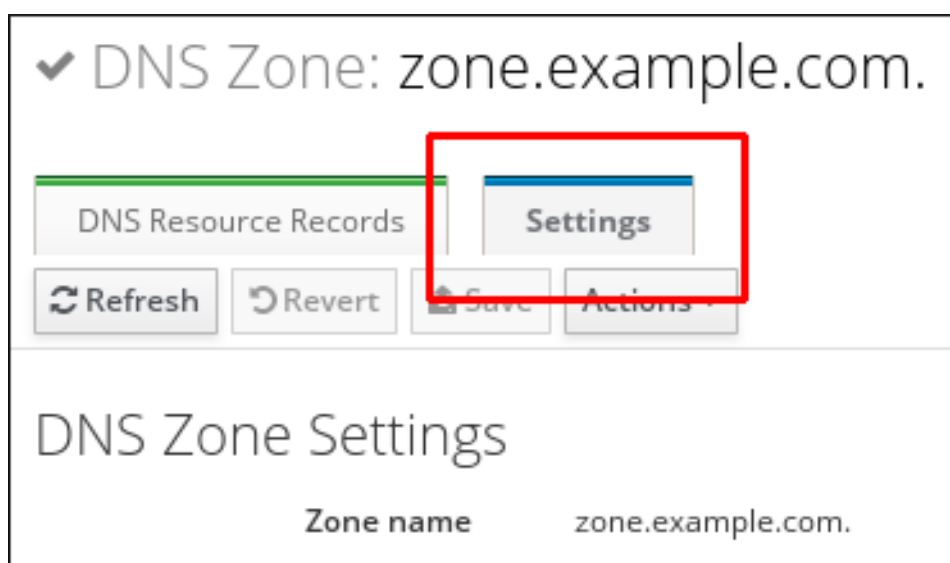
**Figure 15.20. DNS Zone Management**

- Click on the zone name in the list of all zones to open the DNS zone page.



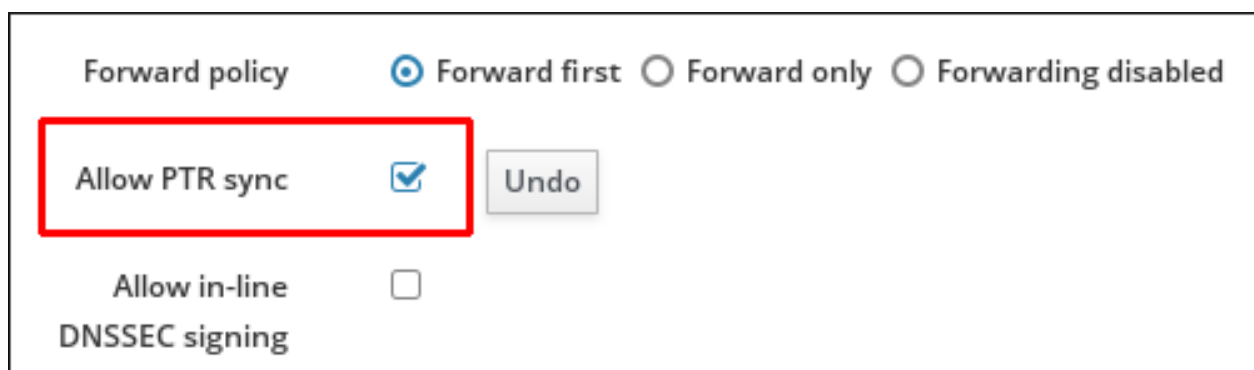
**Figure 15.21. Editing a DNS Zone**

- Click **Settings** to switch to the DNS zone settings tab.



**Figure 15.22. The Settings Tab in the Master Zone Edit Page**

- Select the **Allow PTR sync** check box.



Forward policy ☒ Forward first ☐ Forward only ☐ Forwarding disabled

Allow PTR sync ☒ Undo

Allow in-line DNSSEC signing ☐

**Figure 15.23. Enabling PTR Synchronization**

5. Click **Save** at the top of the page to confirm the new configuration.

### Configuring PTR Record Synchronization from the Command Line

Note that PTR record synchronization must be configured on the zone where A or AAAA records are stored, not on the reverse DNS zone where PTR records are located.

To configure a DNS zone to allow its forward and reverse entries to be synchronized automatically, set the **--allow-sync-ptr** option to **1** when the zone is created or when it is edited. For example, using the **ipa dnszone-mod** command when editing an existing zone:

```
[user@server ~]$ ipa dnszone-mod --allow-sync-ptr=1 server.example.com
```

The default **--allow-sync-ptr** value is **0**, which disables synchronization.

#### 15.6.3. Updating DNS Dynamic Update Policies

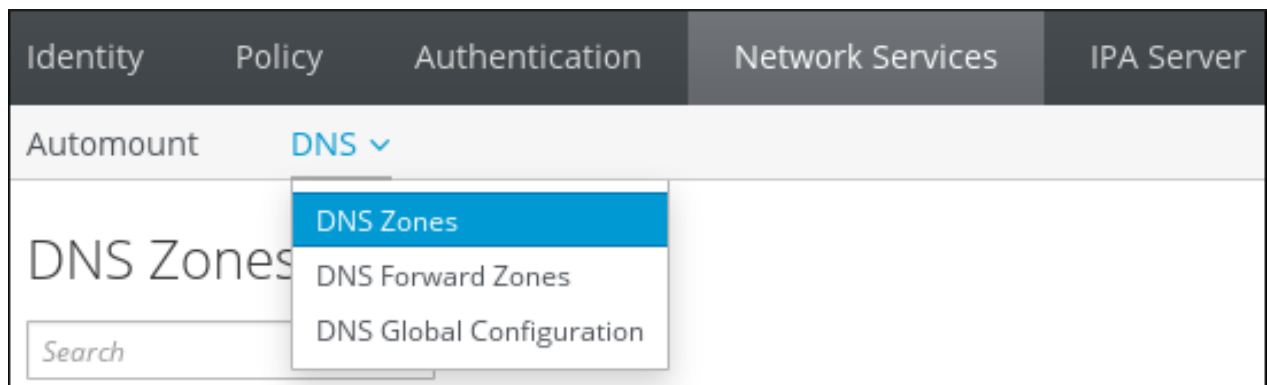
DNS domains maintained by IdM servers can accept a DNS dynamic update according to RFC 3007 [6].

The rules that determine which records can be modified by a specific client follow the same syntax as the **update-policy** statement in the **/etc/named.conf** file. For more information on dynamic update policies, see the [BIND 9 documentation](#).

Note that if dynamic DNS updates are disabled for the DNS zone, all DNS updates are declined without reflecting the dynamic update policy statement. For information on enabling dynamic DNS updates, see [Section 15.6.1, “Enabling Dynamic DNS Updates”](#).

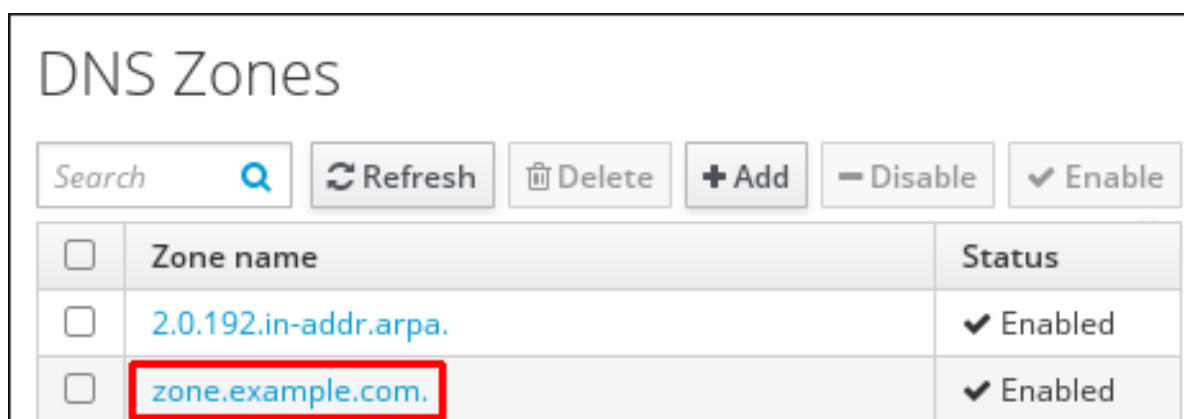
### Updating DNS Update Policies in the Web UI

1. Open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



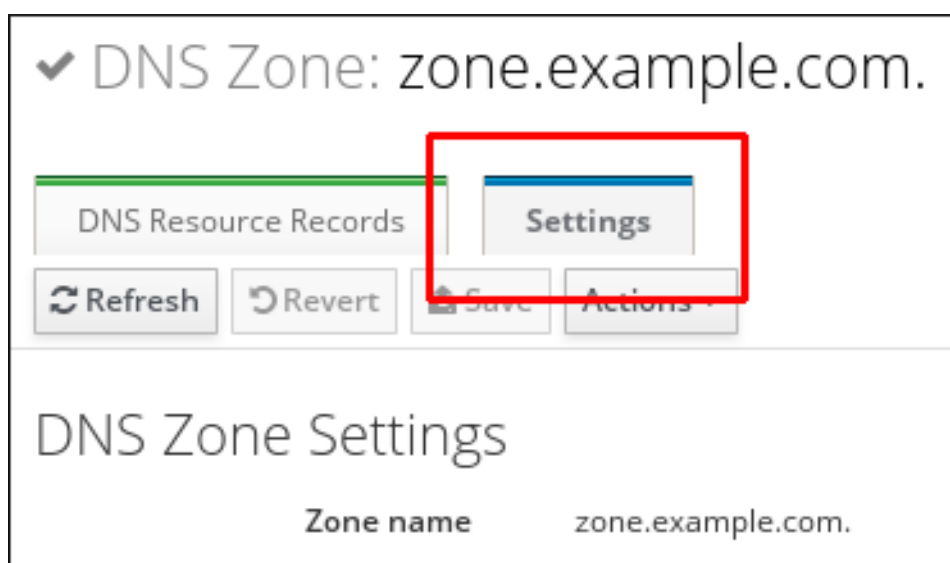
**Figure 15.24. DNS Zone Management**

- Click on the zone name in the list of all zones to open the DNS zone page.



**Figure 15.25. Editing a DNS Zone**

- Click **Settings** to switch to the DNS zone settings tab.



**Figure 15.26. The Settings Tab in the Master Zone Edit Page**

- Set the required update policies in a semi-colon separated list in the **BIND update policy** text box.

Dynamic update ☒ True ☐ False

BIND update policy

```
grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self
* AAAA; grant EXAMPLE.COM krb5-self * SSHFP;
```

Allow query

**Figure 15.27. DNS Update Policy Settings**

- Click **Save** at the top of the DNS zone page to confirm the new configuration.

### Updating DNS Update Policies from the Command Line

To set the DNS update policy from the command line, use the **--update-policy** option and add the access control rule in a statement after the option. For example:

```
$ ipa dnszone-mod --update-policy "grant EXAMPLE.COM krb5-self * A;
grant EXAMPLE.COM krb5-self * AAAA; grant EXAMPLE.COM krb5-self *
SSHFP;"
```

## 15.7. Managing DNS Forwarding

DNS forwarding affects how DNS queries are answered. By default, the BIND service integrated with IdM is configured to act as both an authoritative and recursive DNS server.

When a DNS client queries a name belonging to a DNS zone for which the IdM server is authoritative, BIND replies with data contained in the configured zone. Authoritative data always takes precedence over any other data.

When a DNS client queries a name for which the IdM server is not authoritative, BIND attempts to resolve the query using other DNS servers. If no forwarders are defined, BIND asks the root servers on the Internet and uses recursive resolution algorithm to answer the DNS query.

In some cases, it is not desirable to let BIND contact other DNS servers directly and perform the recursion based on data available on the Internet. These cases include:

- ✦ *Split DNS* configuration, also known as *DNS views* configuration, where DNS servers return different answers to different clients. Split DNS configuration is typical for environments where some DNS names are available inside the company network, but not from the outside.
- ✦ Configurations where a firewall restricts access to DNS on the Internet.
- ✦ Configurations with centralized filtering or logging on the DNS level.



- ✱ Configurations with forwarding to a local DNS cache, which helps optimize network traffic.

In such configurations, BIND does not use full recursion on the public Internet. Instead, it uses another DNS server, a so-called *forwarder*, to resolve the query. When BIND is configured to use a forwarder, queries and answers are forwarded back and forth between the IdM server and the forwarder, and the IdM server acts as the DNS cache for non-authoritative data.

## Forward Policies

IdM supports the *first* and *only* standard BIND forward policies, as well as the *none* IdM-specific forward policy.

### Forward first (default)

DNS queries are forwarded to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The forward first policy is the default policy. It is suitable for traffic optimization.

### Forward only

DNS queries are forwarded to the configured forwarder. If a query fails because of a server error or timeout, BIND returns an error to the client. The forward only policy is recommended for environments with split DNS configuration.

### None: Forwarding disabled

DNS queries are not forwarded. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.

## Forwarding Does Not Combine Data from IdM and Other DNS Servers

Forwarding cannot be used to combine data in IdM with data from other DNS servers. The BIND service does not forward queries to another server if the queried DNS name belongs to a zone for which the IdM server is authoritative. As a consequence, forwarding is not used when the client queries a name that does not exist in an IdM-managed zone.

### Example 15.7. Example Scenario

The IdM server is authoritative for the **test.example.** DNS zone. BIND is configured to forward queries to the DNS server with the **192.0.2.254** IP address.

When a client sends a query for the **nonexistent.test.example.** DNS name, BIND detects that the IdM server is authoritative for the **test.example.** zone and does not forward the query to the **192.0.2.254.** server. As a result, the DNS client receives the **NXDomain** answer, informing the user that the queried domain does not exist.

### 15.7.1. Configuring Global Forwarders

*Global forwarders* are DNS servers used for resolving all DNS queries for which an IdM server is not authoritative, as described in [Section 15.7, “Managing DNS Forwarding”](#).

The administrator can configure IP addresses and forward policies for global forwarding in the following two ways:

### Using the `ipa dnsconfig-mod` command or the IdM web UI

Configuration set using these native IdM tools is immediately applied to all IdM DNS servers. As explained in [Section 15.4, “DNS Configuration Priorities”](#), global DNS configuration has higher priority than local configuration defined in the `/etc/named.conf` files.

### By editing the `/etc/named.conf` file

Manually editing the `/etc/named.conf` on every IdM DNS server allows using a different global forwarder and policy on each of the servers. Note that the BIND service must be restarted after changing `/etc/named.conf`.

## Configuring Forwarders in the Web UI

To define the DNS global configuration in the IdM web UI:

1. Click the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Global Configuration** section.
2. To add a new global forwarder, click **Add** and enter the IP address. To define a new forward policy, select it from the list of available policies.

The screenshot displays the 'DNS Global Configuration' interface in the IdM web UI. At the top, there are navigation tabs: Identity, Policy, Authentication, Network Services (which is the active tab), and IPA Server. Under the 'Network Services' tab, a subtab for 'DNS' is selected. The main content area is titled 'DNS Global Configuration'. Below the title, there are three buttons: 'Refresh', 'Revert', and 'Save'. The 'Options' section contains a checkbox labeled 'Allow PTR sync'. The 'Global forwarders' section features two text input fields containing the IP addresses '192.0.2.253' and '192.0.2.254', each followed by an 'Undo' button. Below these fields are 'Add' and 'Undo All' buttons. The 'Forward policy' section at the bottom has three radio buttons: 'Forward first' (which is selected), 'Forward only', and 'Forwarding disabled'.

**Figure 15.28. Editing Global DNS Configuration in the Web UI**

3. Click **Save** to confirm the new configuration.

## Configuring Forwarders from the Command Line

To set a global list of forwarders from the command line, use the `ipa dnsconfig-mod` command. It edits the DNS global configuration by editing the LDAP data. The `ipa dnsconfig-mod` command and its options affect all IdM DNS servers at once and override any local configuration.

For example, to edit the list of global forwarders using `ipa dnsconfig-mod`:

```
[user@server ~]$ ipa dnsconfig-mod --forwarder=192.0.2.254
Global forwarders: 192.0.2.254
```

### 15.7.2. Configuring Forward Zones

Forward zones do not contain any authoritative data and instruct the name server to only forward queries for names belonging into a particular zone to a configured forwarder.



#### Important

Do not use forward zones unless absolutely required. Limit their use to overriding global forwarding configuration. In most cases, **it is sufficient to only configure global forwarding**, described in [Section 15.7.1, “Configuring Global Forwarders”](#), and forward zones are not necessary.

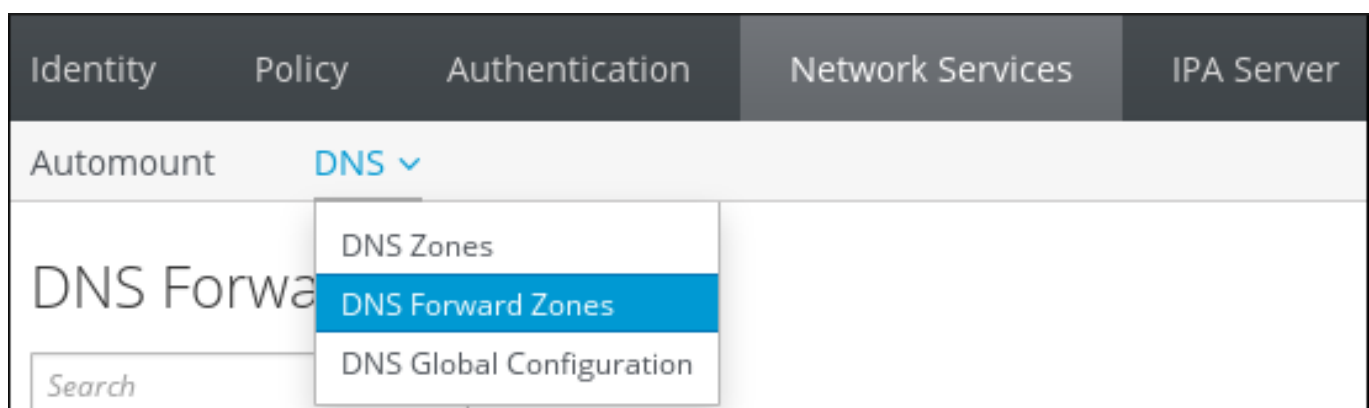
Forward zones are a non-standard solution, and using them can lead to unexpected and problematic behavior. When creating a new DNS zone, Red Hat recommends to always use standard DNS delegation using NS records and to avoid forward zones.

For information on the supported forward policies, see [Section 15.7, “Forward Policies”](#).

For further information about the BIND service, see the [Red Hat Enterprise Linux Networking Guide](#), the BIND 9 Administrator Reference Manual included in the `/usr/share/doc/bind-version_number/` directory, or external sources <sup>[7]</sup>.

## Configuring Forward Zones in the Web UI

To manage forward zones in the web UI, click the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Forward Zones** section.



**Figure 15.29. Managing DNS Forward Zones**

In the **DNS Forward Zones** section, the administrator can handle all required operations regarding forward zones: show current list of forward zones, add a new forward zone, delete a forward zone, display a forward zone, allow to modify forwarders and forward policy per a forward zone, and disable or enable a forward zone.

**Configuring Forward Zones from the Command Line**

To manage forward zones from the command line, use the **ipa dnsforwardzone-\*** commands described below.

**Note**

The **ipa dnsforwardzone-\*** commands behave consistently with the **ipa dnszone-\*** commands used to manage master zones.

The **ipa dnsforwardzone-\*** commands accept several options; notably, the **--forwarder**, **--forward-policy**, and **--name-from-ip** options. For detailed information about the available options, see [Table 15.1, “Zone Attributes”](#) or run the commands with the **--help** option added, for example:

```
ipa dnsforwardzone-add --help
```

**Adding Forward Zones**

Use the **dnsforwardzone-add** command to add a new forward zone. It is required to specify at least one forwarder if the forward policy is not set to **none**.

```
[user@server ~]$ ipa dnsforwardzone-add zone.test. --
forwarder=172.16.0.1 --forwarder=172.16.0.2 --forward-
policy=first

Zone name: zone.test.
Zone forwarders: 172.16.0.1, 172.16.0.2
Forward policy: first
```

**Modifying Forward Zones**

Use the **dnsforwardzone-mod** command to modify a forward zone. It is required to specify at least one forwarder if the forward policy is not **none**. Modifications can be performed in several ways.

```
[user@server ~]$ ipa dnsforwardzone-mod zone.test. --
forwarder=172.16.0.3

Zone name: zone.test.
Zone forwarders: 172.16.0.3
Forward policy: first
```

```
[user@server ~]$ ipa dnsforwardzone-mod zone.test. --forward-
policy=only
```

```
Zone name: zone.test.
Zone forwarders: 172.16.0.3
Forward policy: only
```

## Showing Forward Zones

Use the **dnsforwardzone-show** command to display information about a specified forward zone.

```
[user@server ~]$ ipa dnsforwardzone-show zone.test.

Zone name: zone.test.
Zone forwarders: 172.16.0.5
Forward policy: first
```

## Finding Forward Zones

Use the **dnsforwardzone-find** command to locate a specified forward zone.

```
[user@server ~]$ ipa dnsforwardzone-find zone.test.

Zone name: zone.test.
Zone forwarders: 172.16.0.3
Forward policy: first
-----
Number of entries returned 1
-----
```

## Deleting Forward Zones

Use the **dnsforwardzone-del** command to delete specified forward zones.

```
[user@server ~]$ ipa dnsforwardzone-del zone.test.

-----
Deleted forward DNS zone "zone.test."
-----
```

## Enabling and Disabling Forward Zones

Use **dnsforwardzone-enable** and **dnsforwardzone-disable** commands to enable and disable forward zones. Note that forward zones are enabled by default.

```
[user@server ~]$ ipa dnsforwardzone-enable zone.test.

-----
Enabled forward DNS zone "zone.test."
-----
```

```
[user@server ~]$ ipa dnsforwardzone-disable zone.test.

-----
Disabled forward DNS zone "zone.test."
-----
```

## Adding and Removing Permissions

Use **dnsforwardzone-add-permission** and **dnsforwardzone-remove-permission** commands to add or remove system permissions.

```
[user@server ~]$ ipa dnsforwardzone-add-permission zone.test.
```

```
-----
Added system permission "Manage DNS zone zone.test."
-----
```

```
Manage DNS zone zone.test.
```

```
[user@server ~]$ ipa dnsforwardzone-remove-permission zone.test.
```

```
-----
Removed system permission "Manage DNS zone zone.test."
-----
```

```
Manage DNS zone zone.test.
```

## 15.8. Managing Reverse DNS Zones

A reverse DNS zone can be identified in the following two ways:

- By the zone name, in the format **reverse\_ipv4\_address.in-addr.arpa** or **reverse\_ipv6\_address.ip6.arpa**.

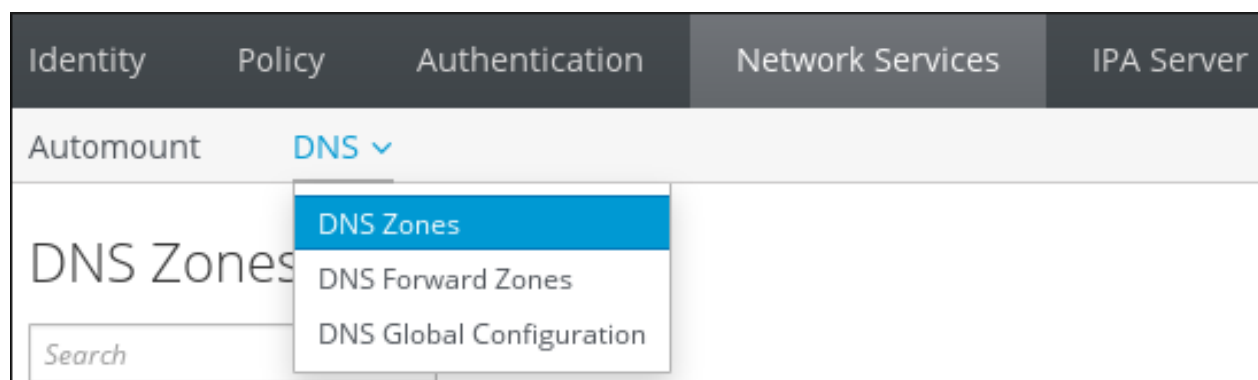
The reverse IP address is created by reversing the order of the components of the IP address. For example, if the IPv4 network is **192.0.2.0/24**, the reverse zone name is **2.0.192.in-addr.arpa.** (with the trailing period).

- By the network address, in the format **network\_ip\_address/subnet\_mask\_bit\_count**

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.

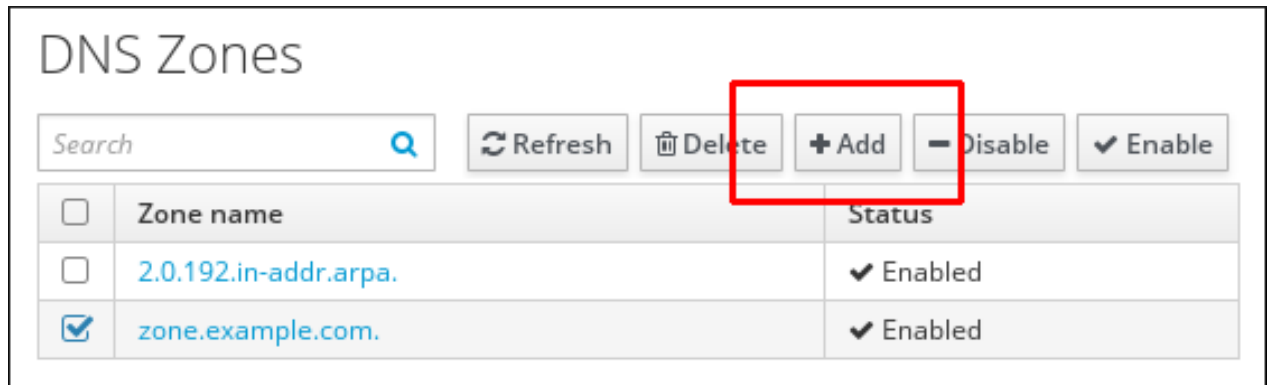
### Adding a Reverse DNS Zone in the Web UI

1. Open the **Network Services** tab, and select the **DNS** subtab, followed by the **DNS Zones** section.



**Figure 15.30. DNS Zone Management**

2. Click **Add** at the top of the list of all zones.

**Figure 15.31. Adding a Reverse DNS Zone**

3. Fill in the zone name or the reverse zone IP network.
  - a. For example, to add a reverse DNS zone by the zone name:

The screenshot shows the 'Add DNS Zone' dialog box. It has a title bar with a close button. Inside, there are two radio buttons: 'Zone name' (selected) and 'Reverse zone IP network'. The 'Zone name' field is filled with '2.0.192.in-addr.arpa.'. There is a note '\* Required field' below the radio buttons. At the bottom, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

**Figure 15.32. Creating a Reverse Zone by Name**

- b. Alternatively, to add a reverse DNS zone by the reverse zone IP network:

**Figure 15.33. Creating a Reverse Zone by IP Network**

The validator for the **Reverse zone IP network** field warns you about an invalid network address during typing. The warning will disappear once you enter the full network address.

4. Click **Add** to confirm the new reverse zone.

## Adding a Reverse DNS Zone from the Command Line

To create a reverse DNS zone from the command line, use the **ipa dnszone-add** command.

For example, to create the reverse zone by the zone name:

```
[user@server]$ ipa dnszone-add 2.0.192.in-addr.arpa.
```

Alternatively, to create the reverse zone by the IP network:

```
[user@server ~]$ ipa dnszone-add --name-from-ip=192.0.2.0/24
```

## Other Management Operations for Reverse DNS Zones

[Section 15.5, “Managing Master DNS Zones”](#) describes other zone management operations, some of which are also applicable to reverse DNS zone management, such as editing or disabling and enabling DNS zones.

## 15.9. Defining DNS Query Policy

To resolve host names within the DNS domain, a DNS client issues a query to the DNS name server. For some security contexts or for performance, it might 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 with the **ipa dnszone-mod** command to set a list of clients which are allowed to issue queries.



For example:

```
[user@server ~]$ ipa dnszone-mod --allow-  
query=192.0.2.0/24;2001:DB8::/32;203.0.113.1 example.com
```

The default **--allow-query** value is **any**, which allows the zone to be queried by any client.

---

[5] For more information about GSS-TSIG, see [RFC 3545](#).

[6] For the full text of RFC 3007, see <http://tools.ietf.org/html/rfc3007>

[7] For more information, refer to the [BIND 9 Configuration Reference](#).

## **Part IV. Defining Domain-wide System Policies**

## Chapter 16. Using Automount

Automount is a way to manage, organize, and access directories across multiple systems. Automount automatically mounts a directory whenever access to it is requested. 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, see [Section 10.1, “Setting up User Home Directories”](#).

In IdM, automount works with the internal LDAP directory and also with DNS services if configured.

### 16.1. About Automount and IdM

Automount provides a coherent structure to the way that directories are organized. Every directory is called a *mount point* or a *key*. Multiple keys that are grouped together create a *map*, and maps are associated according to their physical or conceptual *location*.

The base configuration file for automount is the **auto.master** file in the **/etc/** directory. If necessary, there can be multiple **auto.master** configuration files in separate server locations.

When the **autofs** utility is configured on a server and the server is a client in an IdM domain, then all configuration information for automount is stored in the IdM directory. Rather than in separate text files, the **autofs** configuration containing 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 works with an existing **autofs** deployment but does not set up or configure **autofs** itself.

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:

- ✱ The **ipa automountlocation\*** commands for *Locations*,
- ✱ The **ipa automountmap\*** commands for direct and indirect *maps*,
- ✱ The **ipa automountkey\*** commands for *keys*.

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](#).

## 16.2. Configuring Automount

in Identity Management, configuring automount entries like locations and maps requires an existing autofs/NFS server. Creating automount entries does not create the underlying **autofs** configuration. **Autofs** can be configured manually using LDAP or SSSD as a data store, or it can be configured automatically.



### Note

Before changing the automount configuration, test that for at least one user, their **/home/** directory can be mounted from the command line successfully. Making sure that NFS is working properly makes it easier to troubleshoot any potential IdM automount configuration errors later.

### 16.2.1. Configuring NFS Automatically

After a system is configured as an IdM client, which includes IdM servers and replicas that are configured as domain clients as part of their configuration, **autofs** can be configured to use the IdM domain as its NFS domain and have **autofs** services enabled.

By default, the **ipa-client-automount** utility automatically configures the NFS configuration files, **/etc/sysconfig/nfs** and **/etc/idmapd.conf**. It also configures SSSD to manage the credentials for NFS. If the **ipa-client-automount** command is run without any options, it runs a DNS discovery scan to identify an available IdM server and creates a default location called **default**.

```
[root@ipa-server ~]# ipa-client-automount
Searching for IPA server...
IPA server: DNS discovery
Location: default
Continue to configure the system with these values? [no]: yes
Configured /etc/nsswitch.conf
Configured /etc/sysconfig/nfs
Configured /etc/idmapd.conf
Started rpcidmapd
Started rpcgssd
Restarting sssd, waiting for it to become available.
Started autofs
```

It is possible to specify an IdM server to use and to create an automount location other than default:

```
[root@server ~]# ipa-client-automount --server=ipaserver.example.com --
location=boston
```

Along with setting up NFS, the **ipa-client-automount** utility configures SSSD to cache automount maps, in case the external IdM store is ever inaccessible. Configuring SSSD does two things:

- ✧ It adds service configuration information to the SSSD configuration. The IdM domain entry is given settings for the autofs provider and the mount location.

```
autofs_provider = ipa
ipa_automount_location = default
```

And NFS is added to the list of supported services (**services = nss,pam,autofs...**) and given a blank configuration entry (**[autofs]**).

- ✧ The Name Service Switch (NSS) service information is updated to check SSSD first for automount information, and then the local files.

```
automount: sss files
```

There may be some instances, such as highly secure environments, where it is not appropriate for a client to cache automount maps. In that case, the **ipa-client-automount** command can be run with the **--no-sssd** option, which changes all of the required NFS configuration files, but does not change the SSSD configuration.

```
[root@server ~]# ipa-client-automount --no-sssd
```

All of the required NFS configuration files — but the list of files is slightly different without SSSD:

- ✧ The command updates **/etc/sysconfig/autofs** instead of **/etc/sysconfig/nfs**.
- ✧ The command configures **/etc/autofs\_ldap\_auth.conf** with the IdM LDAP configuration.
- ✧ The command configures **/etc/nsswitch.conf** to use the LDAP services for automount maps.



## Note

The **ipa-client-automount** command can only be run once. If there is an error in the configuration, then the configuration files need to be edited manually.

### 16.2.2. Configuring autofs Manually to Use SSSD and Identity Management

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 16.4, “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. Configure autofs as one of the services which SSSD manages.
  - a. Open the SSSD configuration file.

```
[root@server ~]# vim /etc/sss/sss.conf
```

- b. Add the autofs service to the list of services handled by SSSD.

```
[sss]
services = nss,pam,autofs
```

- c. Create a new **[autofs]** section. This can be left blank; the default settings for an autofs service work with most infrastructures.

```
[nss]

[pam]
```

```
[sudo]

[autofs]

[ssh]

[pac]
```

- d. Optionally, set a search base for the autofs entries. By default, this is the LDAP search base, but a subtree can be specified in the **ldap\_autofs\_search\_base** parameter.

```
[domain/EXAMPLE]
...
ldap_search_base = "dc=example,dc=com"
ldap_autofs_search_base = "ou=automount,dc=example,dc=com"
```

5. Restart SSSD:

```
[root@server ~]# systemctl restart sssd.service
```

6. Check the **/etc/nsswitch.conf** file, so that SSSD is listed as a source for automount configuration:

```
automount: sss files
```

7. Restart autofs:

```
[root@server ~]# systemctl restart autofs.service
```

8. Test the configuration by listing a user's **/home** directory:

```
[root@server ~]# 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**.



## Note

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.

### 16.2.3. Configuring Automount on Solaris



#### Note

Solaris uses a different schema for autofs configuration than the schema used by Identity Management. Identity Management uses the 2307bis-style automount schema which is defined for 389 Directory Server (and used in IdM's internal Directory Server instance).

1. If the NFS server is running on Red Hat Enterprise Linux, specify on the Solaris machine that NFSv3 is the maximum supported version. Edit the `/etc/default/nfs` file and set the following parameter:

```
NFS_CLIENT_VERSION=3
```

2. Use the `ldapclient` command to configure the host to use LDAP:

```
ldapclient -v manual -a authenticationMethod=none
-a defaultSearchBase=dc=example,dc=com
-a defaultServerList=ipa.example.com
-a
serviceSearchDescriptor=passwd:cn=users,cn=accounts,dc=example,dc=com
-a
serviceSearchDescriptor=group:cn=groups,cn=compat,dc=example,dc=com
-a
serviceSearchDescriptor=auto_master:automountMapName=auto.master,cn=location,cn=automount,dc=example,dc=com?one
-a
serviceSearchDescriptor=auto_home:automountMapName=auto_home,cn=location,cn=automount,dc=example,dc=com?one
-a objectClassMap=shadow:shadowAccount=posixAccount
-a searchTimeLimit=15
-a bindTimeLimit=5
```

3. Enable **automount**:

```
# svcadm enable svc:/system/filesystem/autofs
```

4. Test the configuration.

- a. Check the LDAP configuration:

```
# ldapclient -l auto_master

dn:
automountkey=/home,automountmapname=auto.master,cn=location,cn=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: /home
automountInformation: auto.home
```



- b. List a user's **/home** directory:

```
# ls /home/userName
```

## 16.3. 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.

### 16.3.1. 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 5.4.2, “Other Examples of Adding a Host Entry”](#).
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 12.1, “Adding and Editing Service Entries and Keytabs”](#).

4. Generate an NFS service keytab for the NFS server using the **ipa-getkeytab** command, and save the keys directly to the host keytab. For example:

```
[jsmith@server ~]$ ipa-getkeytab -s ipaserver.example.com -p  
nfs/nfs-server.example.com -k /etc/krb5.keytab
```



### Note

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
```

**Note**

This procedure assumes that the NFS server is running on a Red Hat Enterprise Linux system or a UNIX system which can run **ipa-getkeytab**.

If the NFS server is running on a system which cannot run **ipa-getkeytab**, then create the keytab using system tools. Two things must be done:

- ✧ The key must be created in the **/root** (or equivalent) directory.
- ✧ The **ktutil** command can merge the keys into the system **/etc/krb5.keytab** file. The [ktutil man page](#) describes how to use the tool.

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. Run the **ipa-client-automount** command to configure the NFS settings.

By default, this enables secure NFS in the **/etc/sysconfig/nfs** file and sets the IdM DNS domain in the **Domain** parameter in the **/etc/idmapi.conf** file.

8. Edit the **/etc/exports** file and add the Kerberos information:

```
/export *(rw,sec=sys:krb5:krb5i:krb5p)
```

9. Restart the NFS server and related services.

```
[root@nfs-server ~]# systemctl restart nfs.service
[root@nfs-server ~]# systemctl restart nfs-server.service
[root@nfs-server ~]# systemctl restart nfs-secure.service
[root@nfs-server ~]# systemctl restart nfs-secure-server.service
```

10. Configure the NFS server as an NFS client, following the directions in [Section 16.3.2, “Setting up a Kerberized NFS Client”](#).

### 16.3.2. 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 5.4.2, “Other Examples of Adding a Host Entry”](#).
3. Run the **ipa-client-automount** command to configure the NFS settings.

By default, this enables secure NFS in the `/etc/sysconfig/nfs` file and sets the IdM DNS domain in the **Domain** parameter in the `/etc/idmapd.conf` file.

4. Start the GSS daemon.

```
[root@nfs-client-server ~]# systemctl start rpc-gssd.service
[root@nfs-client-server ~]# systemctl start rpcbind.service
[root@nfs-client-server ~]# systemctl start nfs-idmapd.service
```

5. 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
```

6. Configure SSSD on the client system to manage home directories and renew Kerberos tickets.

- a. Enable SSSD with the **--enablemkhomedir** option.

```
[root@nfs-client-server ~]# authconfig --update --enablesssd
--enablesssdauth --enablemkhomedir
```

- b. Restart the OpenSSH client.

```
[root@nfs-client-server ~]# systemctl start ssh.service
```

- c. Edit the IdM domain section in the SSSD configuration file to set the keytab renewal options.

```
[root@nfs-client-server ~]# vim /etc/sss/sss.conf

[domain/EXAMPLE.COM]
cache_credentials = True
```

```
krb5_store_password_if_offline = True
ipa_domain = example.com
id_provider = ipa
auth_provider = ipa
...
krb5_renewable_lifetime = 50d
krb5_renew_interval = 3600
```

d. Restart SSSD.

```
[root@nfs-client-server ~]# systemctl restart sssd.service
```

## 16.4. 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.

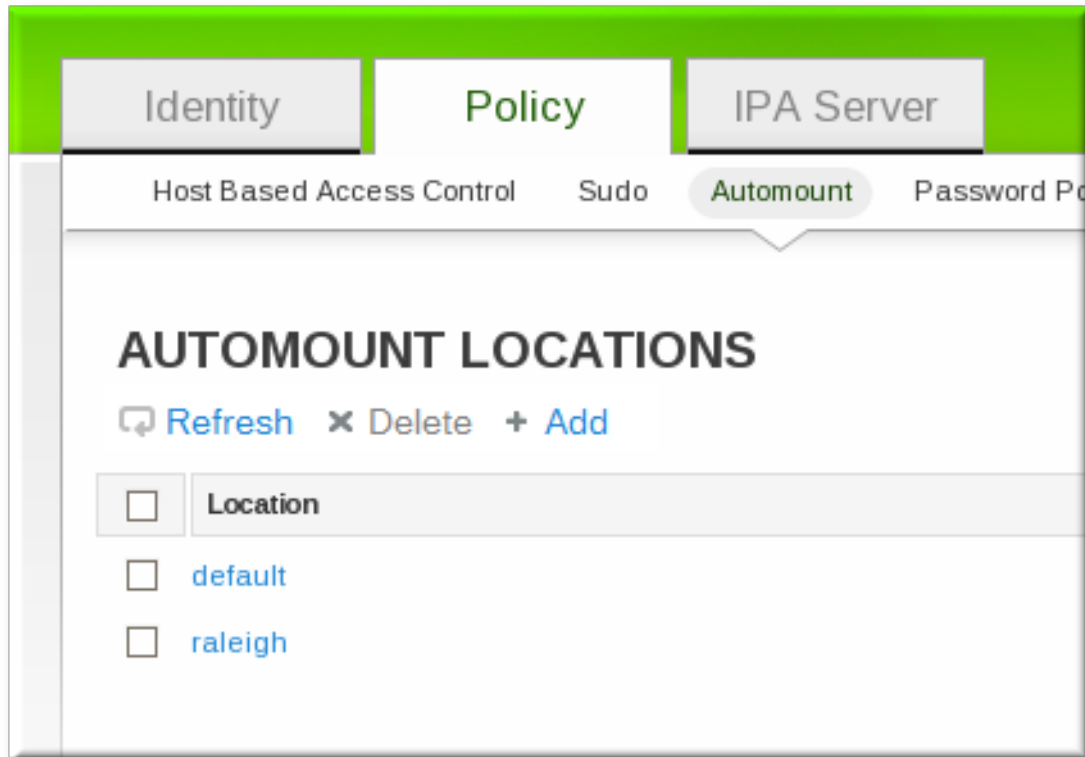


### Important

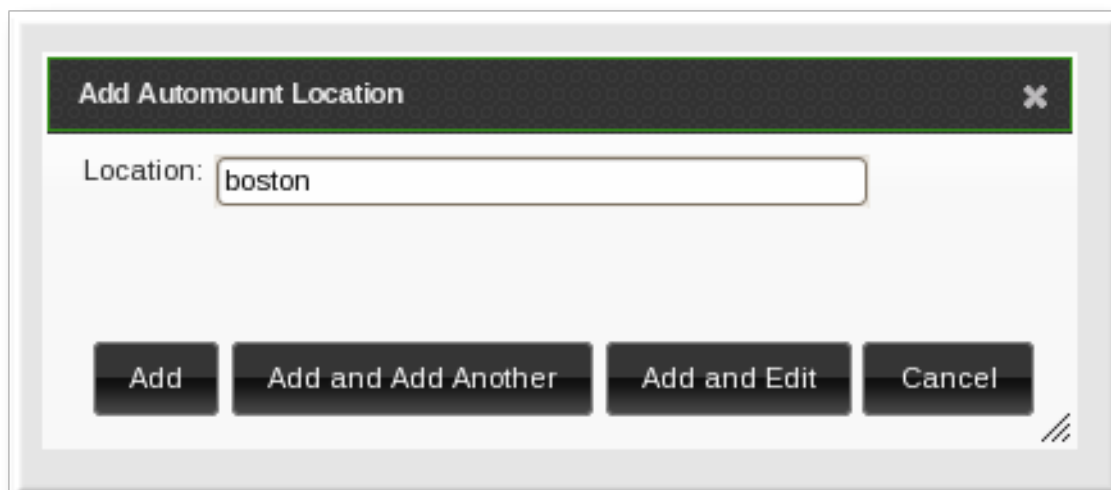
Identity Management does not set up or configure autofs. That must be done separately. Identity Management works with an existing autofs deployment.

### 16.4.1. 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 16.5.1.1, “Configuring Direct Maps from the Web UI”](#) and [Section 16.5.2.1, “Configuring Indirect Maps from the Web UI”](#).

### 16.4.2. 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:
```

## 16.5. 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.

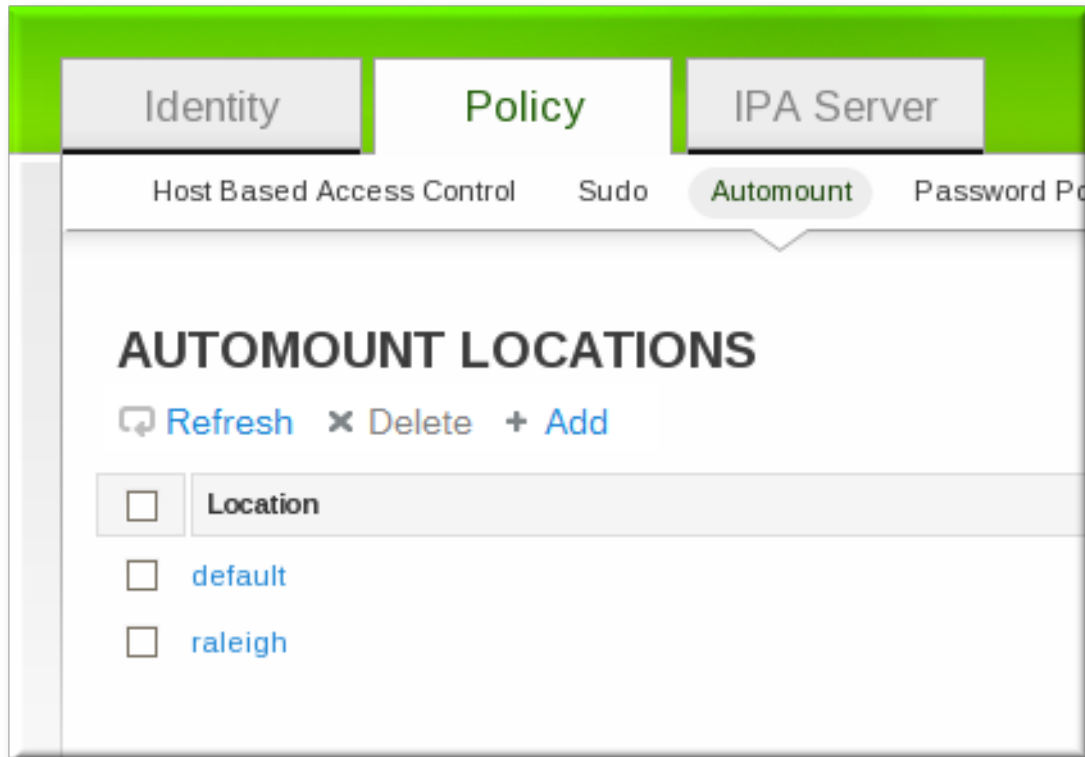
### 16.5.1. Configuring Direct Maps

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:

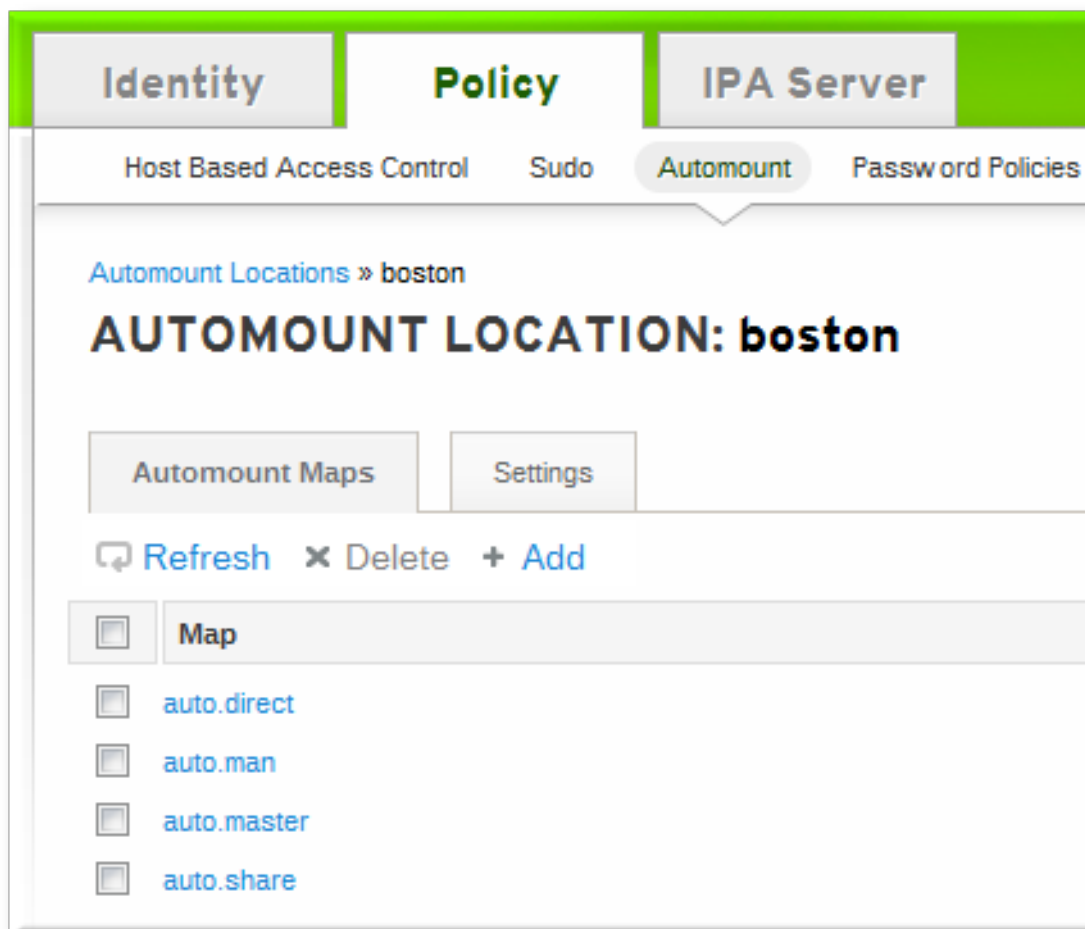
```
-----
/etc/auto.direct:
/shared/man server.example.com:/shared/man
```

#### 16.5.1.1. Configuring Direct Maps from the Web UI

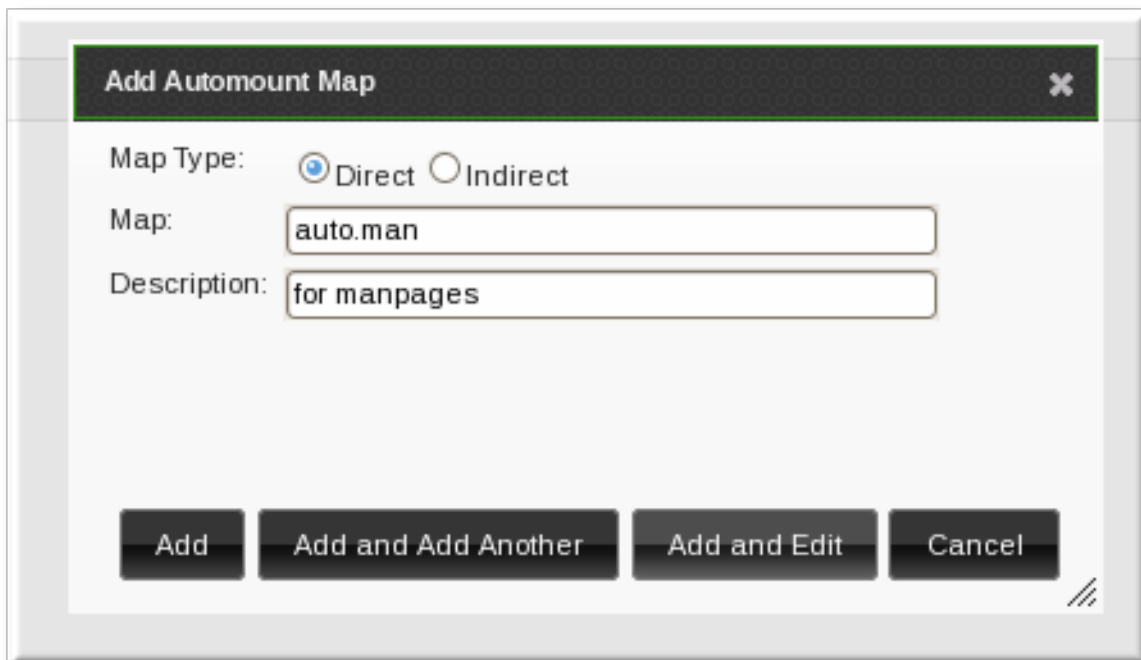
1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click name of the automount location to which to add the map.



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**

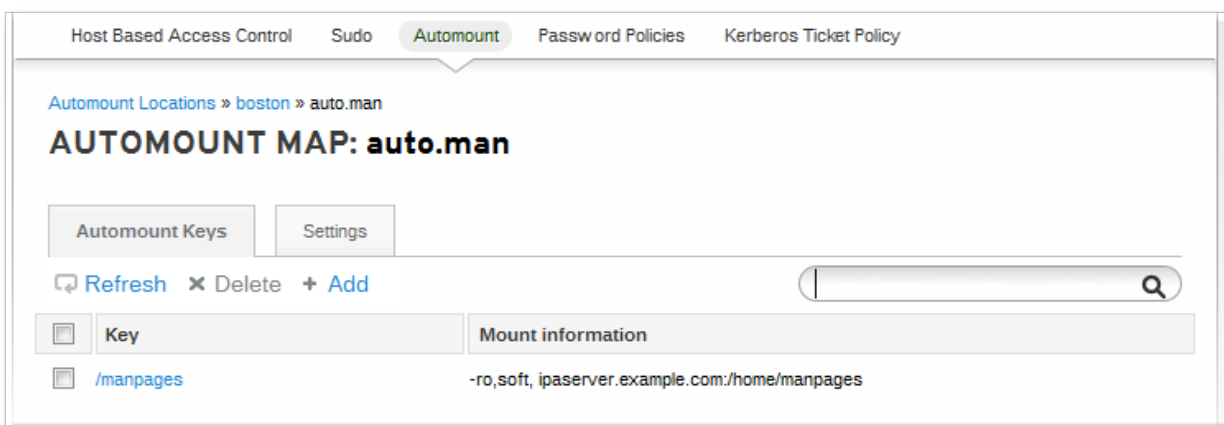
Map Type: ☒ Direct ☐ Indirect

Map:

Description:

**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

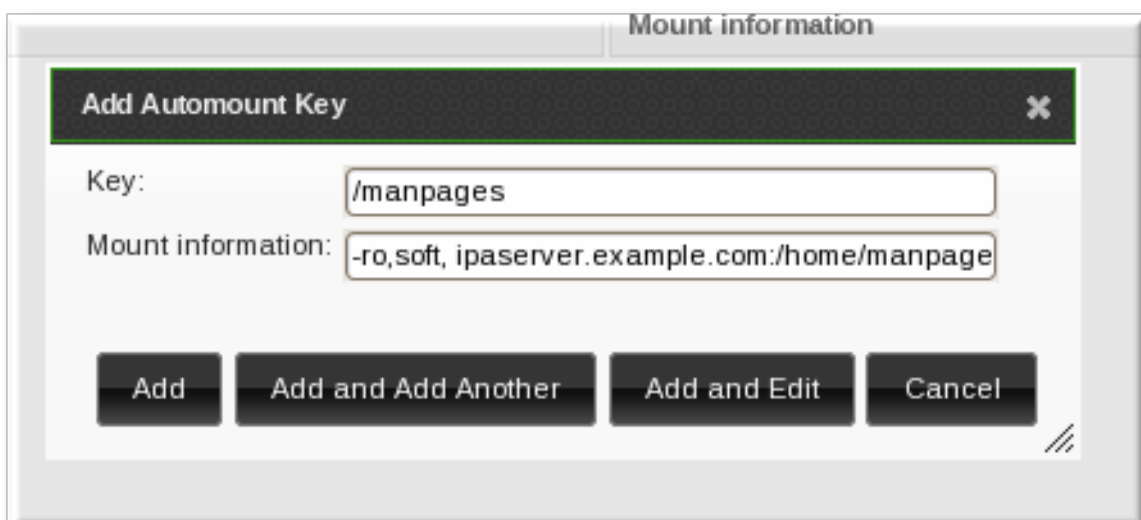
**AUTOMOUNT MAP: auto.man**

Automount Keys Settings

[Refresh](#) [Delete](#) [+ Add](#)

Key	Mount information
<input type="checkbox"/> /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:

**Add** **Add and Add Another** **Add and Edit** **Cancel**

8. Click the **Add** button to save the new key.



### 16.5.1.2. 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:

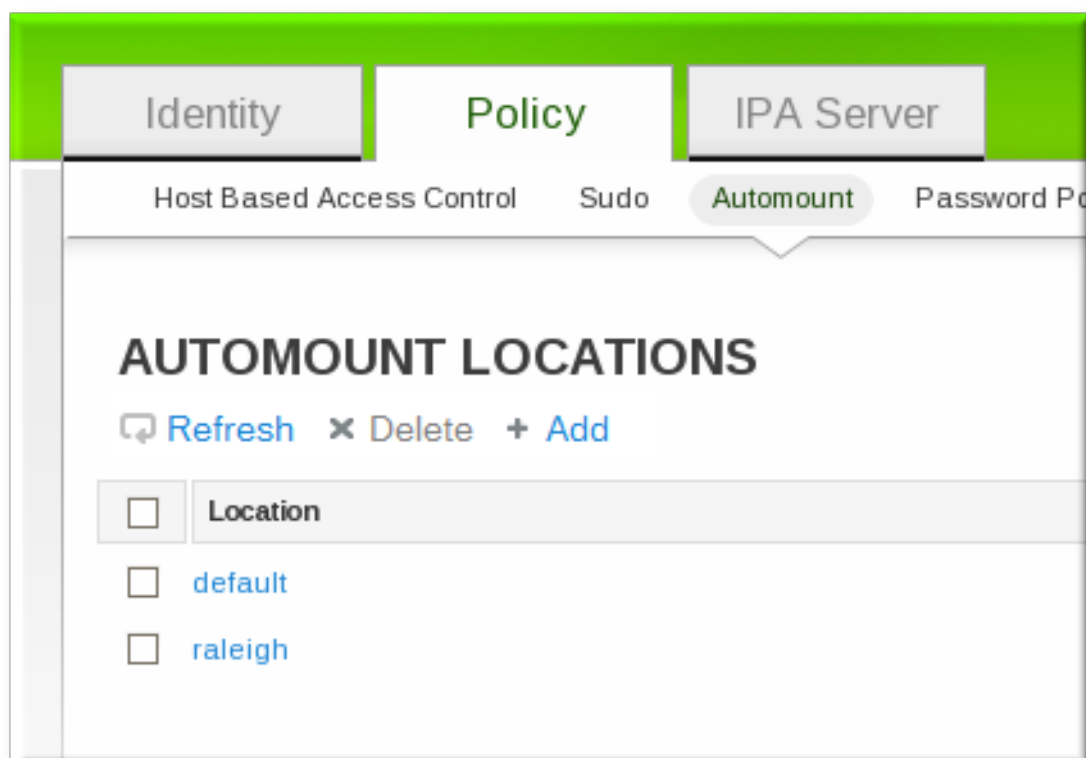
```
ldapclient -a
serviceSearchDescriptor=auto_direct:automountMapName=auto.direct,cn=location,cn=automount,dc=example,dc=com?one
```

## 16.5.2. Configuring Indirect Maps

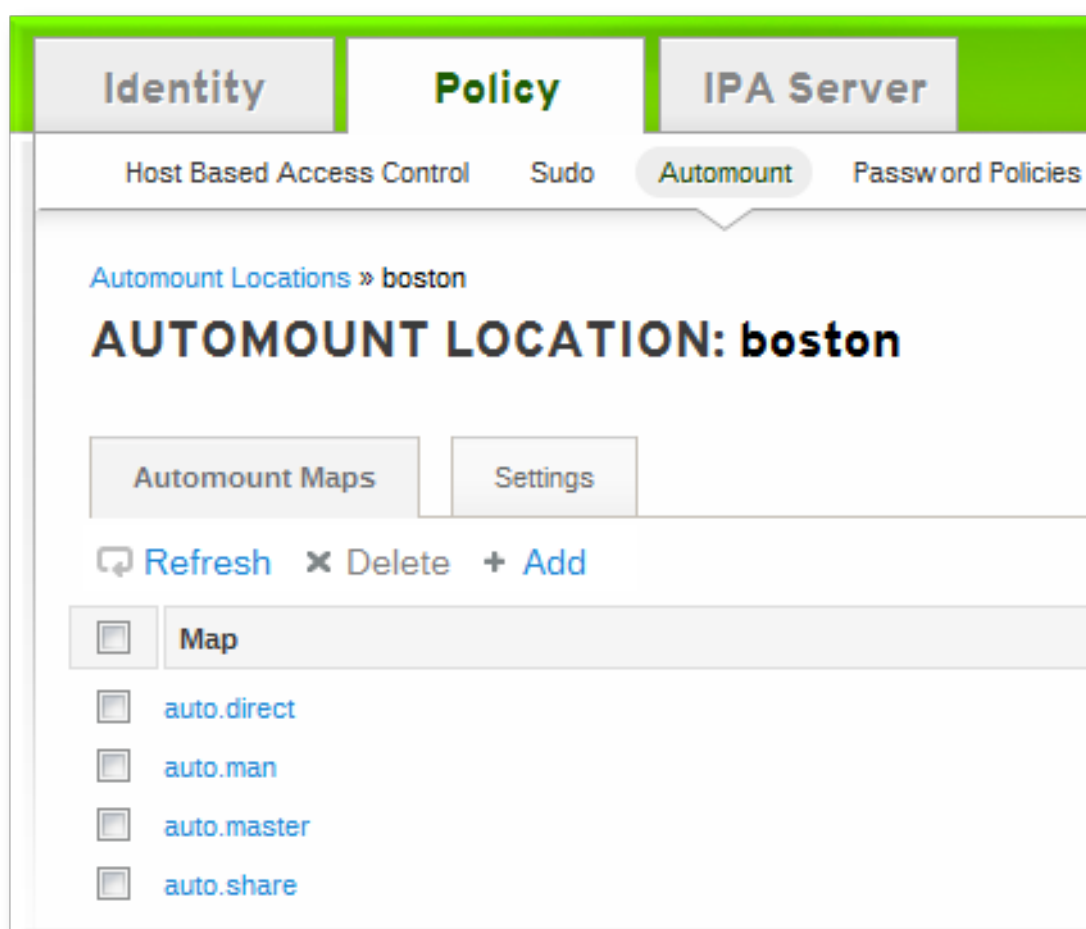
An indirect map essentially specifies a relative path for maps. A parent entry sets the base directory for all of the indirect maps. The indirect map key sets a sub directory; whenever the indirect map location is loaded, the key is appended to that base directory. For example, if the base directory is **/docs** and the key is **man**, then the map is **/docs/man**.

### 16.5.2.1. Configuring Indirect Maps from the Web UI

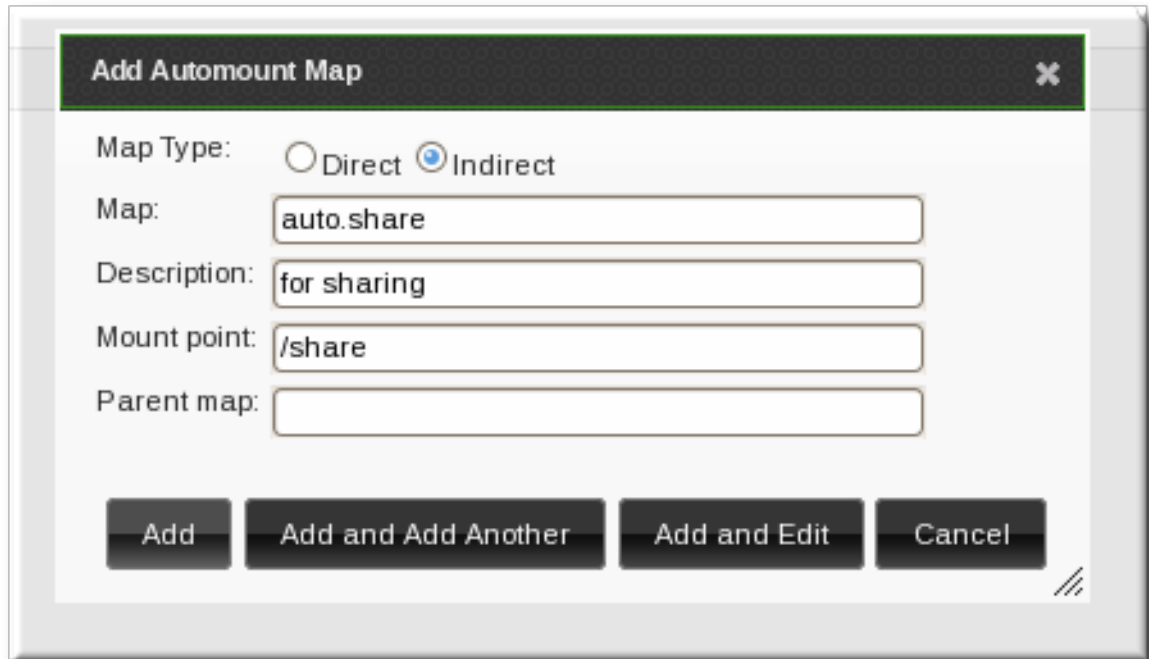
1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click name of the automount location to which to add the map.



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.

### 16.5.2.2. 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=locati
on,cn=automount,dc=example,dc=com?one
```

### 16.5.3. 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 17. 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.

### 17.1. 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 priority, 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 17.1, “Password Policy Settings”](#).

**Table 17.1. Password Policy Settings**

Configuration Property	Command-Line Option	Description
<b>Options for both the UI and CLI</b>		

Configuration Property	Command-Line Option	Description
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	<p>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:</p> <ul style="list-style-type: none"> <li>✧ Upper-case characters</li> <li>✧ Lower-case characters</li> <li>✧ Digits</li> <li>✧ Special characters (for example, punctuation)</li> <li>✧ 8-bit characters (characters whose decimal code starts at 128 or below)</li> <li>✧ Number of repeated characters</li> </ul> <p>This weights in the opposite direction, so that too many repeated characters does meet the quorum to satisfy the "level" expressed by krbPwdMinDiffChars.</p>

Configuration Property	Command-Line Option	Description
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.
<div>  <b>Note</b>            Even with the password history set to zero, users cannot reuse a <i>current</i> password.         </div>		
<b>Options for the CLI only</b>		
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.

## 17.2. Viewing Password Policies

There can be multiple password policies configured in IdM. There is always a global policy, which is set when the server is created. 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 **pwdpolicy-show** command. The CLI can also display the password policy in effect for a user.

### 17.2.1. Viewing the Global Password Policy

The global password policy is created as part of the initial IdM server setup. This policy applies to every user until a group-level password policy supersedes it.

The default settings for the global password policy are listed in [Table 17.2, “Default Global Password Policy”](#).

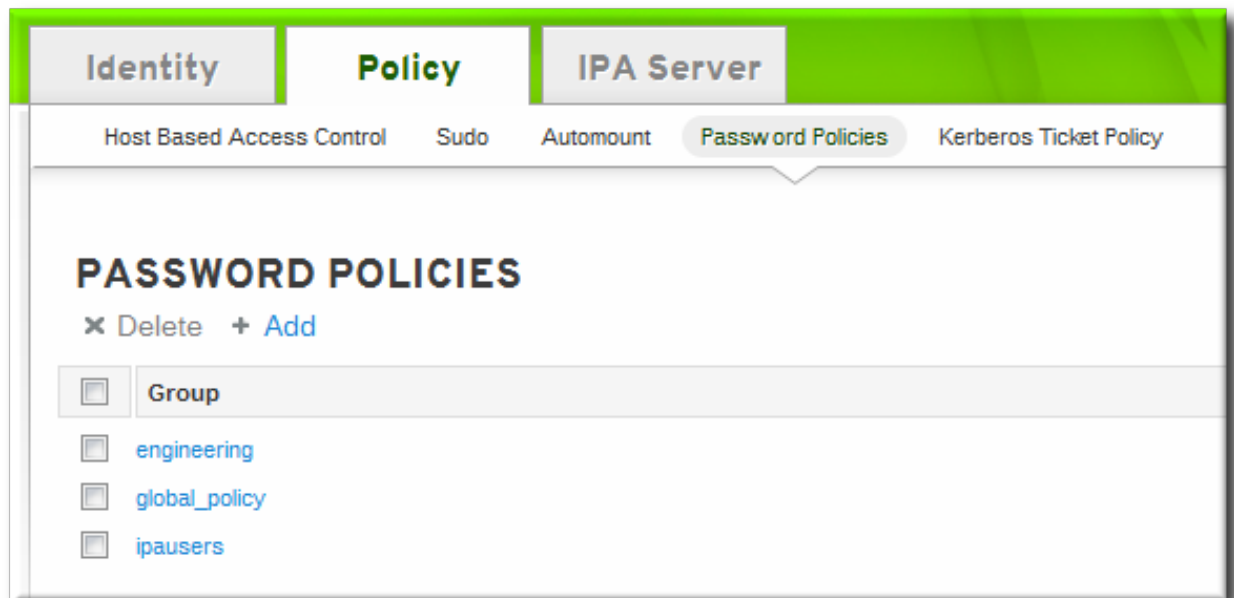
**Table 17.2. Default Global Password Policy**

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

#### 17.2.1.1. 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:	

### 17.2.1.2. With the Command Line

To view the global policy, simply run the **pwpolicy-show** command with no arguments:

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-show
```

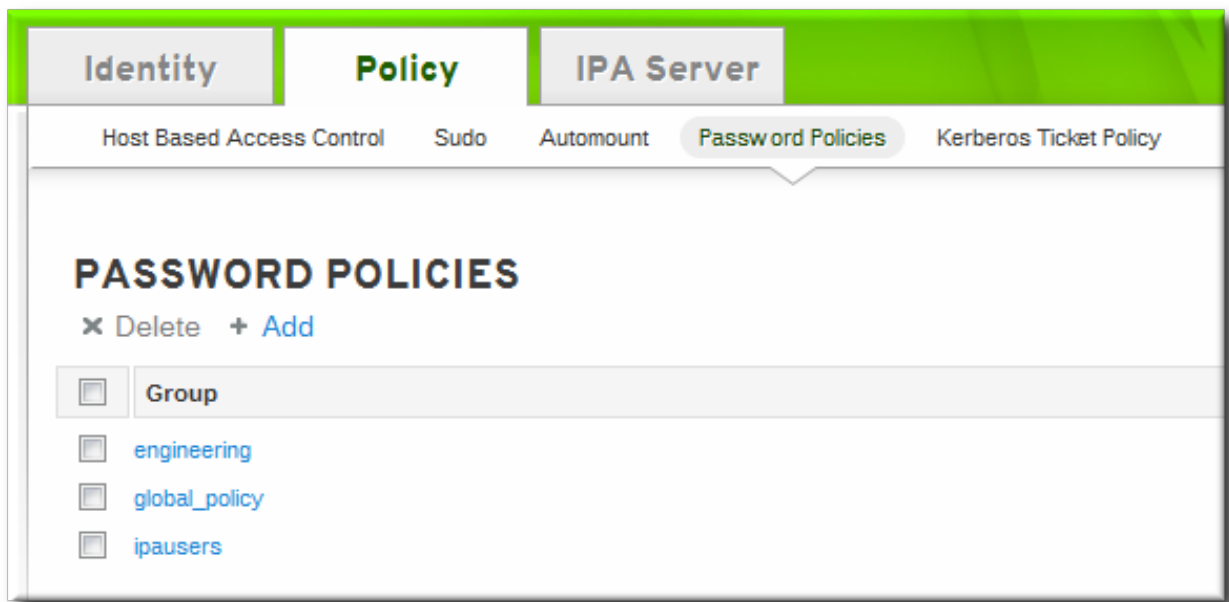
```
Group: global_policy
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 0
Character classes: 0
```

Min length: 8  
Max failures: 6  
Failure reset interval: 60  
Lockout duration: 600

## 17.2.2. Viewing Group-Level Password Policies

### 17.2.2.1. 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.

### 17.2.2.2. With the Command Line

For a group-level password policy, specify the group name with the command:

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-show ipausers
Group: ipausers
Max lifetime (days): 120
Min lifetime (hours): 10
Min length: 10
Priority: 50
```

### 17.2.3. 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.

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-show --user=jsmith
```

```
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
```

## 17.3. 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.

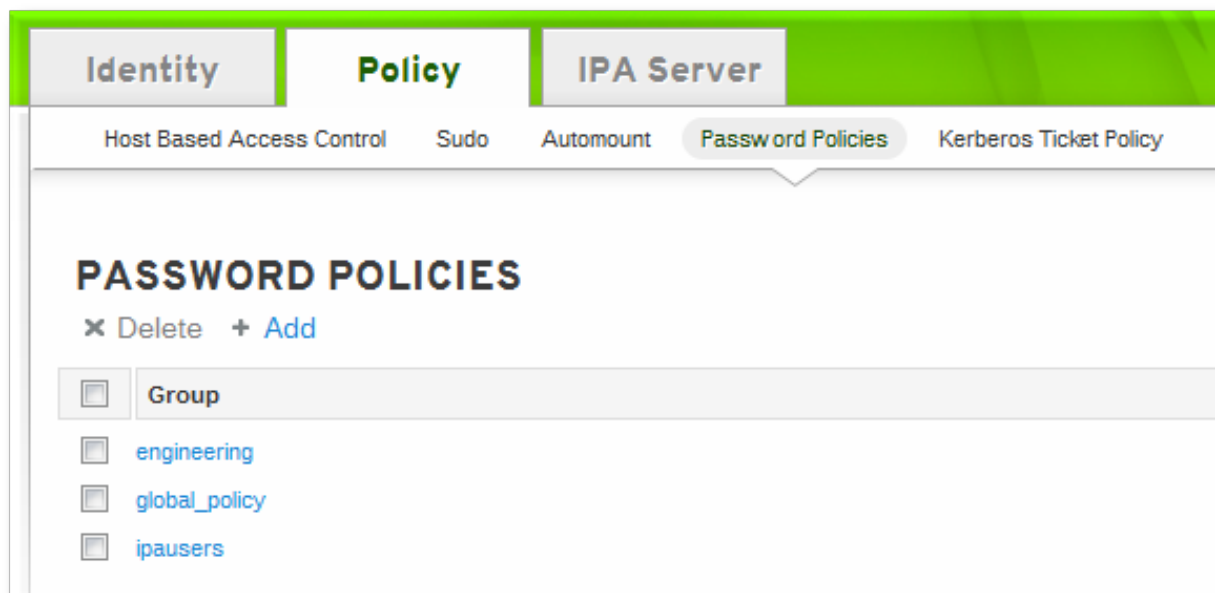


### Note

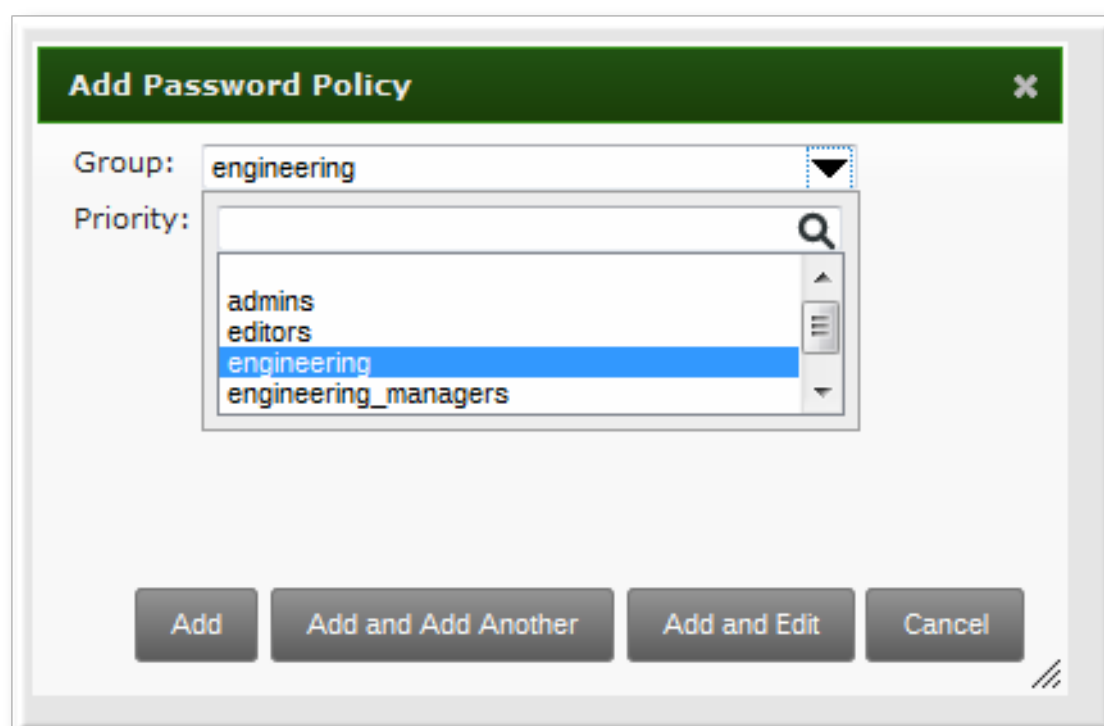
It is not possible to set a password policy for a non-existent group.

### 17.3.1. 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. Conversely, the highest priority policy has the lowest number.

Only one password policy is in effect for a user, and that is the highest priority policy.



### 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 *number* of different categories of character that must be used in the password. This does not set which classes must be used; it sets the number of different (unspecified) classes which must be used in a password. For example, a character class can be a number, special character, or capital; the complete list of categories is in [Table 17.1, “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.

### 17.3.2. Creating Password Policies with the Command Line

Password policies are added with the **pwpolicy-add** command.

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-add groupName --attribute-value
```

For example:

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-add exampleGroup --minlife=7 --maxlife=49
--history= --priority=1
Group: exampleGroup
Max lifetime (days): 49
Min lifetime (hours): 7
Priority: 1
```



## Note

Setting an attribute to a blank value effectively removes that attribute from the password policy.

### 17.3.3. 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
```

## 17.4. 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 17.3, “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 `ldamodify`.** For example, for a single user:

```
[bjensen@ipaserver ~]$ ldapmodify -D "cn=Directory Manager" -w secret -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 **ldamodify** command.



## Note

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.

## 17.5. 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.

```
[root@server ~]# kinit admin
[root@server ~]# ipa pwpolicy-mod examplegroup --priority=10
```

When a user belongs to multiple groups, the group password policy with the lowest priority *number* has the highest priority.

## 17.6. Setting Account Lockout Policies

A brute force attack occurs when an attacker attempts to guess a password by simply flooding 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.



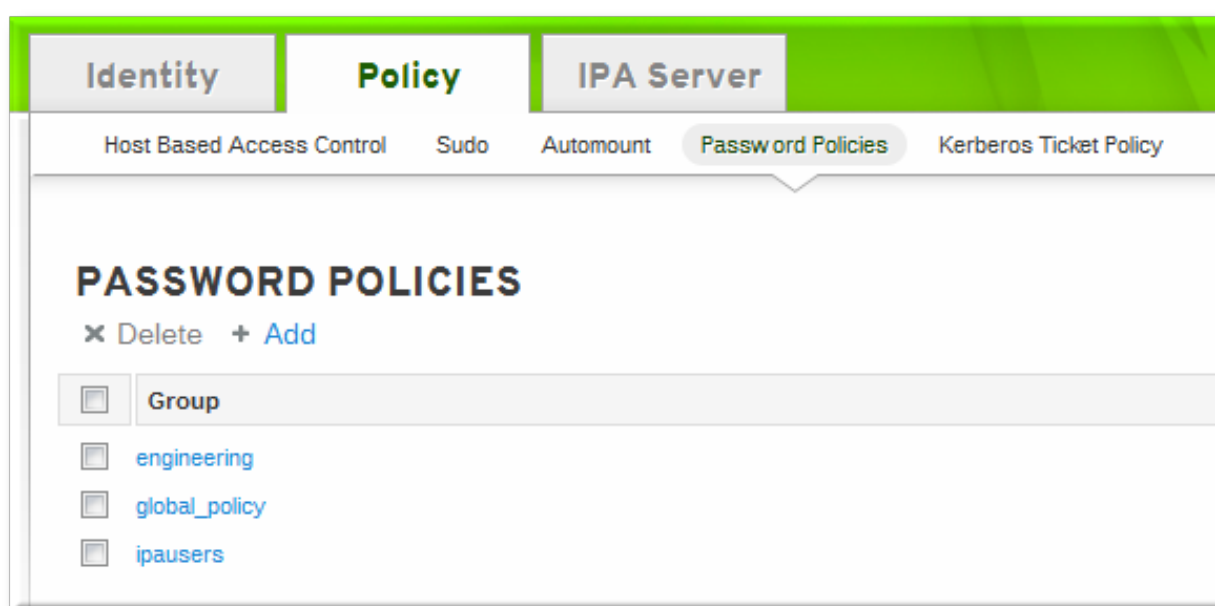
## Note

A user account can be manually unlocked by an administrator using the **ipa user-unlock** command. Also see [Section 10.6, “Unlocking User Accounts After Password Failures”](#).

### 17.6.1. 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.

▼ **PASSWORD POLICY**

Group: `global_policy`

Max lifetime (days):

Min lifetime (hours):

History size:

Character classes:

Min length:

Max failures:

Failure reset interval:

Lockout duration:

Priority:

There are three parts to the account lockout policy:

- ✧ **Max Failures** sets the number of failed login attempts before the account is locked.
- ✧ **Failure reset interval** sets the number of seconds after a failed login attempt before the counter resets. Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally lapses after the set amount of time.
- ✧ **Lockout duration** sets then number of seconds for an account to remain locked after the maximum number of failed attempts is reached. Note that if this field is set to **0**, the account will be permanently locked in such a case.

### 17.6.2. In the CLI

There are three parts to the account lockout policy:

- ✧ The **--maxfail** option specifies the number of failed login attempts before the account is locked.
- ✧ The **--failinterval** option sets the number of seconds after a failed login attempt before the counter resets. Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally lapses after the set amount of time.
- ✧ The **--lockouttime** option sets then number of seconds for an account to remain locked after the maximum number of failed attempts is reached. Note that if the **0** value is used, the account will be permanently locked in such a case.

These account lockout options 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
```

## 17.7. 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 18. 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.

### 18.1. 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.

#### 18.1.1. 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**.

### 18.1.2. 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**.

## 18.2. 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.

### 18.2.1. Setting Global Ticket Policies

#### 18.2.1.1. 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.

```
[root@server ~]# systemctl start krb5kdc.service
```



### Important

Any change to the global Kerberos ticket policy requires a restart of the KDC for the changes to take effect.

#### 18.2.1.2. 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:

```
[root@server ~]# systemctl restart krb5kdc.service
```

### 18.2.2. 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.

### 18.3. Refreshing Kerberos Tickets

Kerberos keys are analogous to passwords. As with password policies, Kerberos tickets come under security policies which require them to be 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:

```
[root@server ~]# ldapsearch -x -b
```



```
"cn=computers,cn=accounts,dc=example,dc=com" "(&
(krblastpwdchange>=20100101000000)
(krblastpwdchange<=20101231235959))" dn krbprincipalname
...
[root@server ~]# 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.

## 18.4. Kerberos Flags for Services and Hosts

Various Kerberos flags can be used to define certain specific aspects of the Kerberos ticket behavior. You can add these flags to service and host Kerberos principals.

Principals in IdM accept the following two Kerberos flags:

### **OK\_AS\_DELEGATE**

Use this flag to specify Kerberos tickets trusted for delegation.

AD clients check the **OK\_AS\_DELEGATE** flag on the Kerberos ticket to determine whether the user credentials can be forwarded or delegated to the specific server; AD forwards the TGT only to services or hosts with **OK\_AS\_DELEGATE** set. With this flag, SSSD can add the AD user TGT to the default Kerberos credentials cache on the IdM client machine.

### **REQUIRES\_PRE\_AUTH**

Use this flag to specify that only pre-authenticated tickets are allowed to authenticate to the principal.

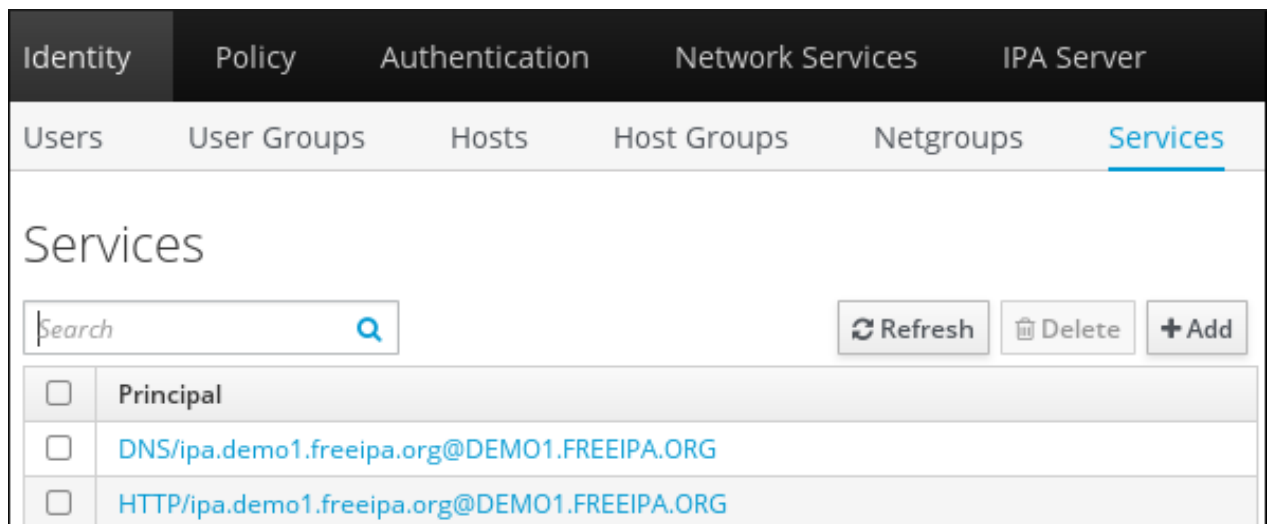
With the **REQUIRES\_PRE\_AUTH** flag set, the KDC requires additional authentication: the KDC issues the TGT for a principal with **REQUIRES\_PRE\_AUTH** only if the TGT has been pre-authenticated.

You can use **REQUIRES\_PRE\_AUTH** to disable pre-authentication for selected services or hosts, which lowers the load on the KDC but also slightly increases the possibility of a brute-force attack on a long-term key to succeed.

### 18.4.1. Setting Kerberos Flags from the Web UI

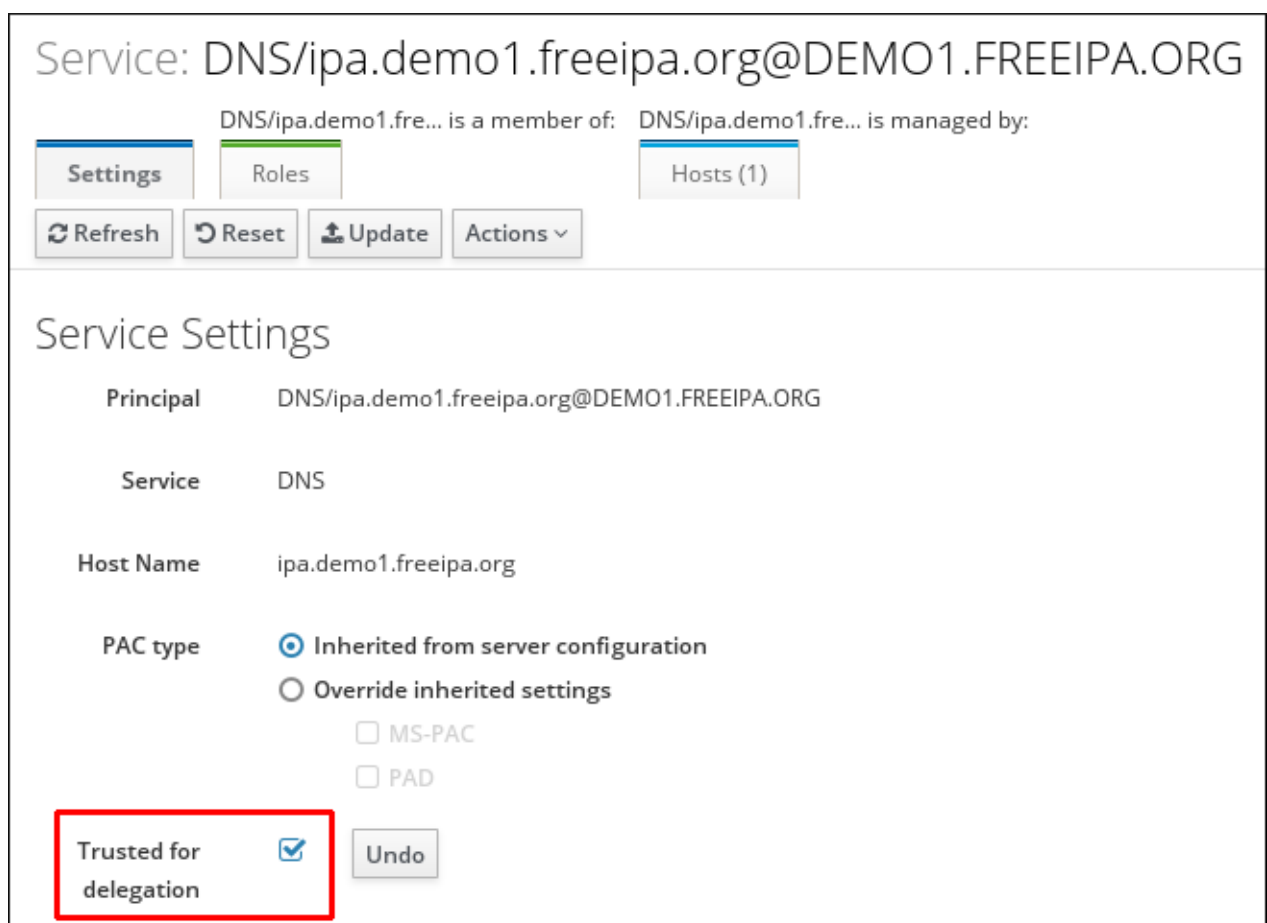
From the IdM web UI, you can currently only add the **OK\_AS\_DELEGATE** flag to a principal:

1. Select the **Services** subtab, accessible through the **Identity** main tab.



**Figure 18.1. List of Services**

2. Click on the service to which you want to add the flag.
3. Check the **Trusted for delegation** option.



**Figure 18.2. Adding the OK\_AS\_DELEGATE Flag**

### 18.4.2. Setting Kerberos Flags from the Command Line

To add a flag to a principal from the command line or to remove a flag, add one of the following options to the `ipa service-mod` command:

- ✱ **--ok-as-delegate** for **OK\_AS\_DELEGATE**
- ✱ **--requires-pre-auth** for **REQUIRES\_PRE\_AUTH**

To add a flag, set the corresponding option to **1**. For example, to add the **OK\_AS\_DELEGATE** flag to the *test/ipa.example.com@EXAMPLE.COM* principal:

```
$ ipa service-mod test/ipa.example.com@EXAMPLE.COM --ok-as-delegate=1
```

To remove a flag or to disable it, set the corresponding option to **0**. For example, to disable the **REQUIRES\_PRE\_AUTH** flag for the *test/ipa.example.com@EXAMPLE.COM* principal:

```
$ ipa service-mod test/ipa.example.com@EXAMPLE.COM --requires-pre-auth=0
```

To find out if **OK\_AS\_DELEGATE** is currently set for a principal, run the **kvno** utility and then the **klist -f** command. **OK\_AS\_DELEGATE** is represented by the **0** character in the **klist -f** output:

```
$ kvno test/ipa.example.com@EXAMPLE.COM
$ klist -f
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@EXAMPLE.COM

Valid starting Expires Service principal
02/19/2014 09:59:02 02/20/2014 08:21:33 test/ipa/example.com@EXAMPLE.COM
Flags: FATO
```

To find out what flags are currently set for a principal, use the **kadmin.local** utility. The current flags are displayed on the **Attributes** line of **kadmin.local** output, for example:

```
# kadmin.local
kadmin.local: getprinc test/ipa.example.com
Principal: test/ipa.example.com@EXAMPLE.COM
Expiration date: [never]
Last password change: Mon Sep 16 15:44:21 EDT 2013
Password expiration date: [none]
Maximum ticket life: 1 day 00:00:00
Maximum renewable life: 7 days 00:00:00
Last modified: Mon Oct 14 23:42:53 EDT 2013 (admin/admin@EXAMPLE.COM)
Last successful authentication: Wed Mar 11 08:01:03 EDT 2015
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 6
Key: vno 1, aes256-cts-hmac-sha1-96, no salt
Key: vno 1, aes128-cts-hmac-sha1-96, no salt
Key: vno 1, des3-cbc-sha1, no salt
Key: vno 1, arcfour-hmac, no salt
Key: vno 1, camellia128-cts-cmac, no salt
Key: vno 1, camellia256-cts-cmac, no salt
MKey: vno 1
Attributes: REQUIRES_PRE_AUTH OK_AS_DELEGATE OK_TO_AUTH_AS_DELEGATE
Policy: [none]
```

## 18.5. 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 System-Level Authentication Guide](#).

## 18.6. 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 19. Using sudo

Identity Management provides a mechanism for predictably and consistently applying **sudo** policies across the IdM domain. The **sudo** policies apply to domain users and domain hosts.

### 19.1. 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.

#### 19.1.1. 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 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/sudo-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.

#### 19.1.2. sudo and Netgroups

As [Section 19.1.1, “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.

In order for netgroups and **sudo**, which relies on netgroups, to function properly, the NIS domain name is required to be set. However, while **sudo** configuration requires NIS-formatted netgroups and that a NIS domain be named for netgroups, this NIS domain does not actually need to exist. It is not required to have a NIS server installed.



### Note

The client installation process, executed by the **ipa-client-install** command, sets a NIS domain name automatically to the IdM domain name by default.

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.

The Identity Management Directory Server instance uses the standard LDAP schema for NIS objects, defined in [RFC 2307](#).

### 19.1.3. Supported **sudo** Clients

Any system which is supported as an IdM client system can be configured as a **sudo** client in IdM.

## 19.2. Setting up **sudo** Commands and Command Groups

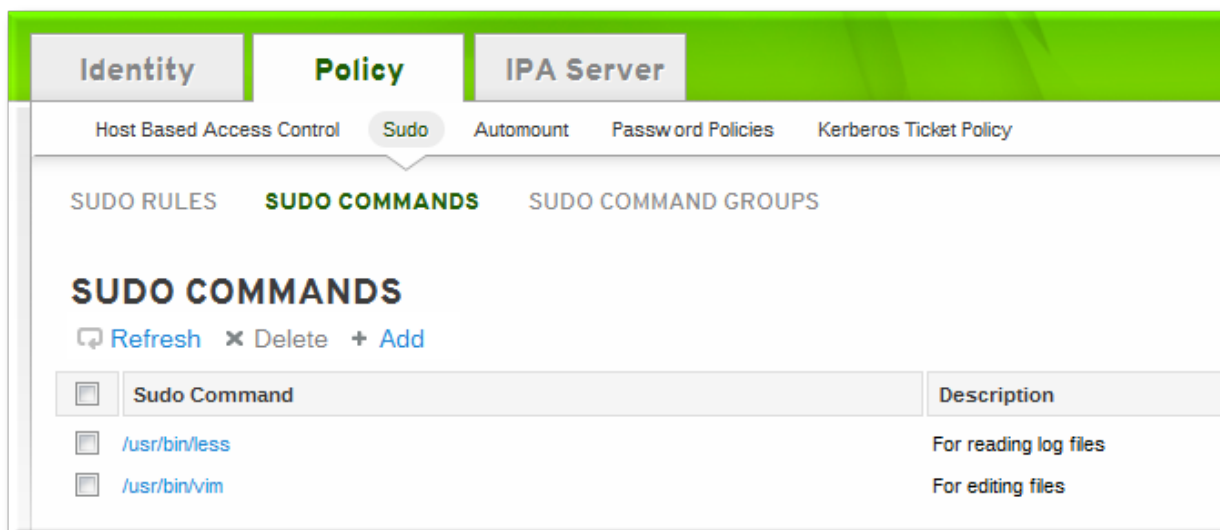
Just as in regular **sudo** configuration, any command which will be governed by **sudo** access must be listed in the configuration. Identity Management adds an extra control measure with *sudo command groups*, which allow a group of commands to be defined and then applied to the **sudo** configuration 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.

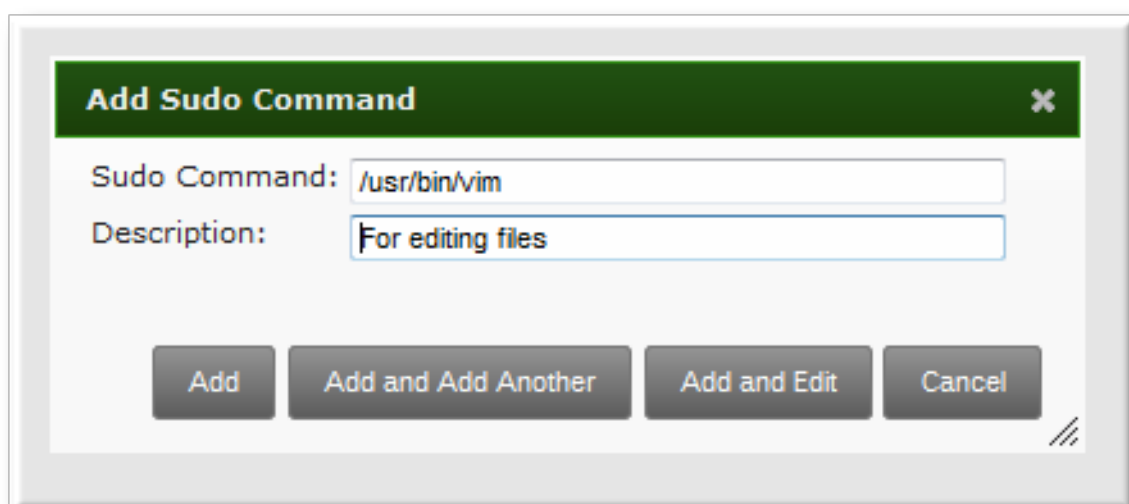
### 19.2.1. Adding **sudo** Commands

#### 19.2.1.1. 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 settings 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.

### 19.2.1.2. 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'
-----
```

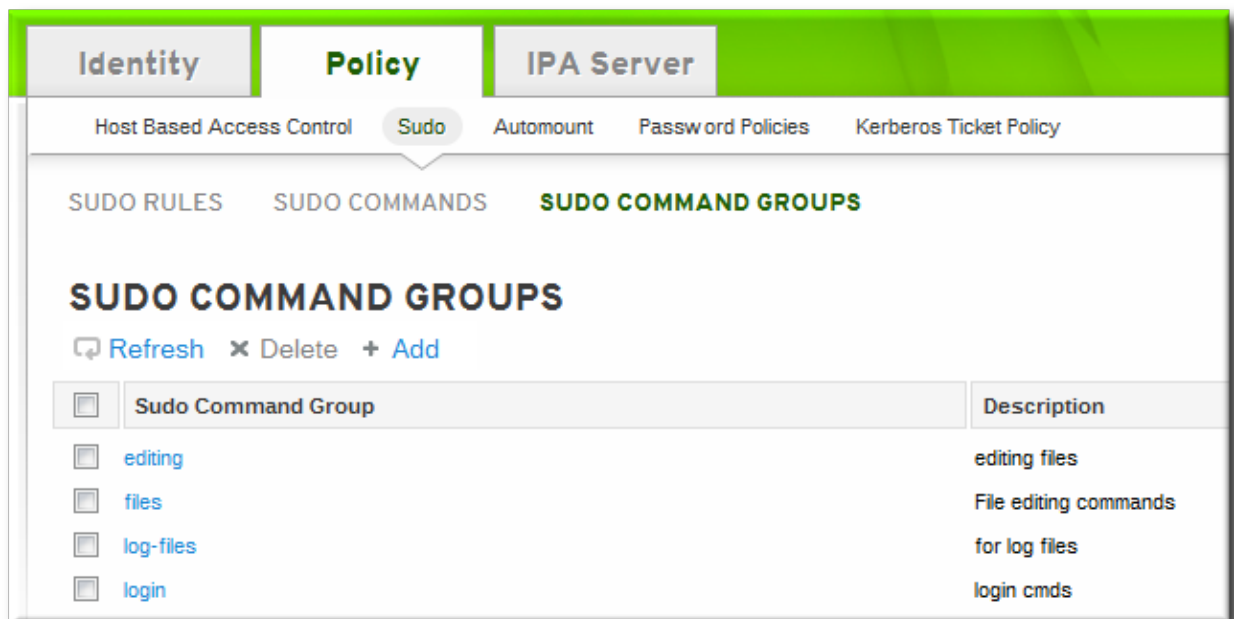


```
Added sudo command "/usr/bin/less"
-----
sudo Command: /usr/bin/less
Description: For reading log files
```

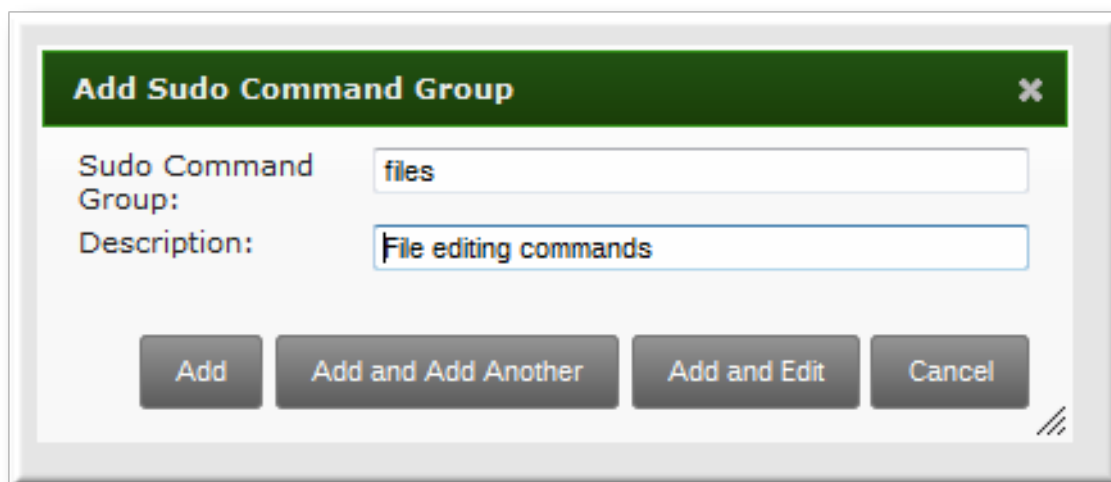
## 19.2.2. Adding sudo Command Groups

### 19.2.2.1. Adding sudo Command Groups with the Web UI

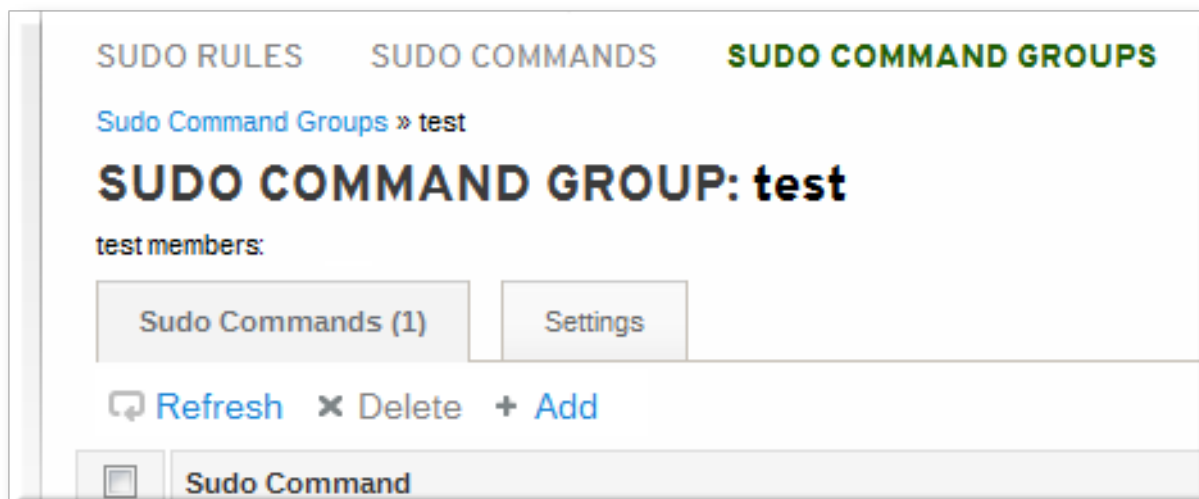
1. Click the **Policy** tab.
2. Click the **Sudo** subtab, and then select the **Sudo Command Groups** link.
3. Click the **Add** link at the top of the list of command groups.



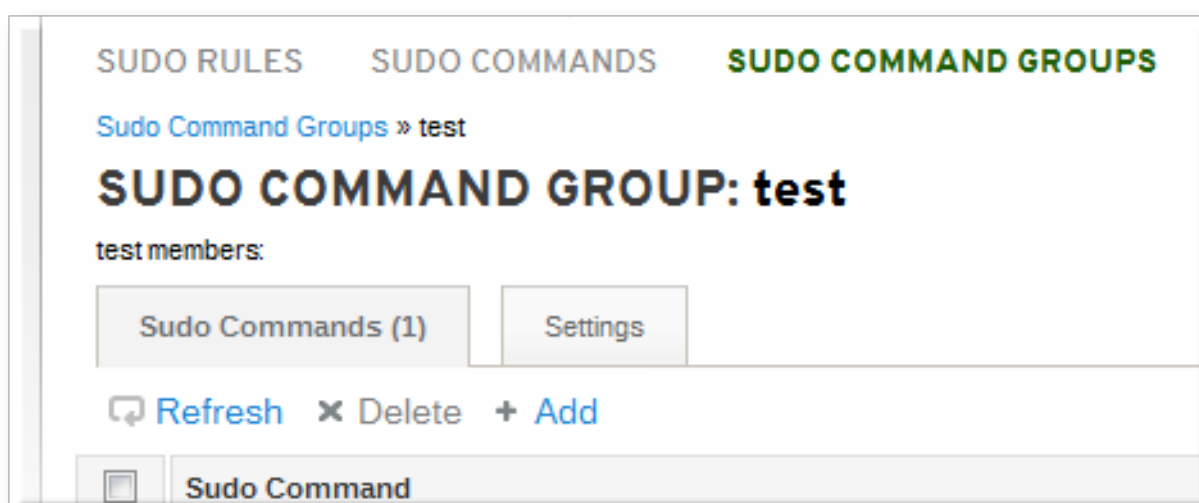
4. Enter the name and description for the new command group.



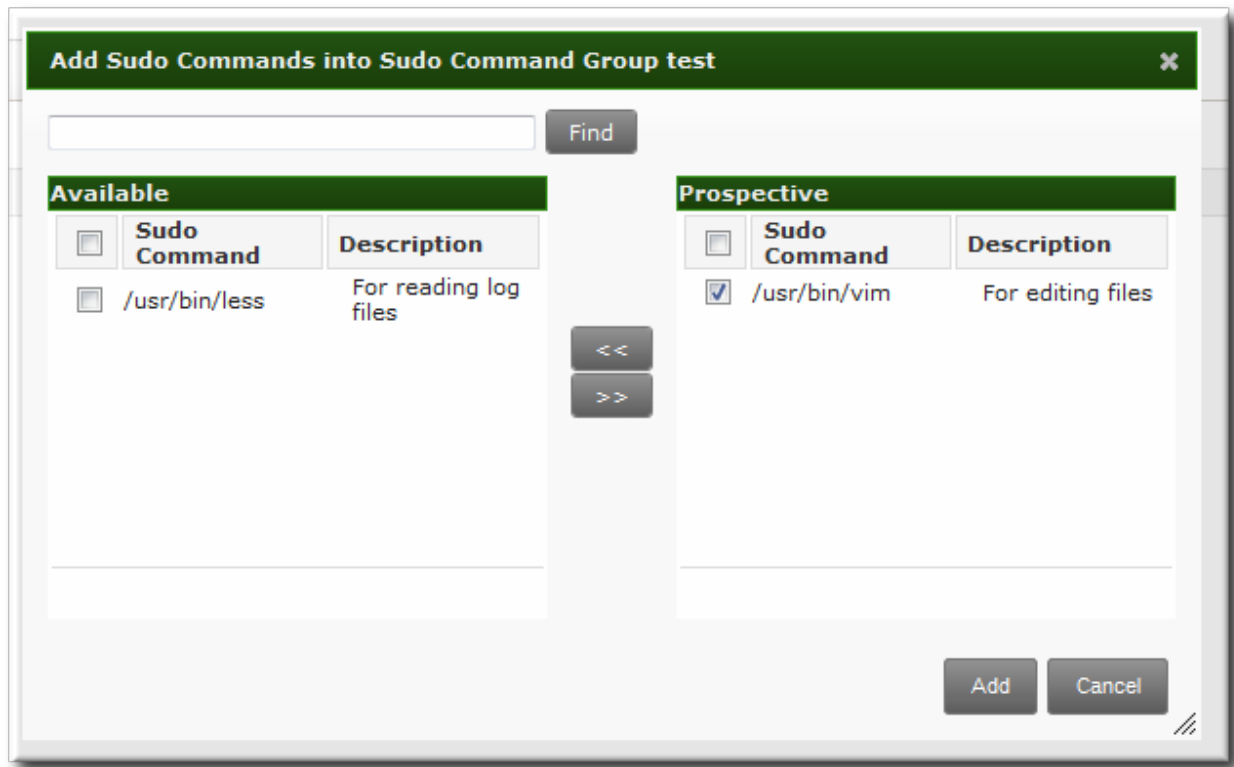
5. Click the **Add and Edit** button to go immediately to the settings pages for the group.
6. In the **Sudo Commands** tab, click the **Add** button to add a sudo command to the group.



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.

### 19.2.2.2. 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  
 -----

## 19.3. 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).

### 19.3.1. 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 a 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.

**Add Hosts into Sudo Rule files-commands** [X]

Filter available Hosts [Filter]

Available			Prospective	
<input type="checkbox"/>	Hosts	>	<input type="checkbox"/>	Hosts
<input type="checkbox"/>	qe-server.example.com		<input checked="" type="checkbox"/>	server.example.com
		<		

**External**

[Empty text input field]

[Add] [Cancel]

## Figure 19.1. 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.

### 19.3.2. About sudo Options Format

The **sudo** rule can be configured to use any supported **sudoers** options. For a complete list of options, see the `sudoers(5)` man page.

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. Instead, each option must be added individually, either through the UI or the command-line tools.

```
[jsmith@server ~]$ ipa sudorule-add-option readfiles
Sudo Option: mail_badpass
-----
Added option "mail_badpass" to Sudo rule "readfiles"
-----
[jsmith@server ~]$ ipa sudorule-add-option readfiles
Sudo Option: syslog=local2
-----
Added option "syslog=local2" to Sudo rule "readfiles"
-----
...
```

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"
```

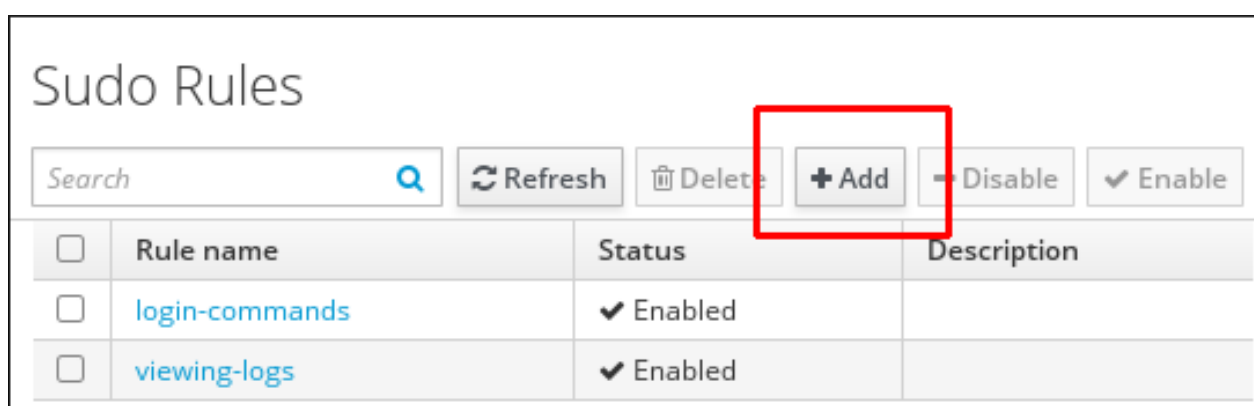
For example, the same command in the IdM command line has all of the variables on one line and no spaces around the equals sign.

```
[jsmith@server ~]$ ipa sudorule-add-option readfiles
Sudo Option: env_keep="COLORS DISPLAY EDITOR HOSTNAME HISTSIZE INPUTRC
KDEDIR LESSSECURE LS_COLORS MAIL PATH PS1 PS2 ... XAUTHORITY"
```

To use multiple **sudoers** options in Identity Management, configure each one as a separate option setting, rather than all on one line.

### 19.3.3. Defining sudo Rules in the Web UI

1. Click the **Policy** tab.
2. Click the **Sudo** subtab, and then select **Sudo Rules**.
3. Click the **Add** link at the top of the list of sudo rules.



**Figure 19.2. Adding a New sudo Rule**

4. Enter the name for the rule.

The screenshot shows the 'Add Sudo Rule' dialog box. It has a title bar with a close button. Inside, there is a 'Rule name' field with a blue asterisk indicating it is required. The field contains the text 'files-commands'. Below the field, there is a note '\* Required field'. At the bottom, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

**Figure 19.3. Naming a New sudo 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.

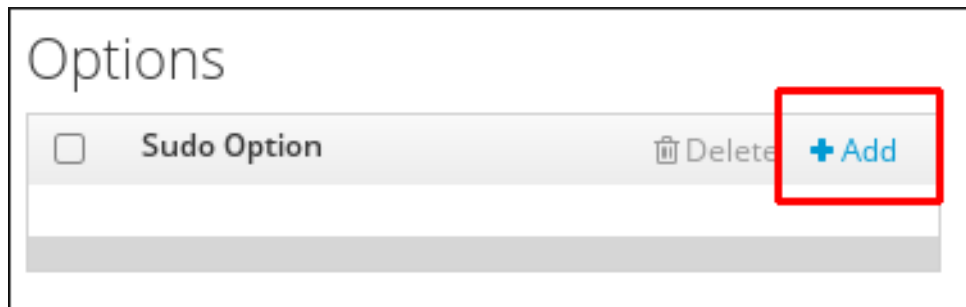
6. *Optional.* In the **Options** area, add any **sudoers** options.



## Note

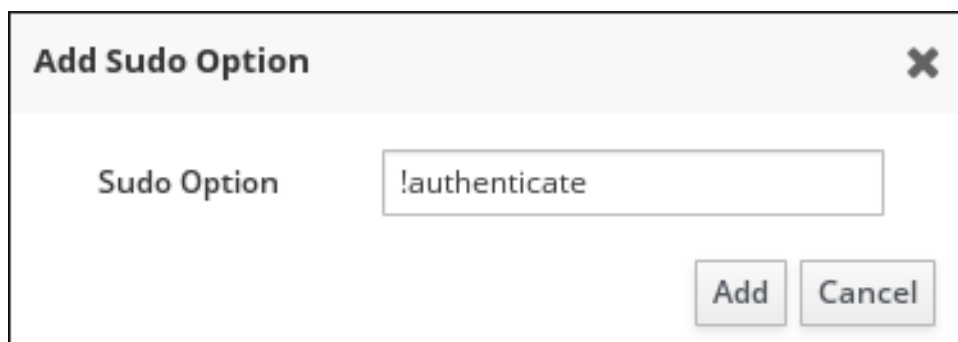
As described in [Section 19.3.2, “About sudo Options Format”](#), do not use options with whitespace in the values. Rather than adding a list of options in one line, add a single option setting for each desired option.

- a. Click the **Add** link at the right of the options list.



**Figure 19.4. Adding a sudo Option**

- b. Enter the **sudoers** option.



**Figure 19.5. Entering a 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.

Who

User category the rule applies to: ☐ Anyone ☒ Specified Users and Groups

<input type="checkbox"/>	Users	External	Delete	+ Add
<input type="checkbox"/>	manager			
<input type="checkbox"/>	employee			
<input type="checkbox"/>	helpdesk			

<input type="checkbox"/>	User Groups	Delete	+ Add
<input type="checkbox"/>	admins		

**Figure 19.6. Adding Users to a sudo Rule**

- b. Click the checkbox by the users to add to the rule, and click the right arrow button to move the users to the selection box.

Add Users into Sudo Rule files-commands

Filter available Users  Filter

Available

<input type="checkbox"/>	Users
<input type="checkbox"/>	xyz

External

Prospective

<input type="checkbox"/>	Users
<input checked="" type="checkbox"/>	employee
<input checked="" type="checkbox"/>	helpdesk
<input checked="" type="checkbox"/>	manager

Add Cancel

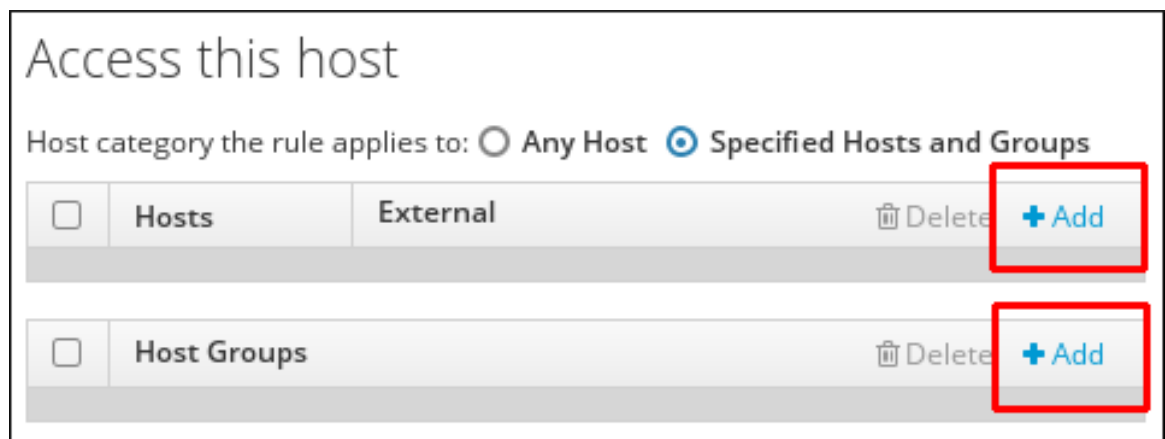
**Figure 19.7. Selecting Users for a sudo Rule**

- c. Click **Add**.

It is possible to configure both IdM users and external system users ([Section 19.3.1, “About External Users”](#)).

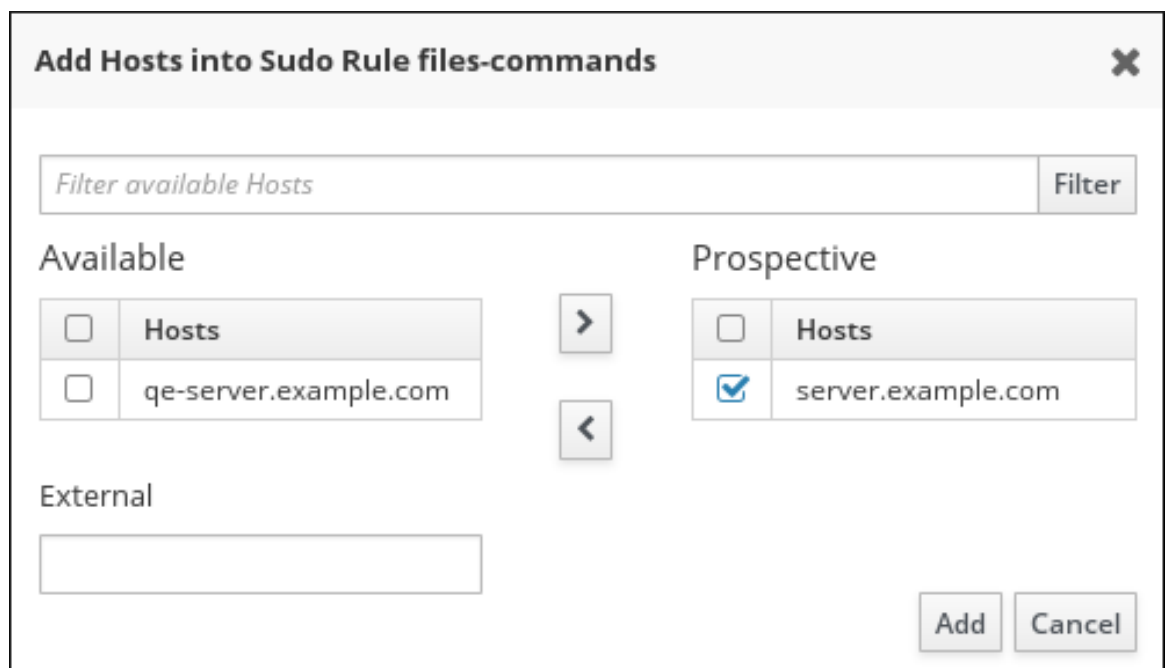


8. In the **Access This Host** area, select the hosts on which the sudo rule is in effect.
  - a. Click the **Add** link at the right of the hosts list.



**Figure 19.8. Adding Hosts to a sudo Rule**

- b. Click the checkbox by the hosts to include with the rule, and click the right arrow button to move the hosts to the selection box.



**Figure 19.9. Selecting Hosts for a sudo Rule**

- 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.

Run Commands

Command category the rule applies to: ☐ Any Command ☒ Specified Commands and Groups

Allow

<input type="checkbox"/>	Sudo Allow Commands	Delete	+ Add
<input type="checkbox"/>	Sudo Allow Command Groups	Delete	+ Add

Deny

<input type="checkbox"/>	Sudo Deny Commands	Delete	+ Add
<input type="checkbox"/>	Sudo Deny Command Groups	Delete	+ Add

**Figure 19.10. Adding Commands to a sudo Rule**

- b. Click the checkbox by the commands or command groups to include with the rule, and click the right arrow button to move the commands to the selection box.

Add Allow Sudo Commands into Sudo Rule files-commands

Filter available Sudo Commands Filter

Available

<input type="checkbox"/>	Sudo Commands
<input type="checkbox"/>	editing
<input type="checkbox"/>	log-files
<input type="checkbox"/>	login

Prospective

<input type="checkbox"/>	Sudo Commands
<input checked="" type="checkbox"/>	files

Add Cancel

**Figure 19.11. Selecting Commands for a sudo Rule**

- 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.

**As Whom**

RunAs User category the rule applies to: ☐ Anyone ☒ Specified Users and Groups

<input type="checkbox"/>	RunAs Users	External	Delete	+ Add
<input type="checkbox"/>	Groups of RunAs Users		Delete	+ Add

RunAs Group category the rule applies to: ☐ Any Group ☒ Specified Groups

<input type="checkbox"/>	RunAs Groups	External	Delete	+ Add
--------------------------	--------------	----------	--------	-------

**Figure 19.12. Configuring sudo Rules to Execute Commands as a Specific User**

- b. Click the checkbox by the users to run the command as, and click the right arrow button to move the users to the selection box.

**Add RunAs Users into Sudo Rule files-commands** ✕

Filter available Users  Filter

**Available**

<input type="checkbox"/>	Users
<input type="checkbox"/>	employee
<input type="checkbox"/>	helpdesk

**Prospective**

<input type="checkbox"/>	Users
<input checked="" type="checkbox"/>	admin
<input checked="" type="checkbox"/>	manager

External

Add Cancel

**Figure 19.13. Selecting Users for the Command**

- c. Click **Add**.

### 19.3.4. Defining sudo Rules in the Command Line

Each element is added to the rule command using a different command (listed in [Table 19.1, “sudo Commands”](#)).

The basic outline of a **sudo** rule command is:

```
$ ipa sudorule-add* options ruleName
```

### Example 19.1. 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.

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ 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 **--sudocmds**, or a group of commands, using **--sudocmdgroups**.

```
[jsmith@server ~]$ ipa sudorule-add-allow-command --sudocmds
"/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.

```
[jsmith@server ~]$ 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.

```
[jsmith@server ~]$ 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 19.2. 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:

```

[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa sudorule-add-allow-command --sudocmds
"/usr/bin/less" readfiles
[jsmith@server ~]$ ipa sudorule-add-allow-command --sudocmds
"/usr/bin/tail" readfiles
[jsmith@server ~]$ ipa sudorule-add-deny-command --sudocmds
"/usr/bin/vim" readfiles

```

### Example 19.3. 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](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:

```

[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa sudorule-add-option readfiles
Sudo Option: !authenticate
-----
Added option "!authenticate" to Sudo rule "readfiles"
-----

```



### Note

As described in [Section 19.3.2, “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 19.4. 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 19.5. 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 19.3.1, "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 19.1. sudo Commands**

Command	Description
<b>sudorule-add</b>	Add a sudo rule entry.
<b>sudorule-add-user</b>	Add 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.
<b>sudorule-add-host</b>	Add a target host for the rule. These are the hosts where the users are granted sudo permissions.
<b>sudorule-add-runasgroup</b>	Set a group to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using <b>sudo-rule</b> .

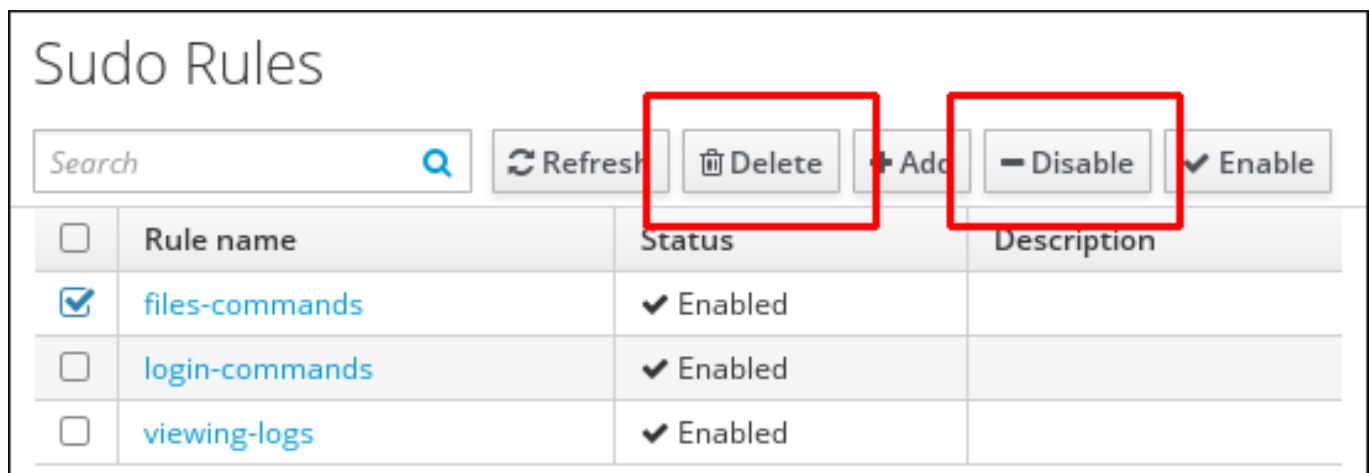
Command	Description
<code>sudo rule-add-runasuser</code>	Set a user to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using <b>sudo-rule</b> .
<code>sudo rule-add-allow-command</code>	Add a command that users in the rule have sudo permission to run.
<code>sudo rule-add-deny-command</code>	Add a command that users in the rule are explicitly <i>denied</i> sudo permission to run.
<code>sudo rule-add-option</code>	Add a sudoers flag to the sudo rule.
<code>sudo rule-disable</code>	Temporarily deactivate a sudo rule entry.
<code>sudo rule-enable</code>	Activate a previously suspended sudo rule.
<code>sudo rule-del</code>	Remove a sudo rule entirely.

### 19.3.5. Suspending and Removing sudo Rules

Defined **sudo** rules can either be temporarily deactivated or entirely deleted from the web UI or from the command line. Suspended rules are removed from the **ou=sudoers** compat tree without a need for a server restart.

#### Suspending and Removing sudo Rules from the Web UI

To suspend or completely delete a rule from the web UI, use the **Disable** or **Delete** buttons at the top of the list of **sudo** rules:



**Figure 19.14. Suspending or Deleting a sudo Rule from the Web UI**

#### Suspending and Removing sudo Rules from the Command Line

To suspend a rule from the command line, run a command such as the following:

```
ipa sudorule-disable files-commands
```

To completely delete a rule from the command line, run a command such as the following:

```
ipa sudorule-del files-commands
```

## 19.4. 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 for its policies.

You can apply **sudo** policies to hosts using SSSD or LDAP. Red Hat strongly recommends to use the SSSD-based configuration.

### 19.4.1. Applying the **sudo** Policies to Hosts Using SSSD

1. Set up the host and **sudo** entries in IdM.
  - a. Set up the **sudo** commands and command groups, as described in [Section 19.2, “Setting up sudo Commands and Command Groups”](#).
  - b. Set up the **sudo** rules, as described in [Section 19.3, “Defining sudo Rules”](#).
  - c. *Optional.* Set up a host group, as described in [Section 11.7, “Managing Host Groups”](#).
  - d. *Optional.* Create a user group and add the users, as described in [Section 10.10.2.1, “Creating User Groups”](#).
2. Configure every system in the IdM domain to use SSSD for **sudo** rules.



#### Note

Only perform this step on systems based on Red Hat Enterprise Linux 7.0. In Red Hat Enterprise Linux 7.1 and later, the **ipa-client-install** utility configures SSSD as the data provider for **sudo** automatically.

- a. Configure **sudo** to look to SSSD for the **sudoers** 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. 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. Enable the **rhel-domainname** service if it is not already enabled to ensure that the NIS domain name will be persistent across reboots.

```
[root@server ~]# systemctl enable rhel-  
domainname.service
```

- b. Set the NIS domain name to use with the **sudo** rules.

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

- c. Configure the system authentication settings to persist the NIS domain name. For example:

```
[root@server ~]# echo "NISDOMAIN=example.com.com" >>  
/etc/sysconfig/network
```

This updates the **/etc/sysconfig/network** and **/etc/yp.conf** files with the NIS domain.

3. 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.

### 19.4.2. Applying the sudo Policies to Hosts Using LDAP



#### Important

Only use the LDAP-based configuration for clients that do not use SSSD. Red Hat recommends to configure all other clients using the SSSD-based configuration, as described in [Section 19.4.1, “Applying the sudo Policies to Hosts Using SSSD”](#).

For information on applying **sudo** policies using LDAP, see the [Identity Management Guide for Red Hat Enterprise Linux 6](#).

The LDAP-based configuration is expected to be used primarily for clients based on Red Hat Enterprise Linux versions earlier than Red Hat Enterprise Linux 7. It is therefore only described in the documentation for Red Hat Enterprise Linux 6.

## Chapter 20. 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.

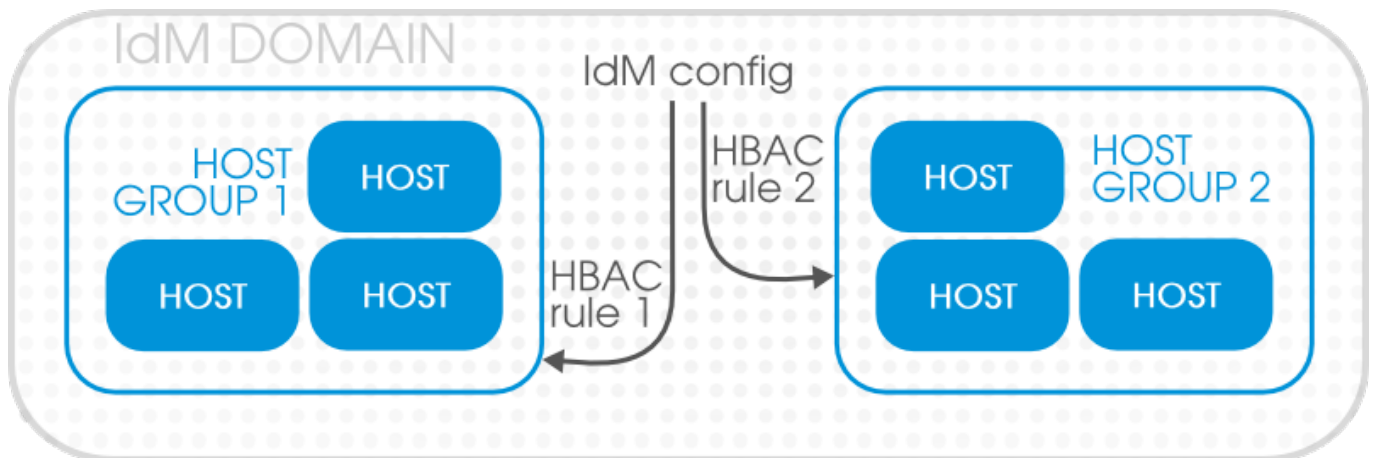


### Note

Using host-based access control requires SSSD to be installed and configured on the IdM client machine.

### 20.1. About Host-Based Access Control

Host-based access control rules (which are described in [Chapter 20, 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 20.1. 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).



### Note

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 added to the host-based access control configuration first to make it available to rules, and then added to the access control rules.

## 20.2. Creating Host-Based Access Control Entries for Services and Service Groups

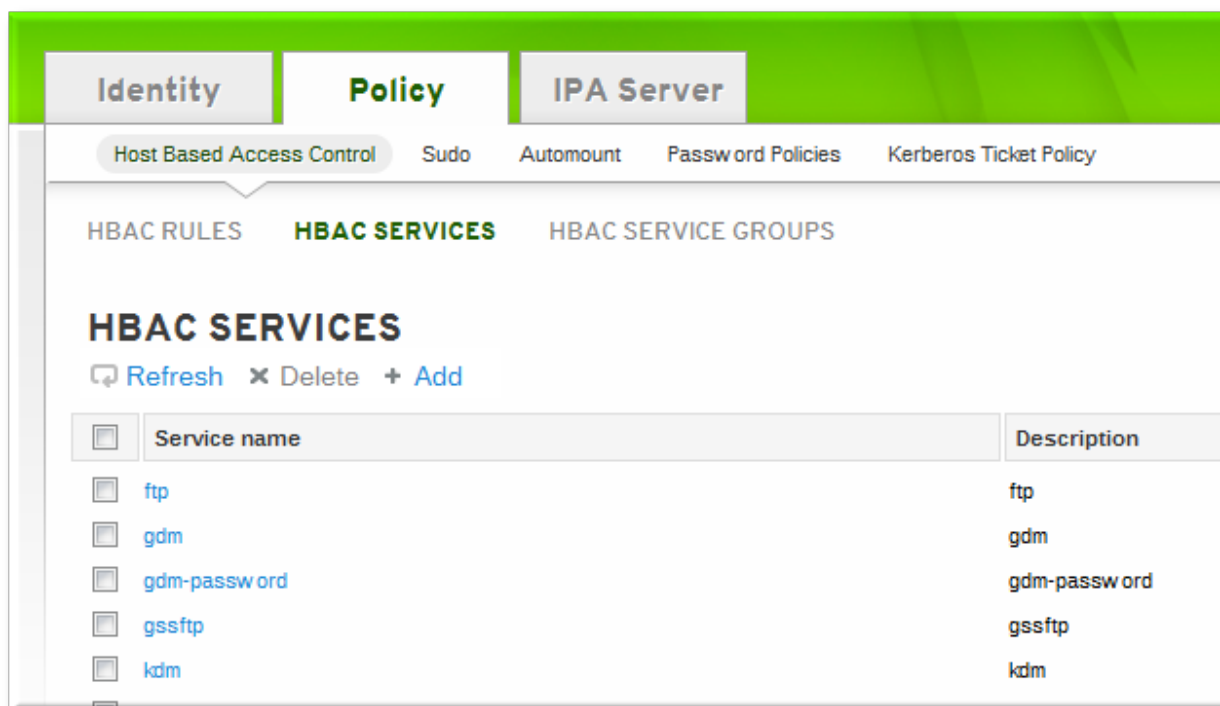
Any PAM service can be identified as to the host-based access control (HBAC) system in IdM. The service entries used in host-based access control are separate from adding a service to the IdM domain. Adding a service to the domain makes it a recognized resource which is available to other resources. 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.

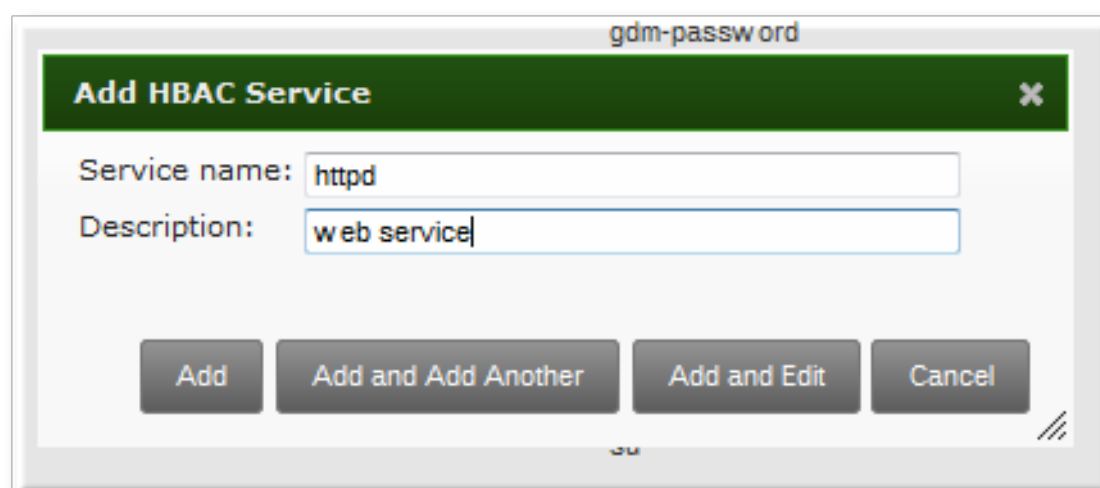
### 20.2.1. Adding HBAC Services

#### 20.2.1.1. 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 20.2.2.1, "Adding Service Groups in the Web UI"](#).

### 20.2.1.2. 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"
-----
Service name: tftp
Description: TFTP service
```

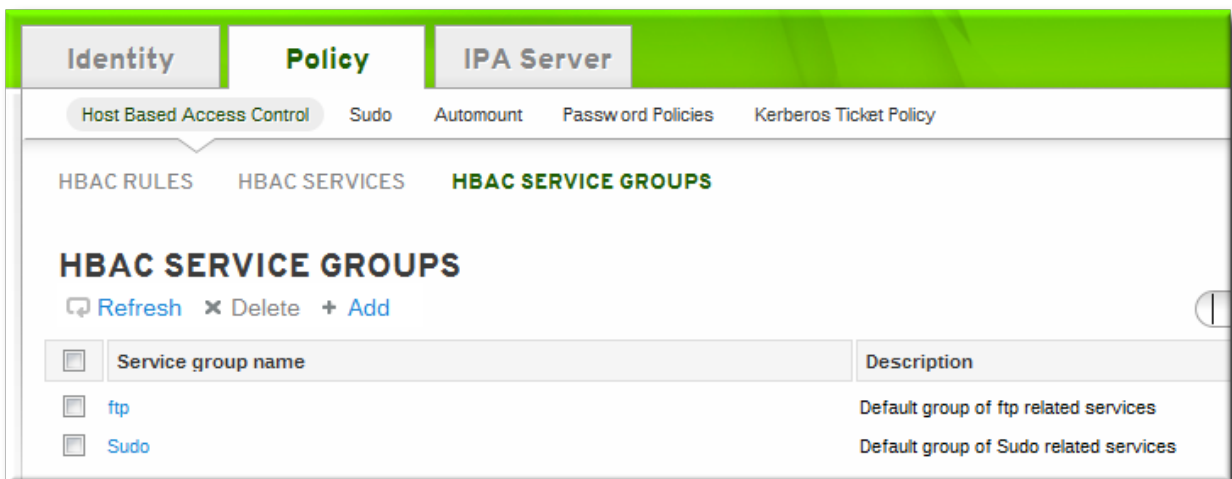
If a service group already exists, then the service can be added to the group using the `hbacsvgroup-add-member` command, as in [Section 20.2.2.2, “Adding Service Groups in the Command Line”](#).

## 20.2.2. Adding Service Groups

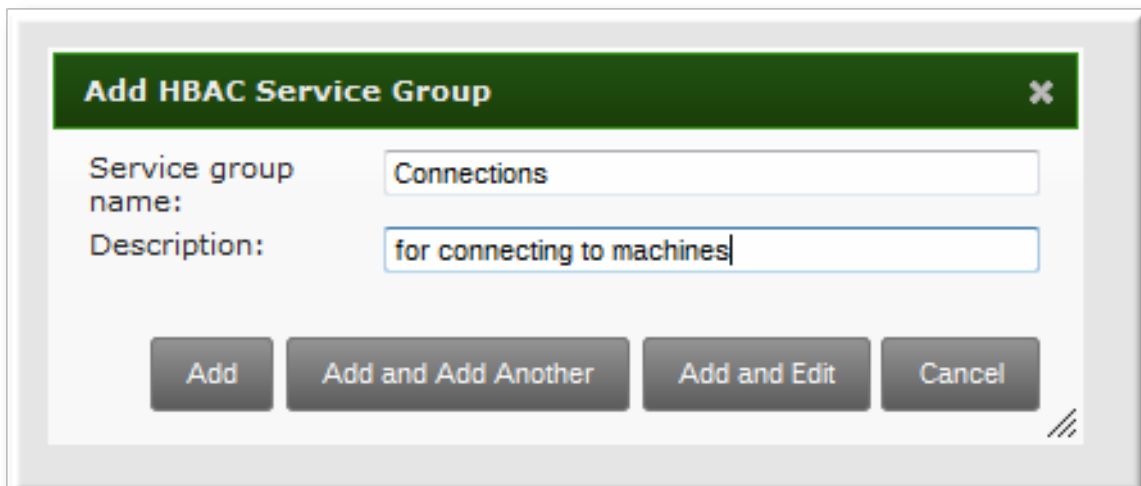
Once the individual service is added, it can be added to the access control rule. However, if there is a large number of services, then it can require frequent updates to the access control rules as services change. Identity Management also allows groups of services to be added to access control rules. 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.

### 20.2.2.1. Adding Service Groups in the Web UI

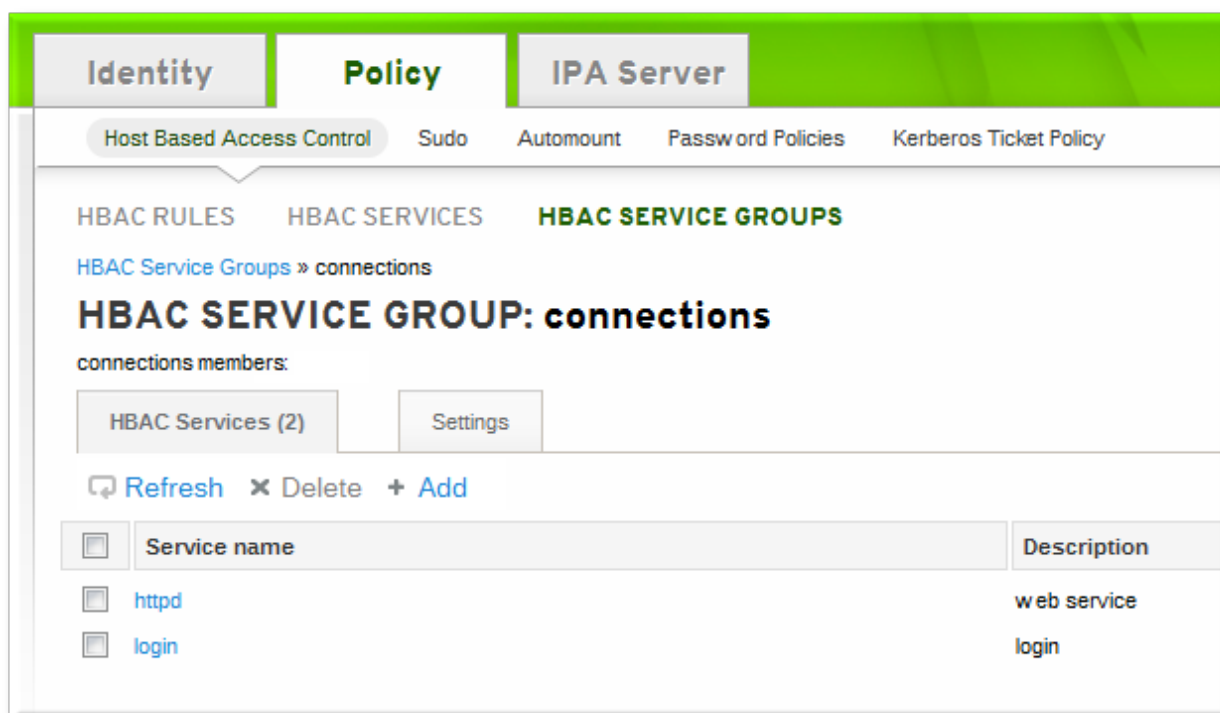
1. Click the **Policy** tab.
2. Click the **Host-Based Access Control** subtab, and then select the **HBAC Service Groups** link.
3. Click the **Add** link at the top of the list of service groups.



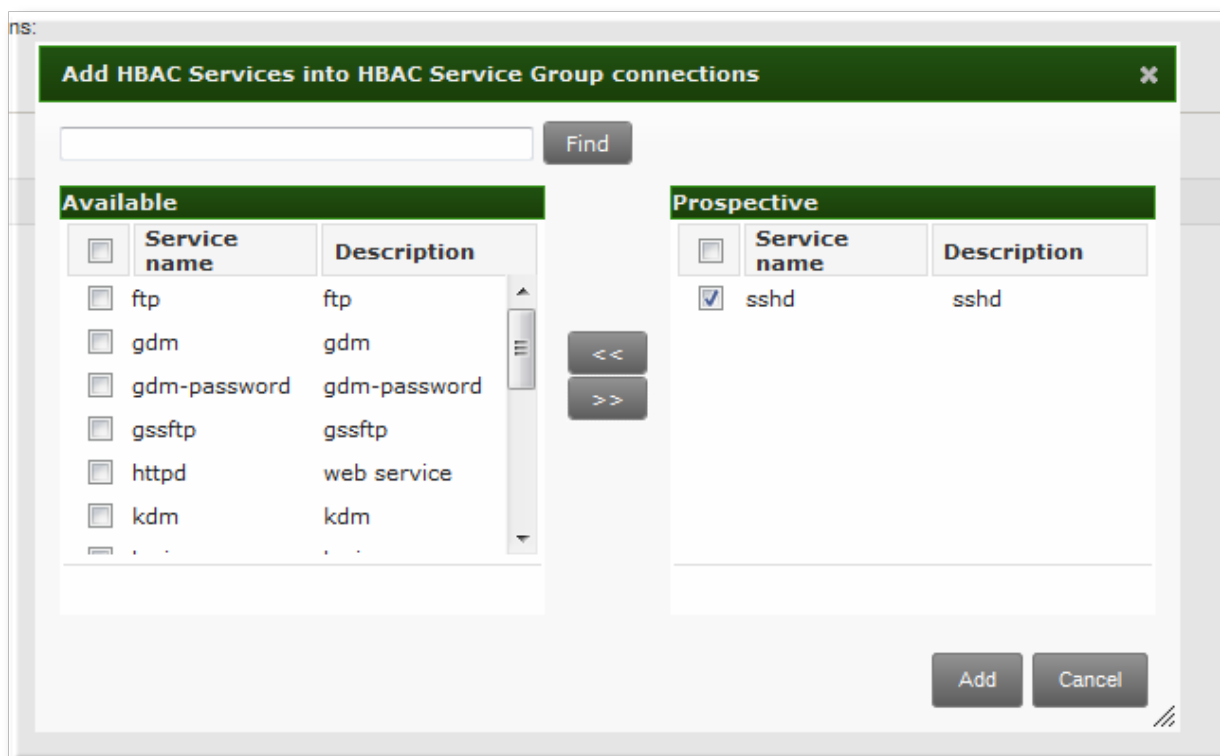
4. Enter the service group name and a description.



- Click the **Add and Edit** button to go immediately to the service group configuration page.
- At the top of the **HBAC Services** tab, click the **Add** link.



- 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.



- Click the **Add** button to save the group membership.

#### 20.2.2.2. 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:

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa hbacsvgroup-add --desc="login services" login
-----
Added HBAC service group "login"
-----
Service group name: login
Description: login services

[jsmith@server ~]$ ipa hbacsvc-add --desc="SSHD service" sshd
-----
Added HBAC service "sshd"
-----
Service name: sshd
Description: SSHD service

[jsmith@server ~]$ ipa hbacsvgroup-add-member --hbacsvcs=sshd login
Service group name: login
Description: login services
-----
Number of members added 1
-----
```



## Note

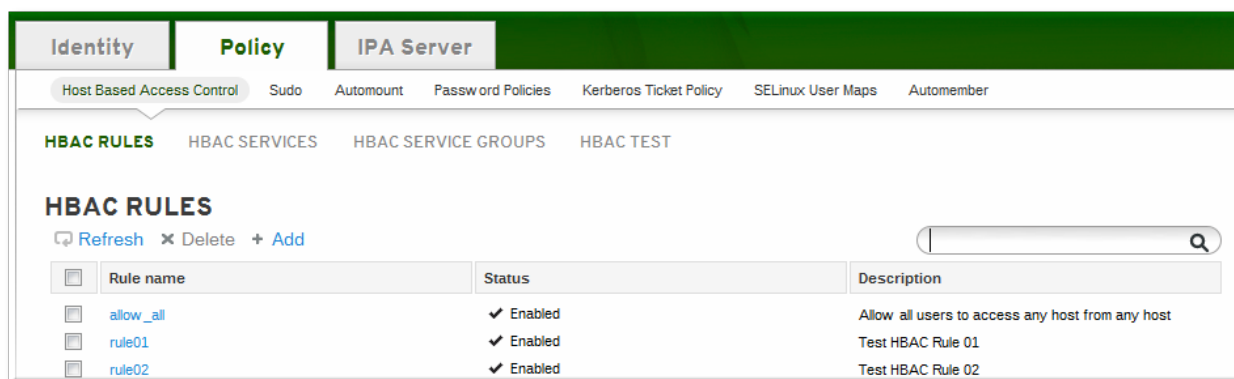
IdM defines two default service groups: **SUDO** for sudo services and **FTP** for services which provide FTP access.

## 20.3. Defining Host-Based Access Control Rules

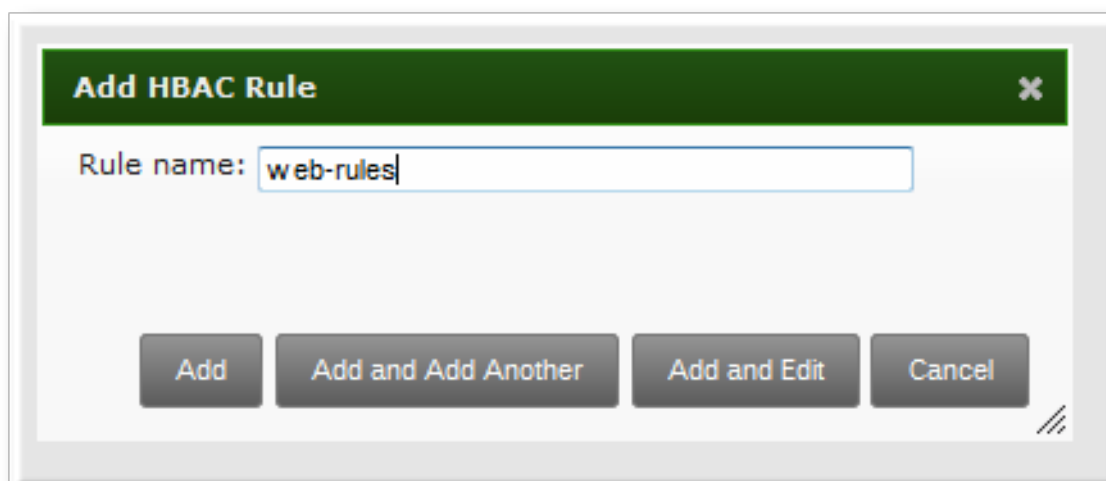
Access controls, at a high level, define *who* has access to *what*. The *who* is an IdM user, and the *what* can be either a host (target host), service, or service group, or a combination of the three.

### 20.3.1. 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.

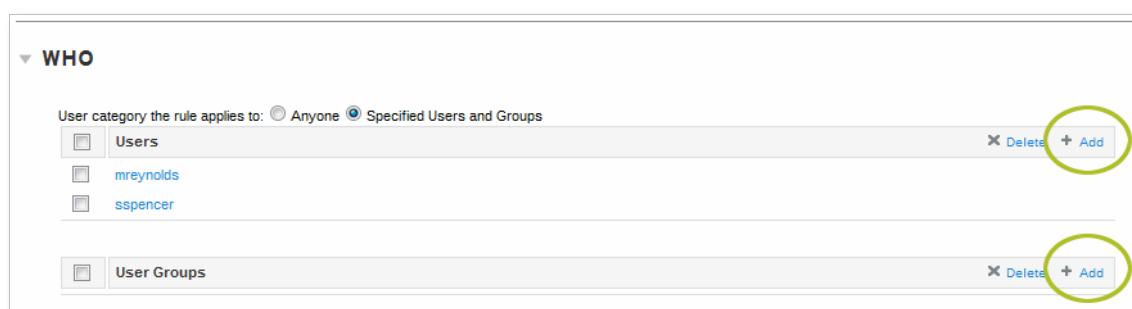
There are a number of configuration areas for the rule. The four basic elements are *who* the rule applies to, what hosts allow access (the target), and, optionally, what services can be accessed.

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:

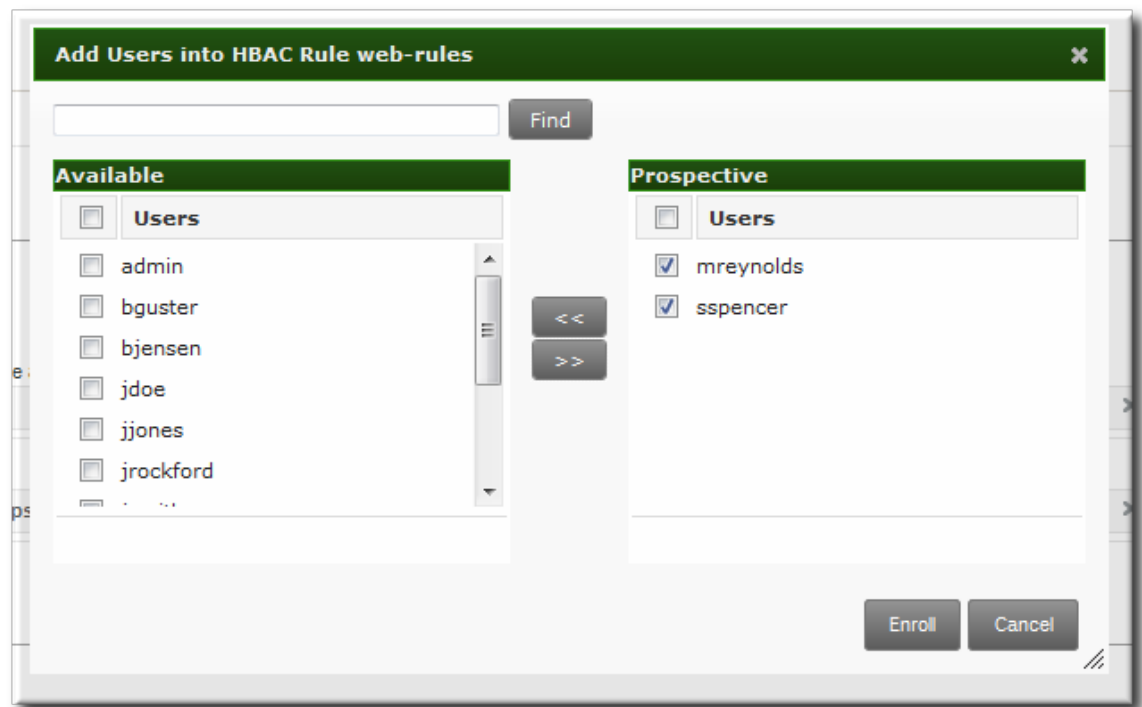
- a. Select the **Specified Users and Groups** radio button.
- b. Click the **+ Add** link at the right of the users list.



- c. 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.



button, >>, to move the users to the selection box.



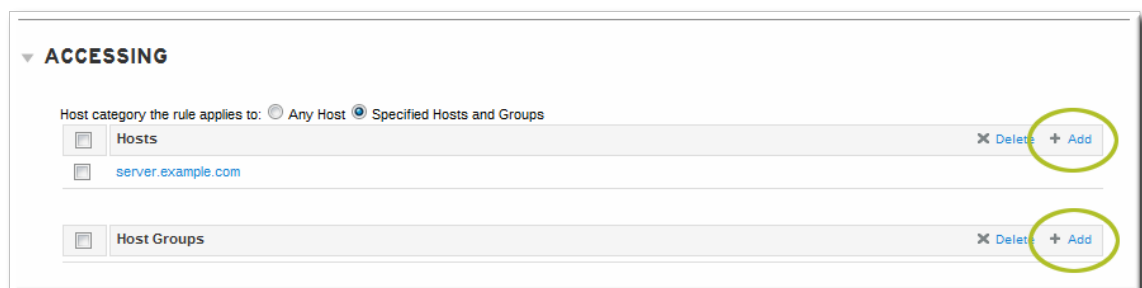
d. Click **Add**.

7. In the **Accessing** area, select the target hosts which can be accessed through this access control rule.

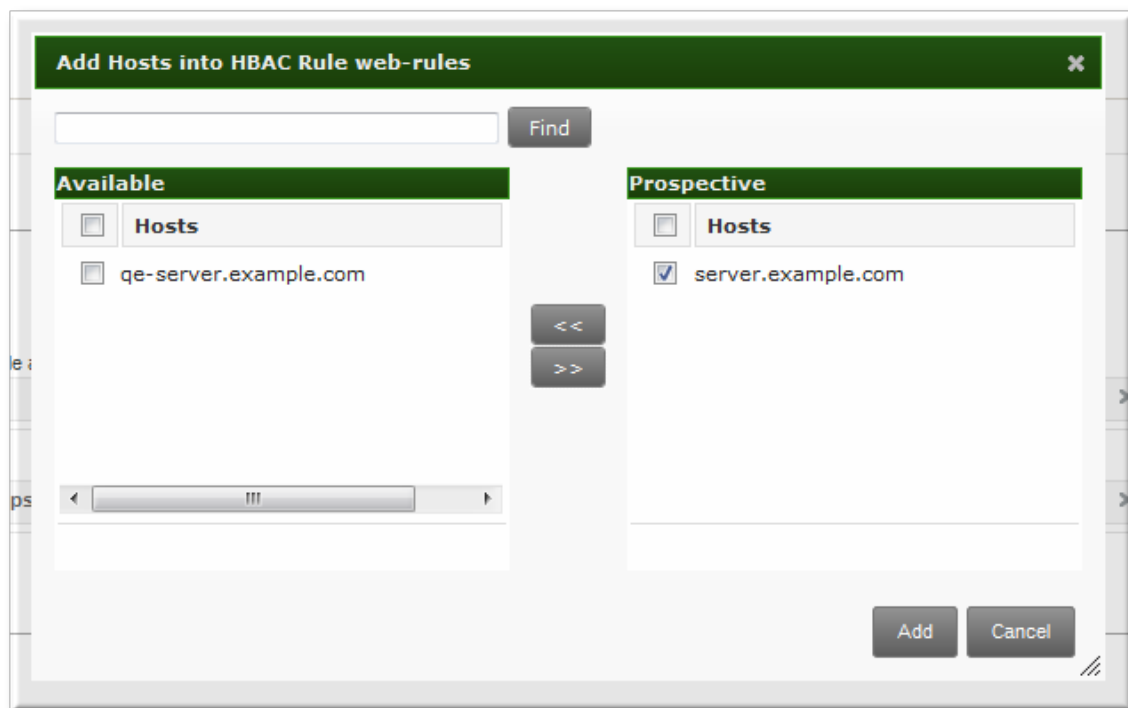
To apply the rule to all IdM hosts, select the **Any Host** radio button.

To apply the rule to a specific set of hosts or host groups:

- a. Select the **Specified Hosts and Groups** radio button.
- b. Click the **+ Add** link at the right of the hosts list.



- c. 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.



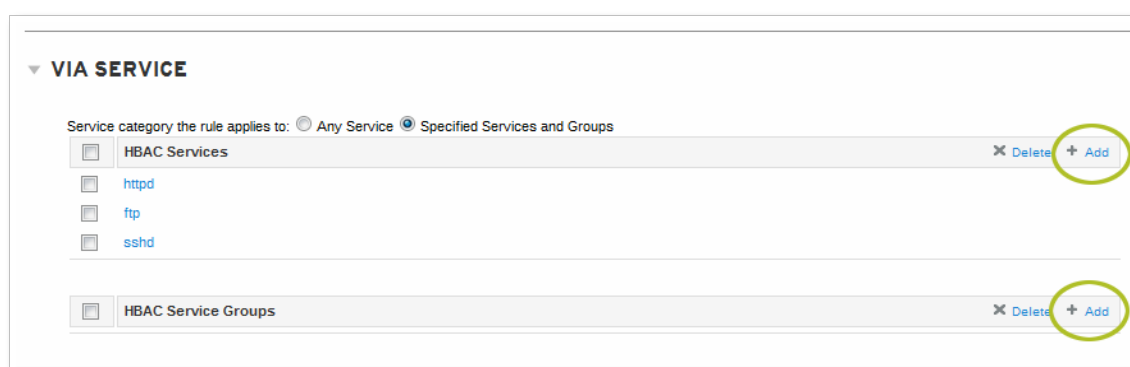
d. Click **Add**.

8. In the **Via Service** area, select specific services on the target hosts which the users are allowed to use to access target machines.

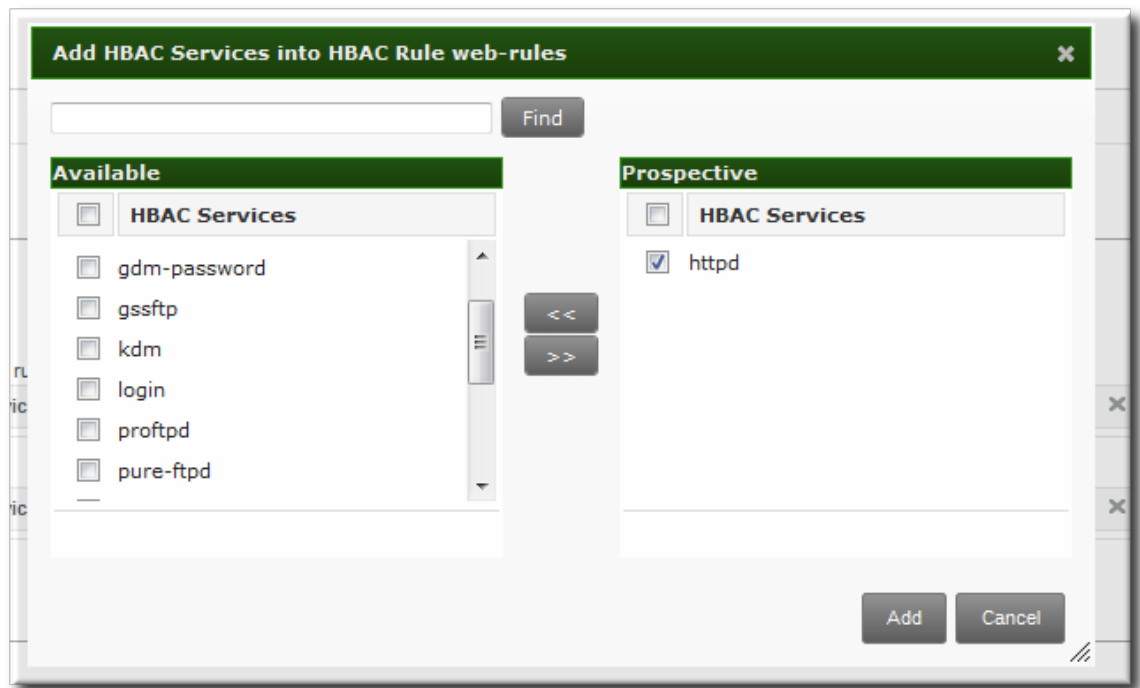
To apply the rule to all IdM hosts, select the **Any Service** radio button.

To apply the rule to a specific set of hosts or host groups:

- a. Select the **Specified Services and Groups** radio button.
- b. 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**.

### 20.3.2. Setting Host-Based Access Control Rules in the Command Line

Access control rules are created using the **hbacrule-\*** commands (listed in [Table 20.1, “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
```



#### Note

To set every user or every host as a target, use the category options, such as **--usercat=all**.

#### Example 20.1. 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 20.2. 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 20.3. 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 20.1. Host-Based Access Control Command and Options**

Command	Description	Arguments	Source or Target Entry
---------	-------------	-----------	------------------------

Command	Description	Arguments	Source or Target Entry
hbacrule-add	Adds a new host-based access control rule.	<ul style="list-style-type: none"> <li>✱ --usercat=all, which applies the rule to every user</li> <li>✱ --hostcat=all, which sets every host as an allowed target server</li> <li>✱ --servicecat=all, which sets every configured service as an allowed target service</li> <li>✱ <i>ruleName</i>, which is the required unique identifier for the new rule</li> </ul>	
hbacrule-add-host	Adds a target host to the access control rule. A target host can be accessed by other servers and users in the domain.	<ul style="list-style-type: none"> <li>✱ --hosts, which adds an individual server or command-separated list of servers as an allowed target server</li> <li>✱ --hostgroups, which adds a host group to the rule and every host within the host group is an allowed target server</li> <li>✱ <i>ruleName</i>, which is the rule to which to add the target server</li> </ul>	Target

Command	Description	Arguments	Source or Target Entry
hbacrule-add-service	Adds a service type to the rule.	<ul style="list-style-type: none"> <li>✳ <code>--hbacsvcs</code>, which adds an individual service type or a list of service types as an allowed target service</li>   <li>Lists of entries can be set by using the option multiple times with the same command invocation or by listing the options in a comma-separated list inside curly braces, such as <code>--option={val1,val2,val3}</code>.</li>   <li>✳ <code>--hbacsvcgroups</code>, which adds a service group to the rule and every service within the service group is an allowed target service</li>   <li>Lists of entries can be set by using the option multiple times with the same command or by listing the options in a comma-separated list inside curly braces, such as <code>--option={val1,val2,val3}</code>.</li>   <li>✳ <code>ruleName</code>, which is the rule to which to add the target service</li> </ul>	Target

Command	Description	Arguments	Source or Target Entry
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.	<ul style="list-style-type: none"> <li>✱ <code>--users</code>, which adds an individual user or command-separated list of users to the rule</li> <li>✱ <code>--groups</code>, which adds a user group to the rule and, thus, every user within the group</li> <li>✱ <code>ruleName</code>, which is the rule to which to add the user</li> </ul>	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).	<code>ruleName</code> , which is the rule to disable or enable	

## 20.4. 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.

### 20.4.1. 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.

### 20.4.2. 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 20.4. 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.

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa hbactest --user=jsmith --
host=target.example.com --service=ssh
-----
Access granted: True
-----
Matched rules: allow_all
Matched rules: sshd-jsmith
Matched rules: web-rules
Not matched rules: allGroup
```

### Example 20.5. Testing a Specific Rule

It is possible to check a specific rule (or several rules).

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ ipa hbactest --user=jsmith --
host=target.example.com --service=ssh --rules=myrule
-----
Access granted: True
-----
notmatched: myrule
```

### Example 20.6. 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.

```
[jsmith@server ~]$ kinit admin
[jsmith@server ~]$ 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 **--disabled** 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.

## 20.4.3. Testing Host-Based Access Control Rules in the UI

As [Section 20.4.1, “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.

**Figure 20.2. 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 20.3. 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 21. 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 system applications 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 (existing) SELinux user contexts and grant or restrict access to clients and services within the IdM domain, per host, based on the defined SELinux policies.

### 21.1. About Identity Management, SELinux, and Mapping Users

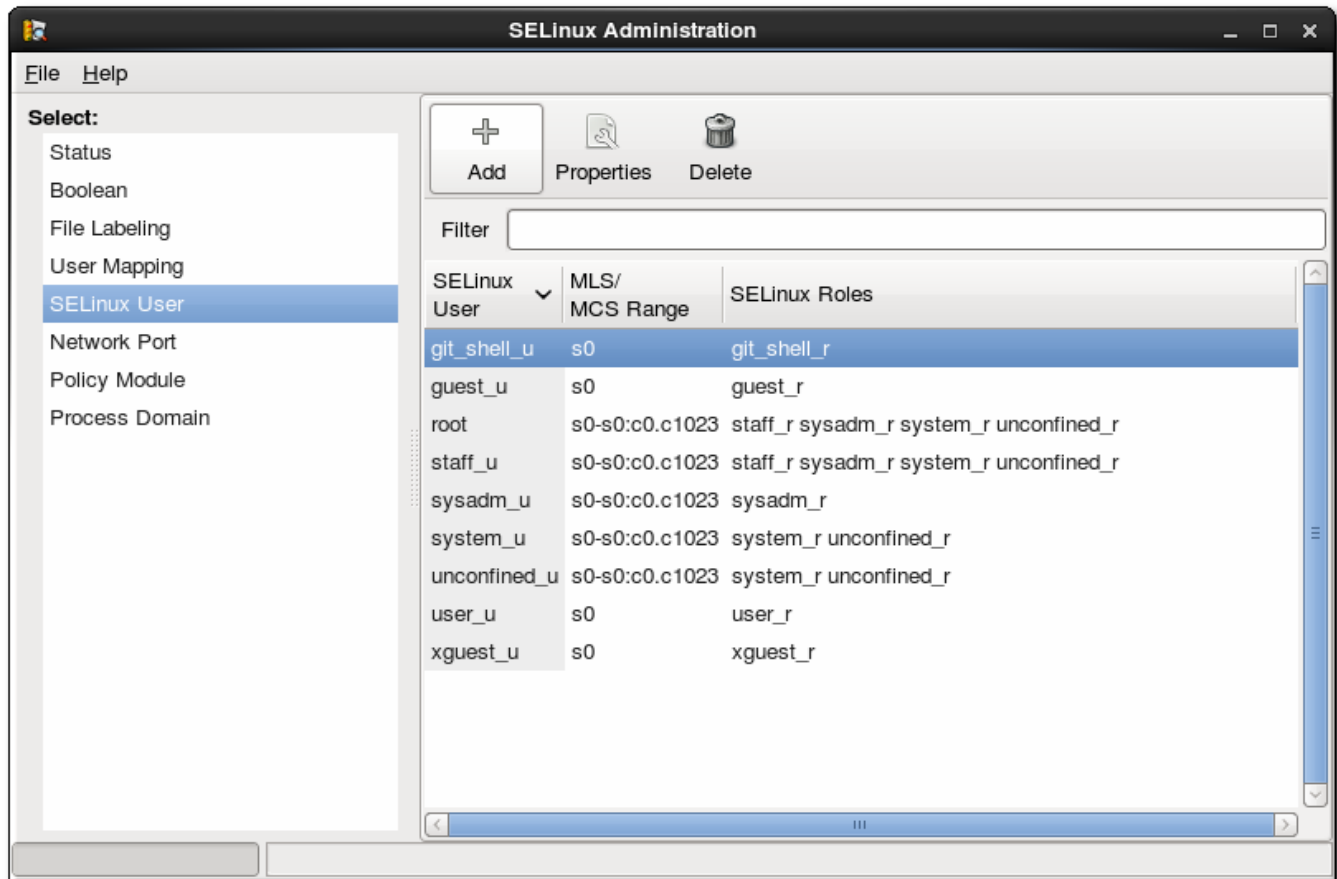


#### Note

Identity Management does not create or modify the SELinux contexts on a system. Rather, it uses existing contexts as the basis to map IdM users (in the domain) to SELinux users (on a system).

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*, 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. Every resource on a system (users, applications, files, processes) is assigned a context.

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 21.1. 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 20, 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.

## 21.2. Configuring SELinux User Map Order and Defaults

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 the underlying SELinux users configuration on the systems it manages.

The available *system* SELinux user maps 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 sets a default SELinux user (one of the users from the total SELinux map list) to use for unmapped IdM user entries. This way, even unmapped IdM users have a functional SELinux context.



### Note

This configuration defines the map order of available system SELinux users. This does not define any IdM user SELinux policies. The IdM user - SELinux user map must be defined and then users are added to the map, as in [Section 21.3, “Mapping SELinux Users and IdM Users”](#).

#### 21.2.1. 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. At the top, there are tabs for 'Identity', 'Policy', and 'IPA Server'. Below these are sub-tabs: 'Role Based Access Control', 'Self Service Permissions', 'Delegations', and 'Configuration'. The 'Configuration' sub-tab is active. The main heading is 'CONFIGURATION'. Below it are links for 'Refresh', 'Reset', and 'Update'. There are four expandable sections: 'SEARCH OPTIONS', 'USER OPTIONS', 'GROUP OPTIONS', and 'SELINUX OPTIONS'. The 'SELINUX OPTIONS' section is expanded, showing two 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.

### 21.2.2. 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: guest_u:s0$guest_u:s0$user_u:s0$staff_u:s0-
s0:c0.c1023$unconfined_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 21.1. List of SELinux Users

The complete list of SELinux users is passed in the **--ipaselinlinuxusermaporder** 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 --
ipaselinlinuxusermaporder="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 21.2. 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 **--ipaselinlinuxusermapdefault**. For example:

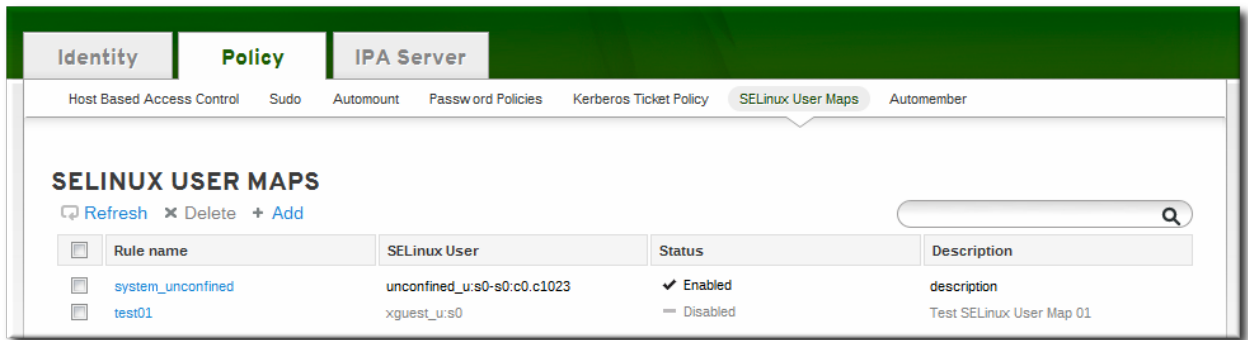
```
[jsmith@server ~]$ ipa config-mod --
ipaselinlinuxusermapdefault="guest_u:s0"
```

## 21.3. Mapping SELinux Users and IdM Users

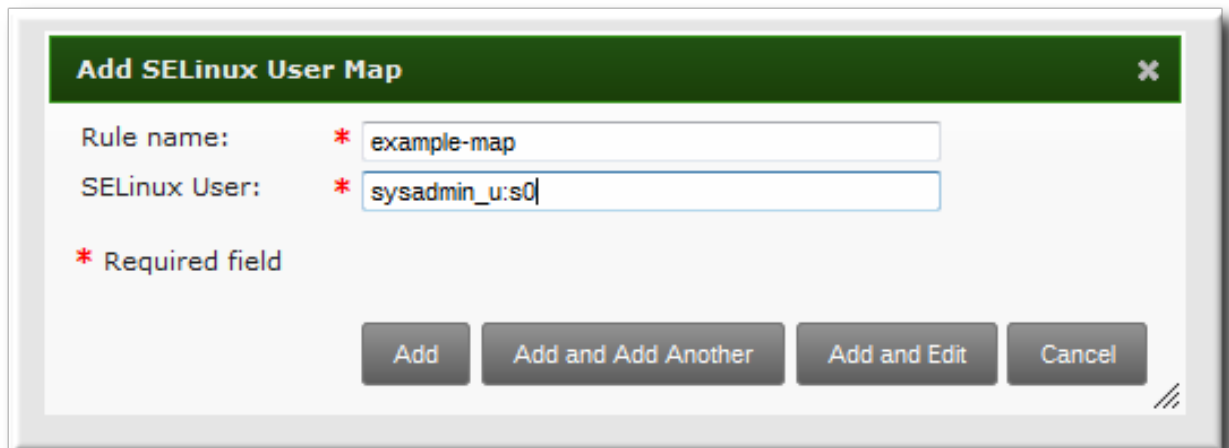
An SELinux map associates an SELinux user context on a local system with an IdM user (or users) within the domain. An SELinux map has three parts: the SELinux user context and an IdM user/host pairing. That IdM user/host pair can be defined in one of two ways: it can be set for explicit users on explicit hosts (or user and host groups), or it can be defined using a host-based access control rule.

### 21.3.1. 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. 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 assigned.**



### Note

The host-based access control rule must contain users and hosts, not just services.

The screenshot shows a web interface for configuring SELinux User Maps. At the top, there is a breadcrumb 'SELinux User Maps » user' and a main heading 'SELINUX USER MAP: user'. Below this is a 'Settings' tab. Under the tab are three buttons: 'Refresh', 'Reset', and 'Update'. A section titled 'GENERAL' is expanded, showing the following fields: 'Rule name' set to 'user', 'Description' with an empty text area, 'SELinux User' with a text input containing 'unconfined\_u:s0-s0:c0.c1023', 'HBAC Rule' with a dropdown menu set to 'web\_admin' and an 'undo' button, and 'Status' with radio buttons for 'Enabled' (selected) and 'Disabled'.

SELinux User Maps » user

## SELINUX USER MAP: user

Settings

[Refresh](#) [Reset](#) [Update](#)

### ▼ GENERAL

Rule name: **user**

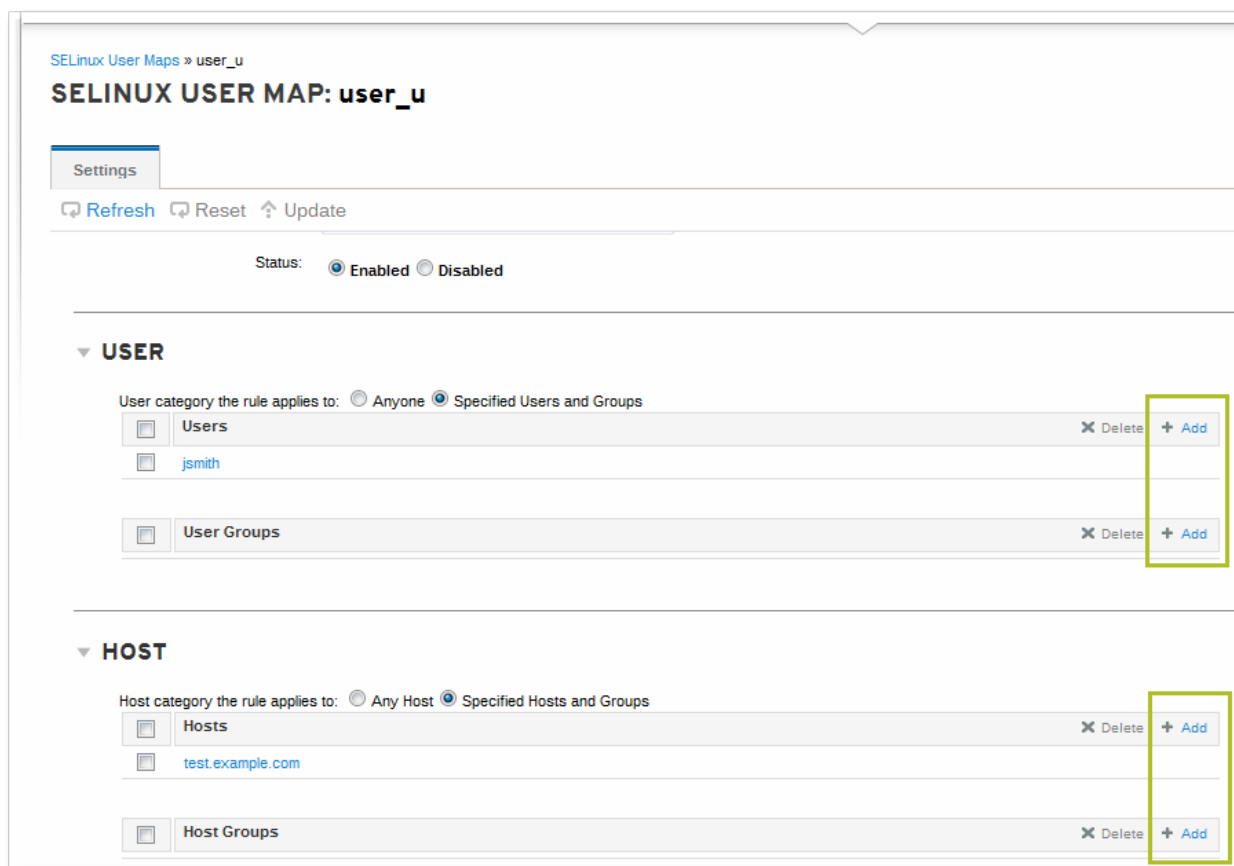
Description:

SELinux User: \*

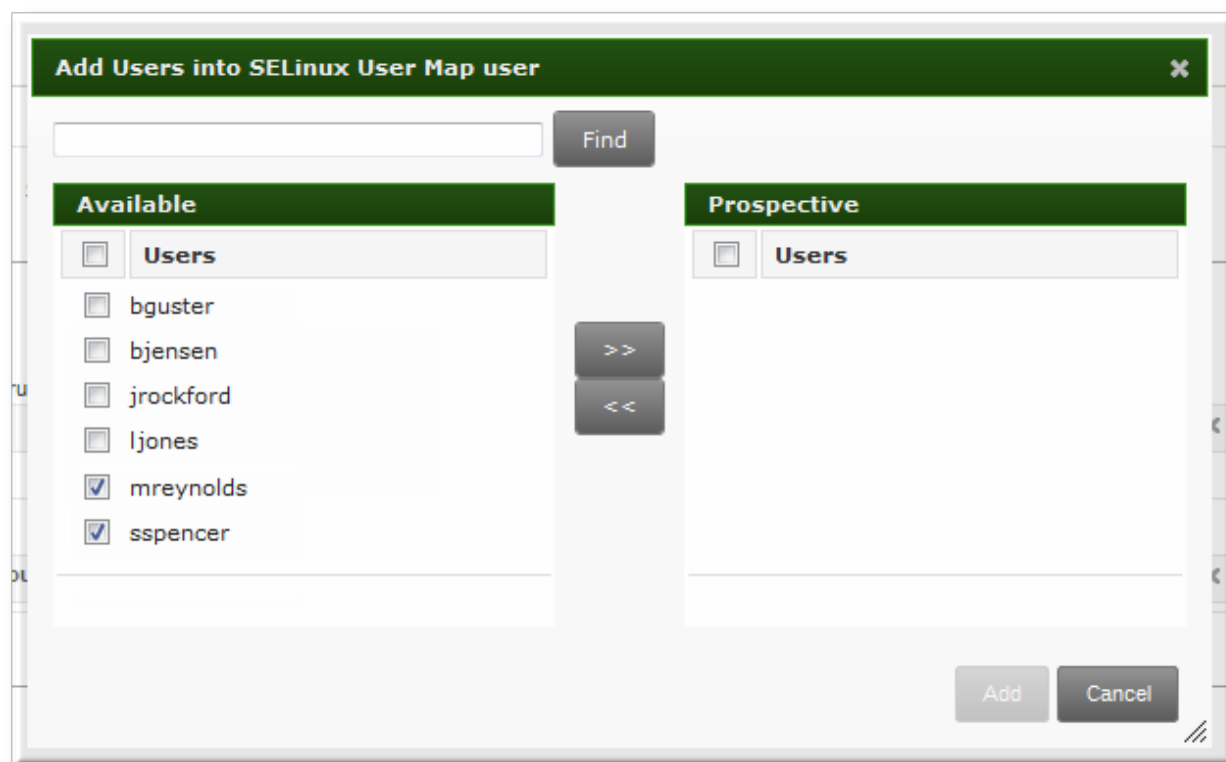
HBAC Rule:  [undo](#)

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.



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.



**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.

**21.3.2. 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 21.3. 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. The user, host, and group options can be used multiple times or can be used once with a comma-separated listed inside curly braces, such as *--option={val1,val2,val3}*.

```
[jsmith@server ~]$ ipa selinuxusermap-add --users=jsmith --
users=bjensen --users=jrockford --hosts=server.example.com --
hosts=test.example.com --selinuxuser="xguest_u:s0" selinux1
```

**Example 21.4. 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 has 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 20, Configuring Host-Based Access Control](#).

### Example 21.5. 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.

### Example 21.6. Removing a User from an SELinux Map

A specific user or host can be removed from an SELinux map by using either the **selinuxusermap-remove-host** or **selinuxusermap-remove-user** command. For example:

```
[jsmith@server ~]$ ipa selinuxusermap-remove-user --users=jsmith  
selinux1
```

## Chapter 22. 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.

### 22.1. 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 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 hosts. Changing the configuration for an existing user or group does not trigger a change group membership.

Automembership is 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 entries by adding them to groups 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.

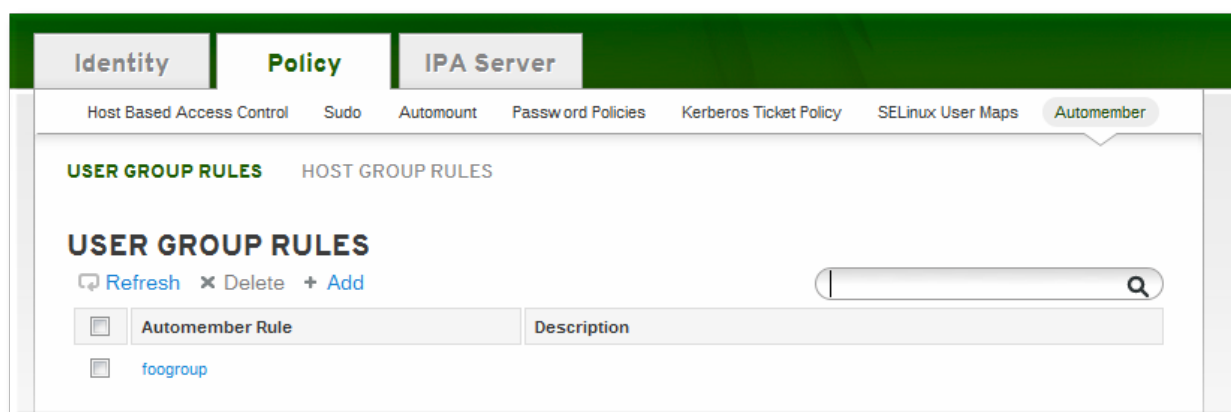
Once the structure is in place, then several things are clear:

- » 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

## 22.2. Defining Automembership Rules (Basic Procedure)

### 22.2.1. From the Web UI

1. Create the user group ([Section 10.10.2.1, “Creating User Groups”](#)) or host group ([Section 11.7.1.1, “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.



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.

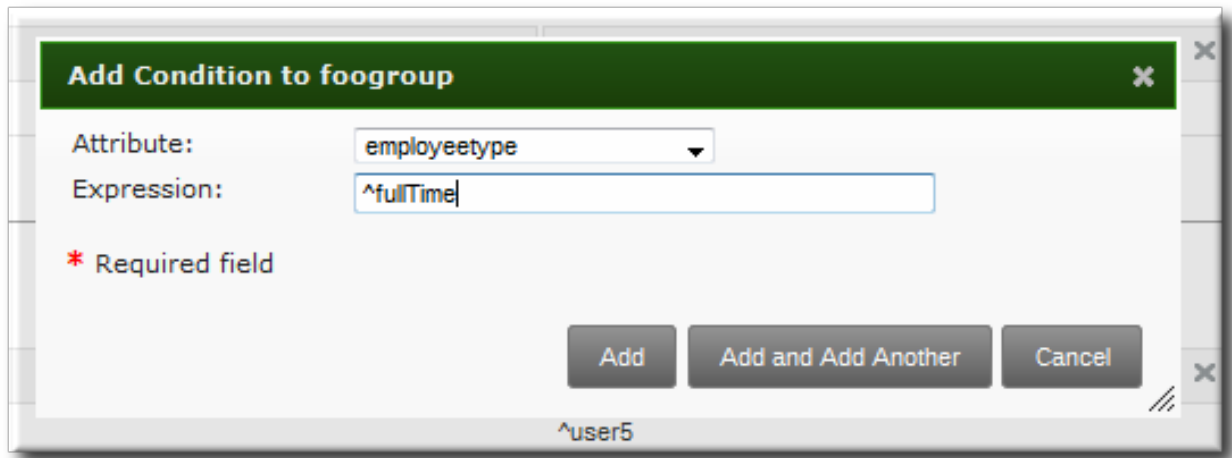
7. Select the attribute to use as the basis for the search and then set the regular expression to use to match the attribute value.

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.

### 22.2.2. From the CLI

There are two commands used to define an automember rule:

- \* A command to target 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 10.10.2.1.2, “With the Command Line”](#)) or host group ([Section 11.7.1.2, “Creating Host Groups from the Command Line”](#)).
2. Create the automember rule 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 inside a set of curly braces with the **--inclusive-regex|--exclusive-regex** options (**--option={pattern1,pattern2}**) 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=employeetype --exclusive-regex=^temp exampleGroup
```



### Note

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 `pcresyntax\(3\)` 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.

## 22.3. 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.

### 22.3.1. 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 `pcresyntax\(3\)` man page](#).

### 22.3.2. 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
```

### 22.3.3. 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 23. Restricting Domains for PAM services

Some environments require that different PAM applications access a different set of SSSD domains. Legacy PAM modules, such as **pam\_ldap** were able to use a separate configuration file as a parameter for a PAM module. This chapter describes a similar feature for SSSD.

One example use case may be an environment that allows external users to authenticate to an FTP server. The server runs as a separate non-privileged user which should only be able to authenticate to a selected SSSD domain, separate from internal company accounts. With this feature, the administrator can allow the FTP user to only authenticate to selected domains specified in the FTP PAM configuration file.

The following options are available for PAM modules and SSSD to restrict access to selected domains in a secure way:

### **pam\_trusted\_users (for sssd.conf)**

This option accepts a list of numerical UIDs or user names that are to be trusted by the SSSD daemon. The default value is the special keyword **all**, which means all users are trusted. This is in line with the current behavior where any user can access any domain.

### **pam\_public\_domains (for sssd.conf)**

This option accepts a comma-separated list of SSSD domains accessible even for untrusted users. Two special keywords, **all** and **none**, are also available. The default value is **none** to make sure that when the administrator starts differentiating between trusted and untrusted domains, he or she is required to manually specify the domains that can be accessed by an untrusted client.

### **domains (for individual PAM module configuration)**

This option accepts a list of domains to which a PAM service will be restricted to authenticate against. The setting interacts with the **domains=** option in the **/etc/sss/sss.conf** file, which specifies the list of domains in the order SSSD will query. The PAM module configuration cannot add to this list but can restrict it by specifying a shorter list.

### **Example 23.1. Sample PAM Module Configuration**

A general configuration line of a **/etc/pam.d/** configuration file has the following form:

```
module-type control-flag module-path arguments
```

In this example, sample configuration for a test module is shown. Arguments to restrict domain access are added at the end of each line. The test module is restricted to only the **openldap** domain and the **pam\_env** module to set/unset environment variables is allowed for all users.

```
$ cat /etc/pam.d/sss_test
auth      required    pam_sss.so domains=openldap
account   required    pam_sss.so domains=openldap
session   required    pam_sss.so domains=openldap
password  required    pam_sss.so domains=openldap
```

In addition to PAM configuration, the relevant snippets `/etc/sss/sss.conf` can look like this:

```
[sss]
domains = ipa, openldap # the list can be restricted by specific PAM
module configuration

[pam]
pam_public_domains = ipa # all users are allowed to access the ipa
domain
pam_trusted_users = root, sss_test # root and sss_test are allowed to
run PAM
```

## **Part V. Configuring the Identity Management Server**



## Chapter 24. Defining Access Control for IdM Users

Access control is a set of security features which defines who can access certain resources, such as machines, services or entries, and what kinds of operations they are allowed to perform. Identity Management provides several access control areas to make it 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. Access Controls for IdM Entries

Access control defines the rights or permissions users have been granted to perform operations on other users or objects.

The Identity Management access control structure is based on standard LDAP access controls. Access within the IdM server is based on the IdM users, stored in the back end Directory Server instance, who are allowed to access other IdM entities, also stored as LDAP entries in the Directory Server instance.

An access control instruction (ACI) has three parts:

#### Actor

This is the entity who is being granted permission to do something. In LDAP access control models, this is called the *bind rule* because it defines who the user is and can optionally require other limits on the bind attempt, such as restricting attempts to a certain time of day or a certain machine.

#### Target

This defines the entry which the actor is allowed to perform operations on.

#### Operation type

*Operation type* — the last part determines 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.

When any operation is attempted, the first thing that the IdM client does is send user credentials as part of the bind operation. The back end Directory Server checks those user credentials and then checks the user account to see if the user has permission to perform the requested operation.

#### 24.1.1. 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

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

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

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

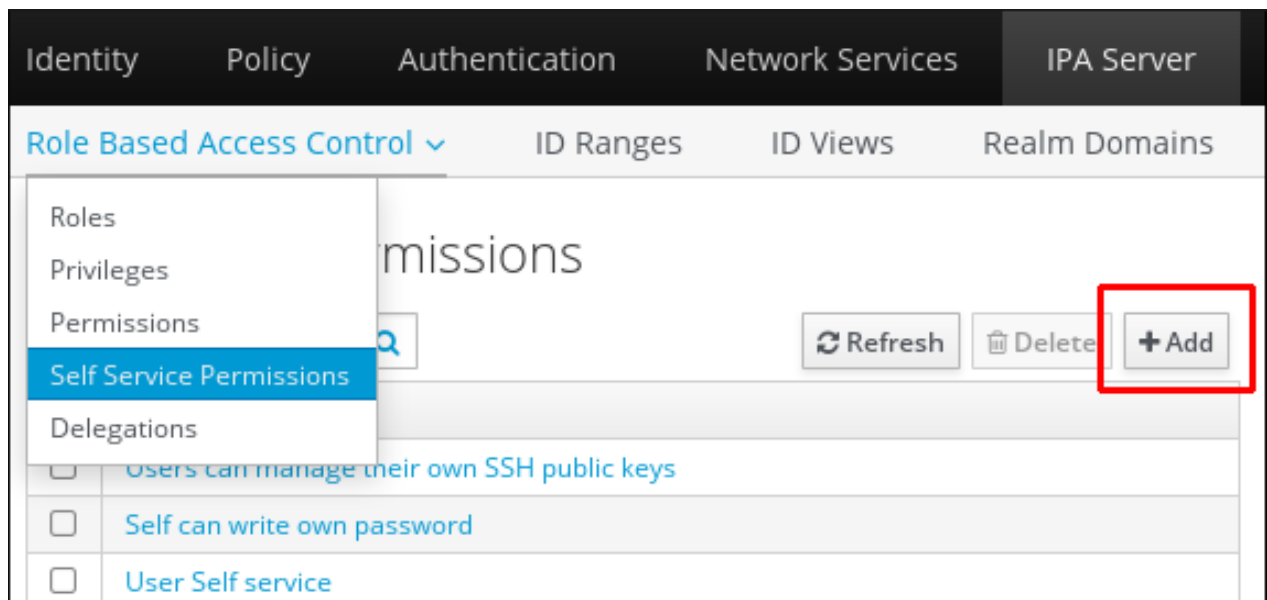
Self-service access control rules define the operations that an entity can perform on itself. These rules define only what attributes a user (or other IdM entity) can edit on their personal entries.

Three 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 personal passwords, including two Samba passwords, the Kerberos password, and the general user password.
- ✦ A rule to manage personal SSH keys.

### 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.
2. Click **Add** at the top of the list of self-service ACIs.



**Figure 24.1. Adding a New Self-Service Rule**

3. Enter the name of the rule in the pop-up window. Spaces are allowed.

 The screenshot shows a dialog box titled 'Add Self Service Permission'. It has a close button (X) in the top right corner. Inside the dialog, there's a 'Self-service name' field with the text 'Adding Personal Info'. Below this is an 'Attributes' section with a search filter and an 'Add' button. A list of attributes is displayed with checkboxes: audio, carlicense, departmentnumber, homedirectory, homepostaladdress, inetuserstatus, internationalisdnumber, ipatokenradiusconfiglink, ipauniqueid, jpegphoto (checked), businesscategory, cn, description, homephone, inetuserhttpurl, ipasshpubkey, ipatokenradiususername, ipauserauthtype, and krbcanonicalname. At the bottom, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'. A note at the bottom left says '\* Required field'.

**Figure 24.2. Form for Adding a Self-Service Rule**

4. Select the checkboxes by the attributes which this ACI will permit users to edit.
5. 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. These two options are required:

- ✱ **--permissions** to set which permissions – such as write, add, or delete – the ACI grants
- ✱ **--attrs** to give the full list of attributes which this ACI grants permission to.

```
[jsmith@server ~]$ ipa selfservice-add "Users can manage their own name
details" --permissions=write --attrs=givenname --attrs=displayname --
attrs=title --attrs=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 ACI. The checkboxes can be selected or deselected.

Self Service Permissions » User Self service

## Self Service Permission: User Self service

**Settings**

Refresh Reset Update

### General

Self-service name User Self service

Attributes \*

Filter  Add

<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory
<input checked="" type="checkbox"/> carlicense	<input checked="" type="checkbox"/> cn
<input type="checkbox"/> departmentnumber	<input checked="" type="checkbox"/> description
<input type="checkbox"/> destinationindicator	<input checked="" type="checkbox"/> displayname
<input type="checkbox"/> employeenumber	<input checked="" type="checkbox"/> employeetype
<input checked="" type="checkbox"/> facsimiletelephonenumber	<input checked="" type="checkbox"/> gecos
<input type="checkbox"/> gidnumber	<input checked="" type="checkbox"/> givenname
<input type="checkbox"/> homedirectory	<input checked="" type="checkbox"/> homephone
<input type="checkbox"/> homepostaladdress	<input checked="" type="checkbox"/> inetuserhttpurl
<input type="checkbox"/> inetuserstatus	<input checked="" type="checkbox"/> initials

**Figure 24.3. 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.

```
[jsmith@server ~]$ ipa selfservice-mod "Users can manage their own name
details" --attrs=givenname --attrs=displayname --attrs=title --
attrs=initials --attrs=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

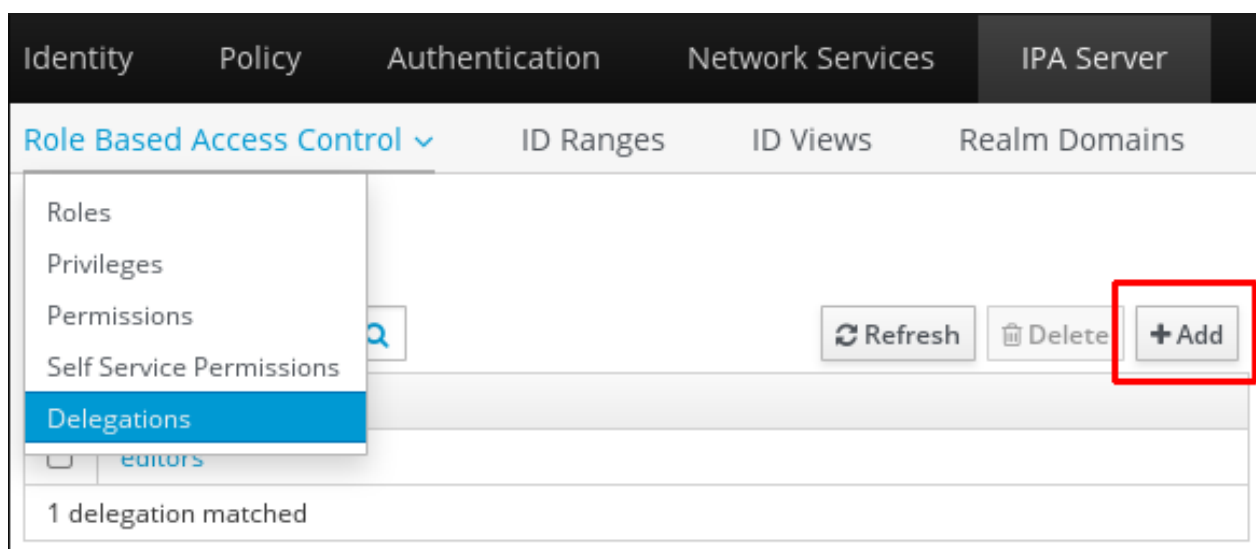
Include all of the attributes when modifying a self-service rule, including existing ones.

## 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.



**Figure 24.4. Adding a New Delegation**

3. Name the new delegation ACI.
4. Set the permissions by selecting the checkboxes whether users will have the right to view the given attributes (read) and add or change the given attributes (write).  
Some users may have a need to see information, but should not be able to edit it.
5. In the **User group** drop-down menu, select the group *who is being granted permissions* to the entries of users in the user group.

**Add Delegation**

Delegation name \*

Permissions ☒ read ☒ write

User group \*

Member user group \*

Attributes \*

☐ audio ☒ businesscategory  
☒ carlicense ☐ cn  
☒ departmentnumber ☐ description  
☐ destinationindicator ☐ displayname  
☐ employeenumber ☒ employeeeetype  
☒ facsimiletelephonenumber ☐ gecoss  
☐ gidnumber ☒ givenname  
☐ homedirectory ☐ homephone  
☐ homepostaladdress ☐ inetuserhttpurl

\* Required field

**Figure 24.5. Form for Adding a Delegation**

6. In the **Member user group** drop-down menu, select the group *whose entries can be edited* by members of the delegation group.
7. In the attributes box, select the checkboxes by the attributes to which the member user group is being granted permission.
8. Click the **Add** button to save the new delegation ACL.

### 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 --attrs=title
--attrs=employeetype --attrs=employeenumber --group=engineering_managers
--membergroup=engineering
-----
Added delegation "basic manager attrs"
-----
Delegation name: basic manager attrs
Permissions: write
Attributes: manager, title, employeetype, 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.

```
[jsmith@server ~]$ ipa delegation-mod "basic manager attrs" --
attrs=manager --attrs=title --attrs=employeetype --attrs=employeenumber
--attrs=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, for example, add, delete, or significantly modify entries.

There are three parts to role-based access controls:

- ✱ The *permission*. The permission defines a specific operation or set of operations (such as read, 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.

With IdM permissions, you can control which users have access to which objects and even which attributes of these objects; IdM enables you to whitelist or blacklist individual attributes or change the entire visibility of a specific IdM function, such as users, groups, or sudo, to all anonymous users, all authenticated users, or just a



certain group of privileged users. This flexible approach to permissions is useful in scenarios when, for example, the administrator wants to limit access of users or groups only to the specific sections these users or groups need to access and to make the other sections completely hidden to them.

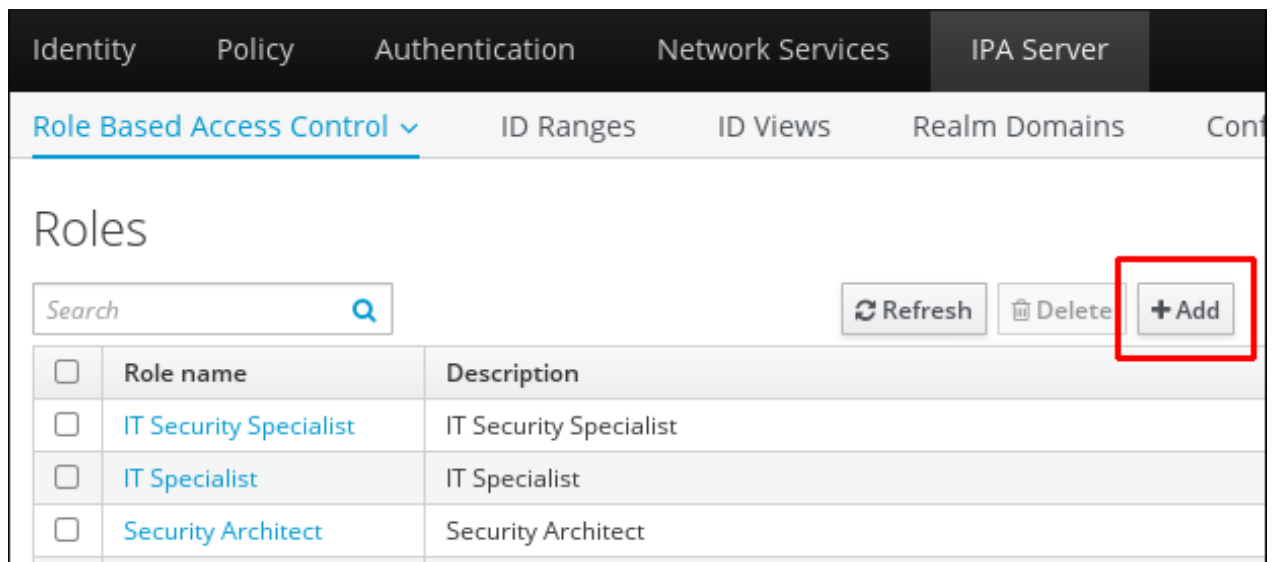
- ✧ The *privileges* available to a role. A privilege is essentially a group of permissions. Permissions are not applied directly to a role. Permissions are added to a privilege so that the privilege creates a coherent and complete picture of a set of access control rules. For example, a permission can 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.

## 24.4.1. Roles

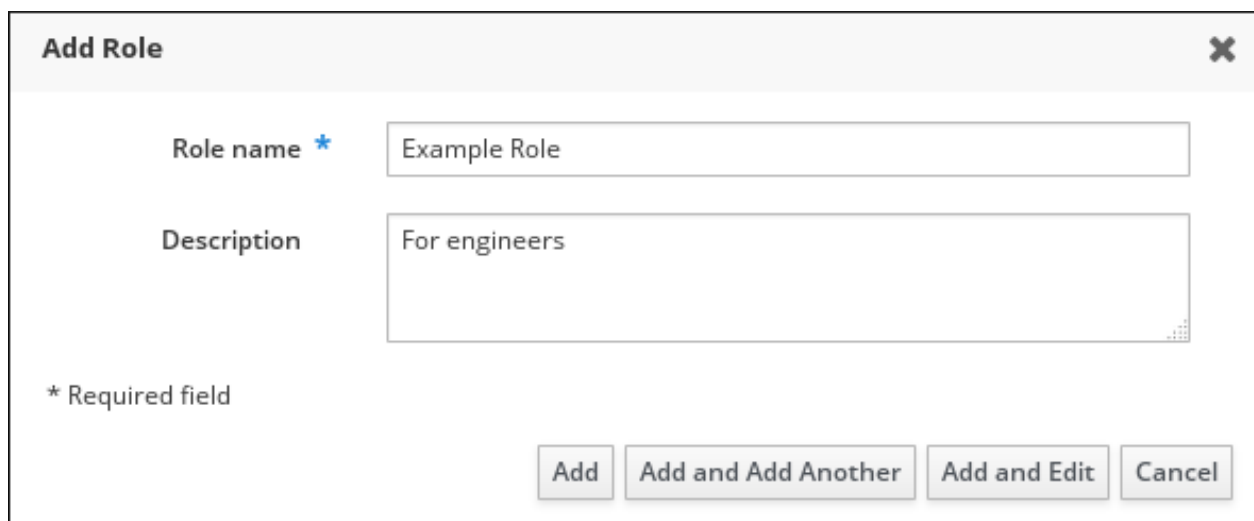
### 24.4.1.1. Creating Roles in the Web UI

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.



**Figure 24.6. Adding a New Role**

3. Enter the role name and a description.



**Add Role** [X]

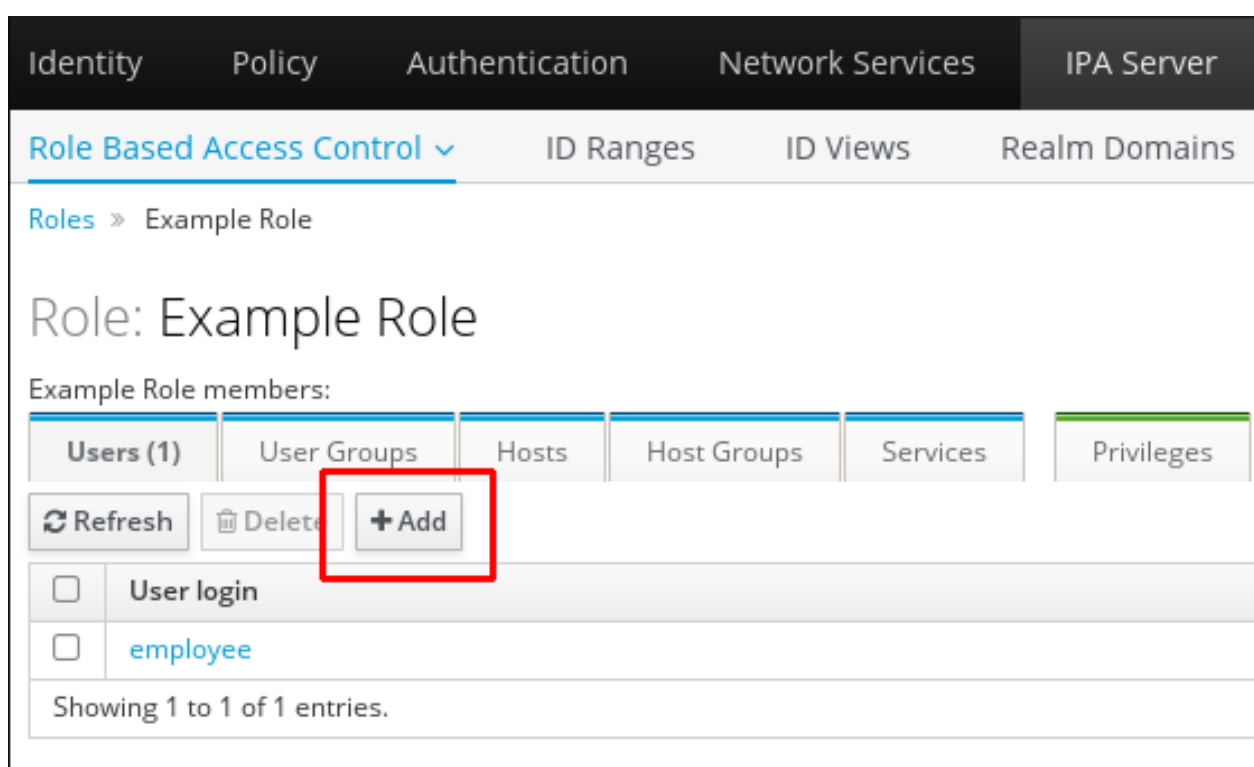
Role name \*

Description

\* Required field

**Figure 24.7. Form for Adding a Role**

- Click the **Add and Edit** button to save the new role and go to the configuration page.
- At the top of the **Users** tab, or in the **Users Groups** tab when adding groups, click **Add**.



Identity Policy Authentication Network Services **IPA Server**

Role Based Access Control ▾ ID Ranges ID Views Realm Domains

Roles » Example Role

## Role: Example Role

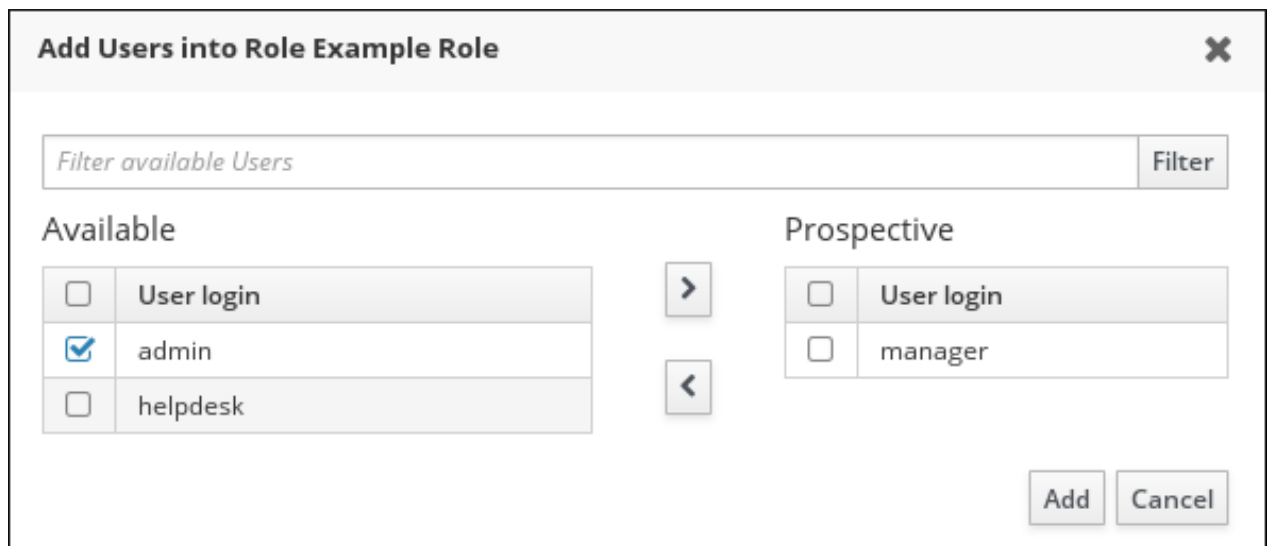
Example Role members:

Users (1)	User Groups	Hosts	Host Groups	Services	Privileges
<input type="button" value="Refresh"/> <input type="button" value="Delete"/> <input type="button" value="+ Add"/>					
<input type="checkbox"/> User login					
<input type="checkbox"/> employee					

Showing 1 to 1 of 1 entries.

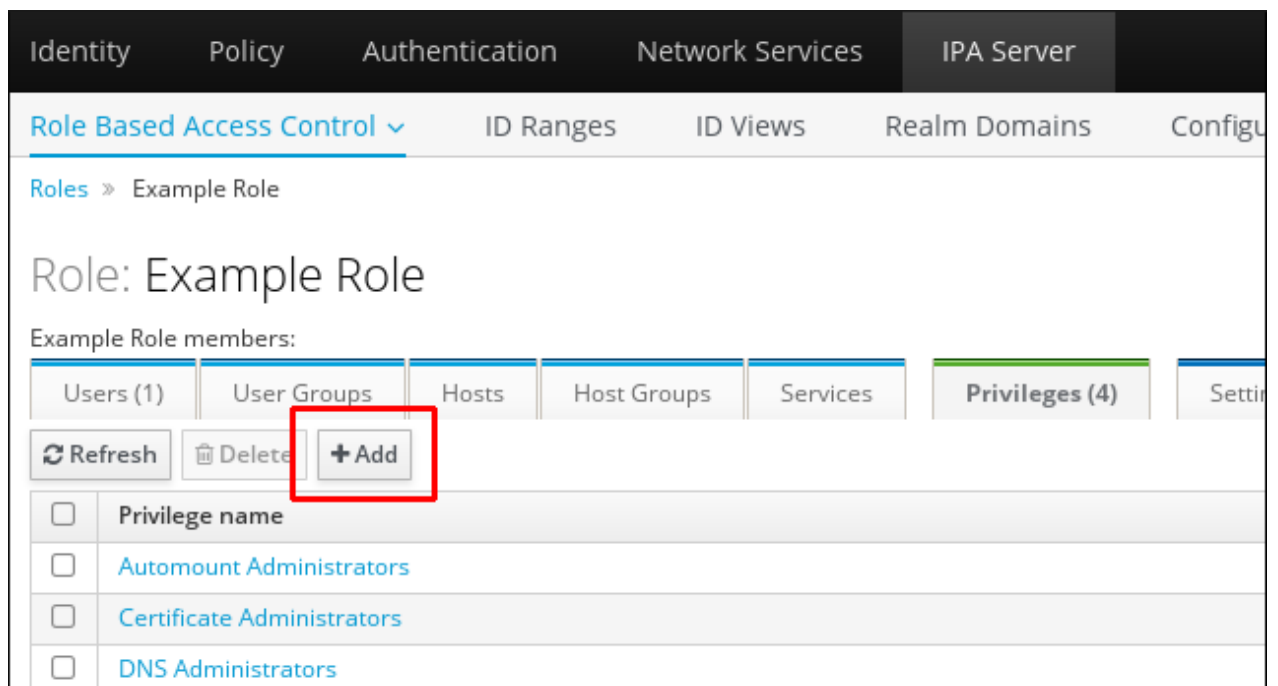
**Figure 24.8. Adding Users**

- Select the users on the left and use the > button to move them to the **Prospective** column.



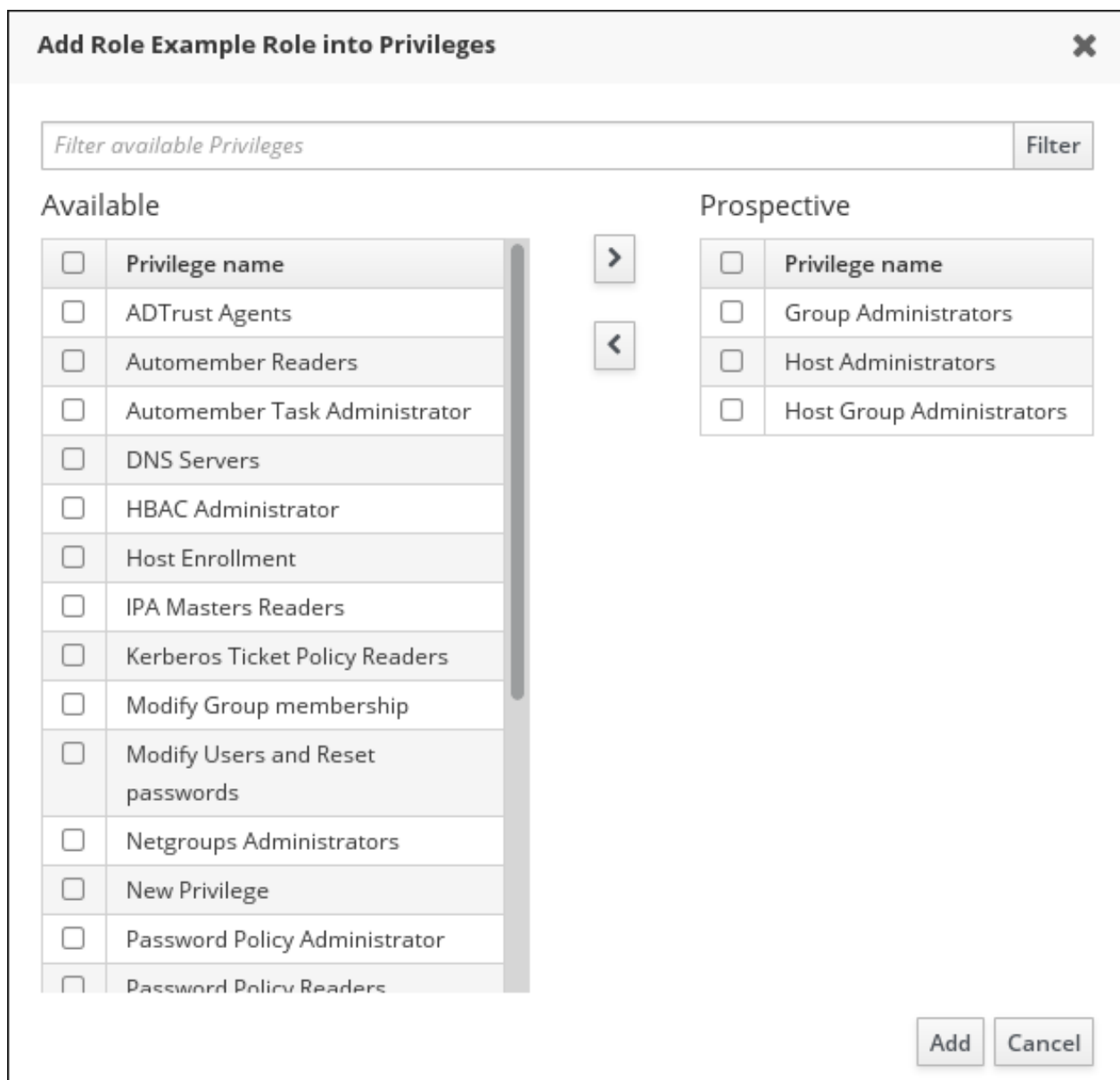
**Figure 24.9. Selecting Users**

- At the top of the **Privileges** tab, click **Add**.



**Figure 24.10. Adding Privileges**

- Select the privileges on the left and use the **>** button to move them to the **Prospective** column.



**Figure 24.11. Selecting Privileges**

9. Click the **Add** button to save.

#### 24.4.1.2. Creating Roles in the Command Line

1. Add the new role:

```
[root@server ~]# kinit admin
[root@server ~]# ipa role-add --desc="User Administrator"
useradmin
-----
Added role "useradmin"
-----
Role name: useradmin
Description: User Administrator
```

2. Add the required privileges to the role:

```
[root@server ~]# ipa role-add-privilege --privileges="User
```

```
Administrators" useradmin
  Role name: useradmin
  Description: User Administrator
  Privileges: user administrators
  -----
  Number of privileges added 1
  -----
```

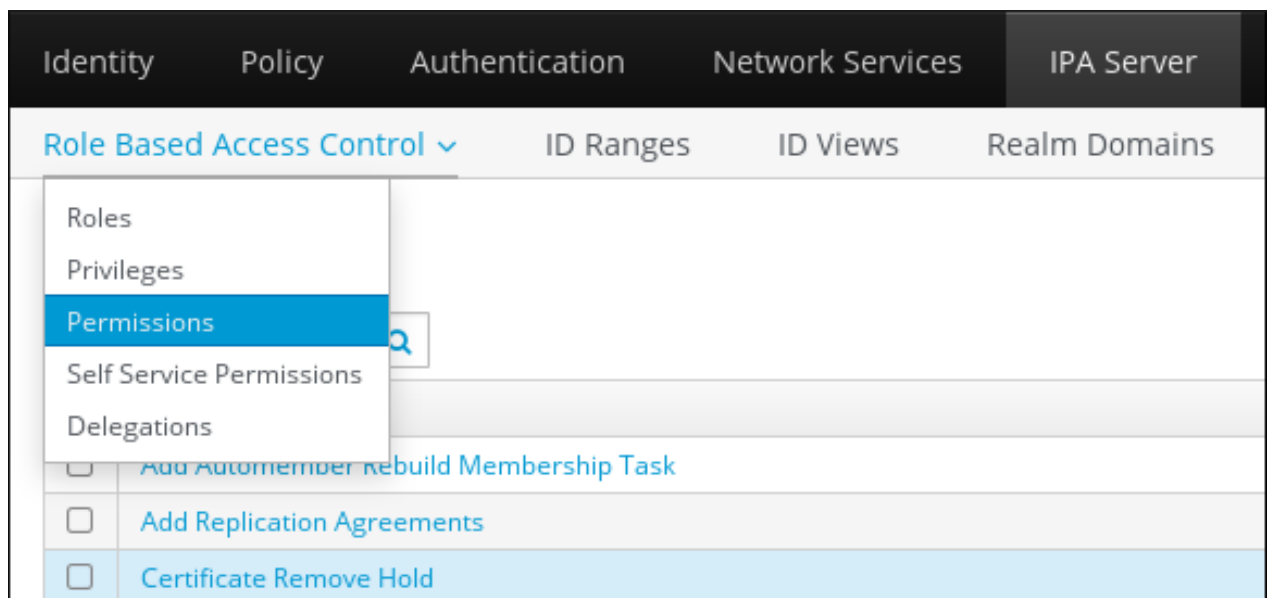
3. Add the required groups to the role. In this case, we are adding only a single group, **useradmin**, which already exists.

```
[root@server ~]# 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. Permissions

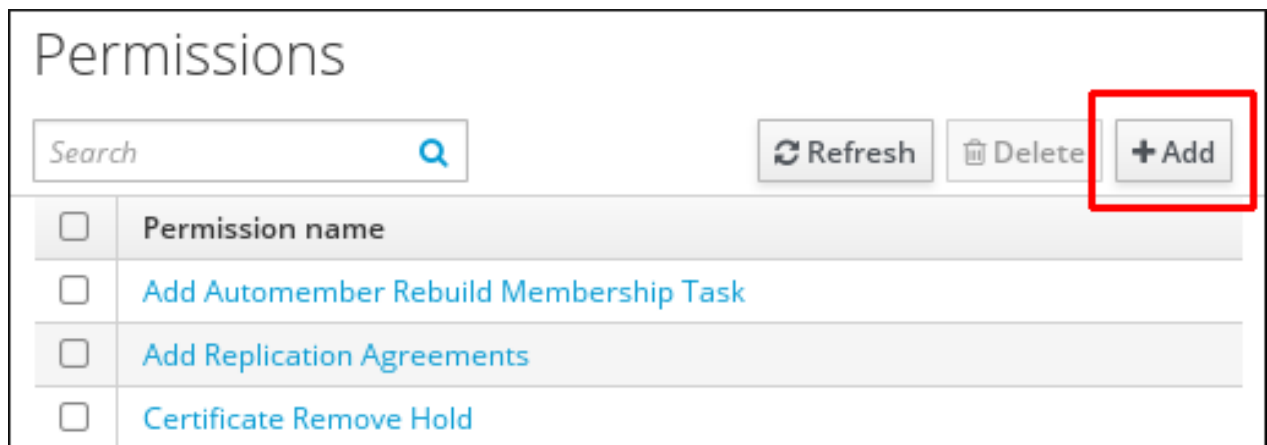
### 24.4.2.1. Creating New Permissions from the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Select the **Permissions** task link.



**Figure 24.12. Permissions Task**

3. Click the **Add** button at the top of the list of permissions.



**Figure 24.13. Adding a New Permission**

4. Define the properties for the new permission in the form that shows up.

Add Permission

Permission name \*

New Permission

Bind rule type

☒ permission
☐ all
☐ anonymous

Granted rights \*

☒ read
☐ search
☐ compare
  
☐ write
☐ add
☐ delete
  
☐ all

Type

Subtree \*

dc=demo1,dc=freeipa,dc=org

Extra target filter

Add

Target DN

Member of group

employees

Undo

Add

Effective attributes

uid

Undo

loginshell

Undo

Add

\* Required field

Add

Add and Add Another

Add and Edit

Cancel

**Figure 24.14. Form for Adding a Permission**

5. Click the **Add** button under the form to save the permission.

You can specify the following permission properties:

1. Enter the name of the new permission.
2. Select the appropriate **Bind rule type**:

- ✧ **permission** is the default permission type, granting access through privileges and roles
- ✧ **all** specifies that the permission applies to all authenticated users
- ✧ **anonymous** specifies that the permission applies to all users, including unauthenticated users



### Note

It is not possible to add permissions with a non-default bind rule type to privileges. You also cannot set a permission that is already present in a privilege to a non-default bind rule type.

3. Choose the rights that the permission grants in **Granted rights**.
4. Define the method to identify the target entries for the permission:
  - ✧ **Type** specifies an entry type, such as user, host, or service. If you choose a value for the **Type** setting, a list of all possible attributes which will be accessible through this ACI for that entry type appears under **Effective Attributes**.

Defining **Type** sets **Subtree** and **Target DN** to one of the predefined values.

- ✧ **Subtree** specifies a subtree entry; every entry beneath this subtree entry is then targeted. Provide an existing subtree entry, as **Subtree** does not accept wildcards or non-existent domain names (DNs). For example:

```
cn=automount,dc=example,dc=com
```

- ✧ **Extra target filter** uses an LDAP filter to identify which entries the permission applies to. The filter can be any valid LDAP filter, for example:

```
(!(objectclass=posixgroup))
```

IdM automatically checks the validity of the given filter. If you enter an invalid filter, IdM warns you about this after you attempt to save the permission.

- ✧ **Target DN** specifies the domain name (DN) and accepts wildcards. For example:

```
uid=*,cn=users,cn=accounts,dc=com
```

- ✧ **Member of group** sets the target filter to members of the given group.

After you fill out the filter settings and click **Add**, IdM validates the filter. If all the permission settings are correct, IdM will perform the search. If some of the permissions settings are incorrect, IdM will display a message informing you about which setting is set incorrectly.

5. If you set **Type**, choose the **Effective attributes** from the list of available ACI attributes. If you did not use **Type**, add the attributes manually by writing them into the **Effective attributes** field. Add a single attribute at a time; to add multiple attributes, click **Add** to add another input field.



**Important**

If you do not set any attributes for the permission, then all attributes are included by default.

**24.4.2.2. Creating New Permissions from the Command Line**

To add a new permission, issue the **ipa permission-add** command. Specify the properties of the permission by supplying the corresponding options:

- ✱ Supply the name of the permission. For example:

```
[root@server ~]# ipa permission-add "dns admin permission"
```

- ✱ **--bindtype** specifies the bind rule type. This options accepts the **all**, **anonymous**, and **permission** arguments. For example:

```
--bindtype=all
```

If you do not use **--bindtype**, the type is automatically set to the default **permission** value.

**Note**

It is not possible to add permissions with a non-default bind rule type to privileges. You also cannot set a permission that is already present in a privilege to a non-default bind rule type.

- ✱ **--permissions** lists the rights granted by the permission. You can set multiple attributes by using multiple **--permissions** options or by listing the options in a comma-separated list inside curly braces. For example:

```
--permissions=read --permissions=write
--permissions={read,write}
```

- ✱ **--attrs** gives the list of attributes over which the permission is granted. You can set multiple attributes by using multiple **--attrs** options or by listing the options in a comma-separated list inside curly braces. For example:

```
--attrs=description --attrs=automountKey
--attrs={description,automountKey}
```

The attributes provided with **--attrs** must exist and be allowed attributes for the given object type, otherwise the command fails with schema syntax errors.

- ✱ **--type** defines the entry object type, such as user, host, or service. Each type has its own set of allowed attributes. For example:

```
[root@server ~]# ipa permission-add "manage service" --permissions=all
--type=service --attrs=krbprincipalkey --attrs=krbprincipalname --
attrs=managedby
```

- ✧ **--subtree** gives a subtree entry; the filter then targets every entry beneath this subtree entry. Provide an existing subtree entry; **--subtree** does not accept wildcards or non-existent domain names (DNs). Include a DN within the directory.

Because IdM uses a simplified, flat directory tree structure, **--subtree** can be used to target some types of entries, like automount locations, which are containers or parent entries for other configuration. For example:

```
[root@server ~]# ipa permission-add "manage automount locations" --
subtree="ldap://ldap.example.com:389/cn=automount,dc=example,dc=com" -
-permissions=write --attrs=automountmapname --attrs=automountkey --
attrs=automountInformation
```

The **--type** and **--subtree** options are mutually exclusive.

- ✧ **--filter** uses an LDAP filter to identify which entries the permission applies to. IdM automatically checks the validity of the given filter. The filter can be any valid LDAP filter, for example:

```
[root@server ~]# ipa permission-add "manage Windows groups" --filter="
(!(objectclass=posixgroup))" --permissions=write --attrs=description
```

- ✧ **--memberof** sets the target filter to members of the given group after checking that the group exists. For example:

```
[root@server ~]# ipa permission-add ManageHost --permissions="write" -
-subtree=cn=computers,cn=accounts,dc=testrelm,dc=com --
attr=nshostlocation --memberof=admins
```

- ✧ **--targetgroup** sets target to the specified user group after checking that the group exists.

The **Target DN** setting, available in the web UI, is not available on the command line.



## Note

For information about modifying and deleting permissions, run the **ipa permission-mod --help** and **ipa permission-del --help** commands.

### 24.4.2.3. Default Managed Permissions

*Managed* permissions are permissions that come pre-installed with Identity Management. They behave like regular user-created permissions, with the following differences:

- ✧ You cannot modify their name, location, and target attributes.
- ✧ You cannot delete them.
- ✧ They have three sets of attributes:

- *default* attributes, which are managed by IdM and the user cannot modify them
- *included* attributes, which are additional attributes added by the user; to add an included attribute to a managed permission, specify the attribute by supplying the **--includedattrs** option with the **ipa permission-mod** command
- *excluded* attributes, which are attributes removed by the user; to add an excluded attribute to a managed permission, specify the attribute by supplying the **--excludedattrs** option with the **ipa permission-mod** command

A managed permission applies to all attributes that appear in the default and included attribute sets but not in the excluded set.

If you use the **--attrs** option when modifying a managed permission, the included and excluded attribute sets automatically adjust, so that only the attributes supplied with **--attrs** are enabled.



### Note

While you cannot delete a managed permission, setting its bind type to **permission** and removing the managed permission from all privileges effectively disables it.

Names of all managed permissions start with **System:**, for example *System: Add Sudo rule* or *System: Modify Services*.

Earlier versions of IdM used a different scheme for default permissions, which, for example, forbade the user from modifying the default permissions and the user could only assign them to privileges. Most of these default permissions have been turned into managed permissions, however, the following permissions still use the previous scheme:

- Add Automember Rebuild Membership Task
- Add Replication Agreements
- Certificate Remove Hold
- Get Certificates status from the CA
- Modify DNA Range
- Modify Replication Agreements
- Remove Replication Agreements
- Request Certificate
- Request Certificates from a different host
- Retrieve Certificates from the CA
- Revoke Certificate
- Write IPA Configuration

If you attempt to modify a managed permission from the web UI, the attributes that you cannot modify will be grayed-out.

Permission: System: Modify Users

Settings Privileges (2)

Refresh Reset Update

Permission settings

Permission name

System: Modify Users

Bind rule type

☒ permission ☐ all ☐ anonymous

Granted rights

☐ read ☐ search ☐ compare ☒ write

☐ add ☐ delete ☐ all

**Figure 24.15. Grayed-Out Attributes**

If you attempt to modify a managed permission from the command line, the system will not allow you to change the attributes that you cannot modify. For example, attempting to change a default **System: Modify Users** permission to apply to groups fails:

```
$ ipa permission-mod 'System: Modify Users' --type=group
ipa: ERROR: invalid 'ipapermlocation': not modifiable on managed
permissions
```

You can, however, make the **System: Modify Users** permission not to apply to the **GECOS** attribute:

```
$ ipa permission-mod 'System: Modify Users' --excludedattrs=gecos
-----
Modified permission "System: Modify Users"
```

#### 24.4.2.4. Permissions in Earlier Versions of Identity Management

Earlier versions of Identity Management handled permissions differently, for example:

- ✧ Only write, add, and delete permission types were available.
- ✧ The permission-setting options were not as fine-grained, as it was not possible to, for example, add both a filter and a subtree in the same permission.

- ✱ The global IdM ACI granted read access to all users of the server, even anonymous – that is, not logged-in – users.

The new way of handling permissions has significantly improved the IdM capabilities for controlling user or group access, while retaining backward compatibility with the earlier versions. Upgrading from an earlier version of IdM deletes the global IdM ACI on all servers and replaces it with *managed permissions*.

Permissions created in the previous way are automatically converted to the new style whenever you modify them. If you do not attempt to change them, the previous-style permissions stay unconverted. Once a permission uses the new style, it can never downgrade to the previous style.



## Note

It is still possible to assign permissions to privileges on servers running an earlier version of IdM.

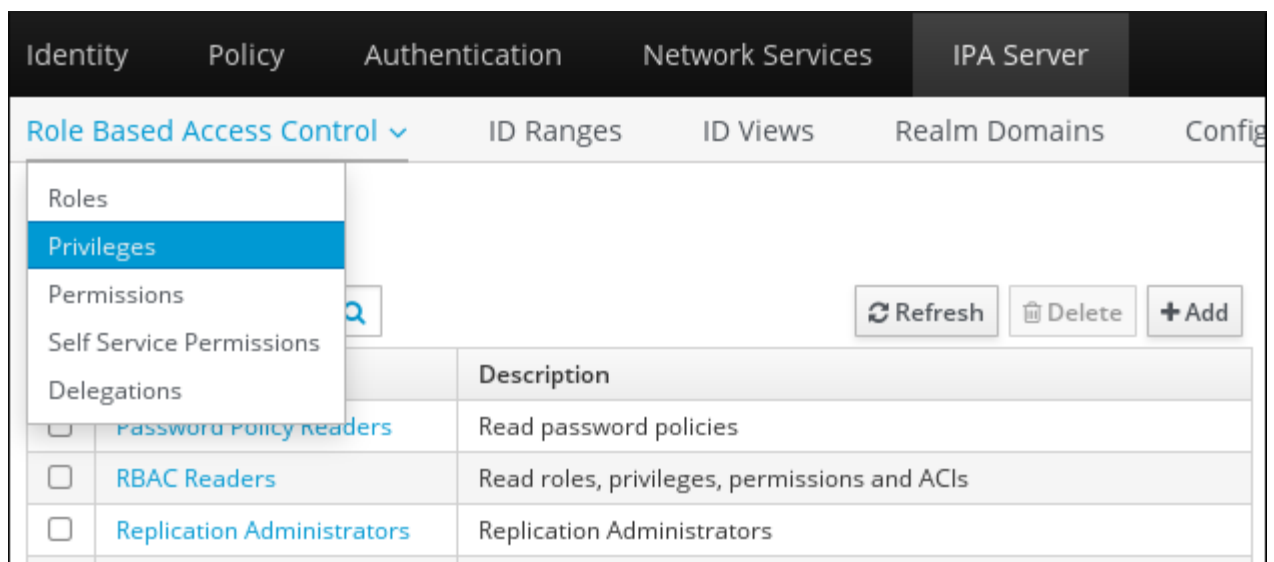
The **ipa permission-show** and **ipa permission-find** commands recognize both the new-style permissions and the previous-style permissions. While the outputs from both of these commands display permissions in the new style, they do not change the permissions themselves; they upgrade the permission entries before outputting the data only in memory, without committing the changes to LDAP.

Both the previous-style and the new-style permissions have effect on all servers – those running previous versions of IdM, as well as those running the current IdM version. However, you cannot create or modify the new-style permissions on servers running previous versions of IdM.

### 24.4.3. Privileges

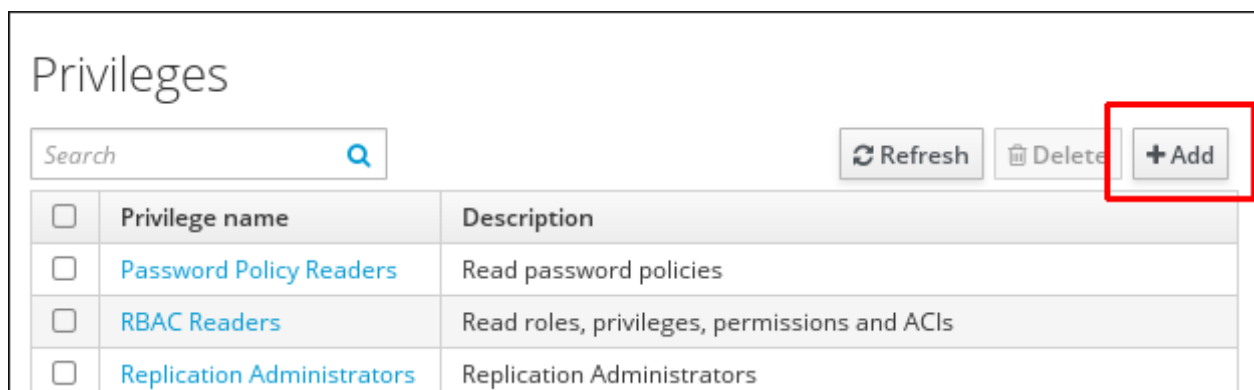
#### 24.4.3.1. Creating New Privileges from the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Select the **Privileges** task link.



**Figure 24.16. Privileges Task**

- Click the **Add** link at the top of the list of privileges.

**Figure 24.17. Adding a New Privilege**

- Enter the name and a description of the privilege.

**Add Privilege** [X]

Privilege name \*

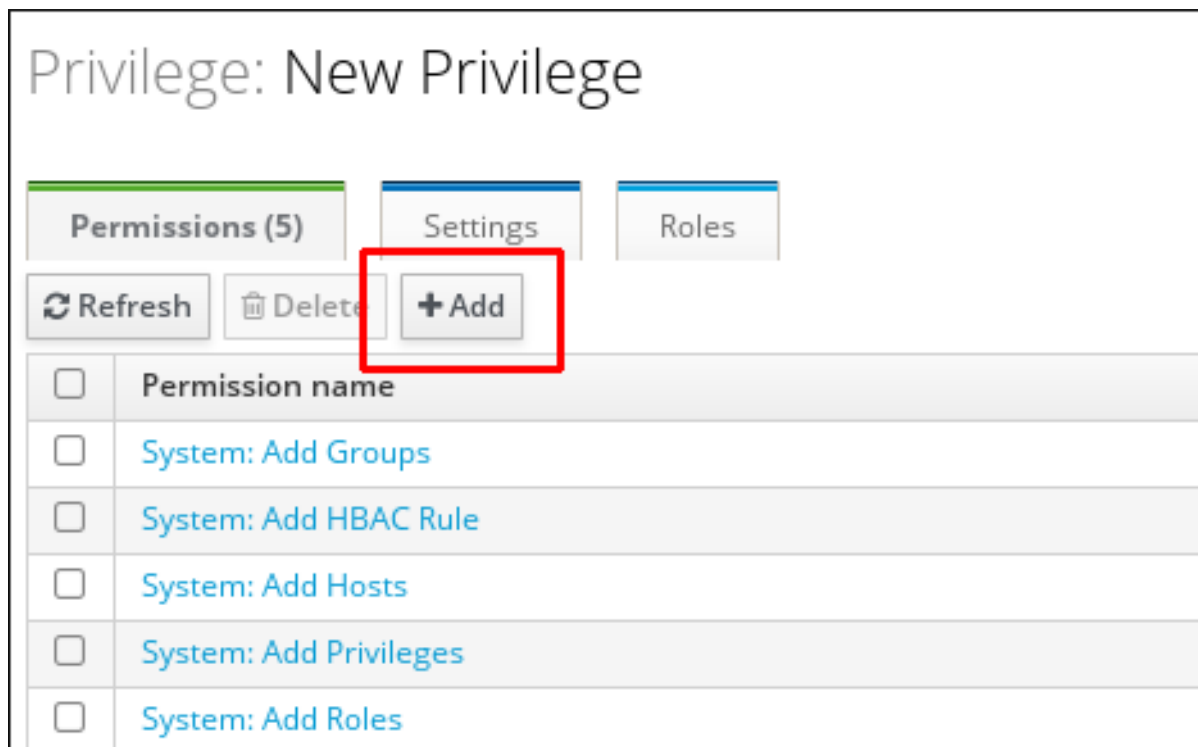
Description

\* Required field

[Add] [Add and Add Another] [Add and Edit] [Cancel]

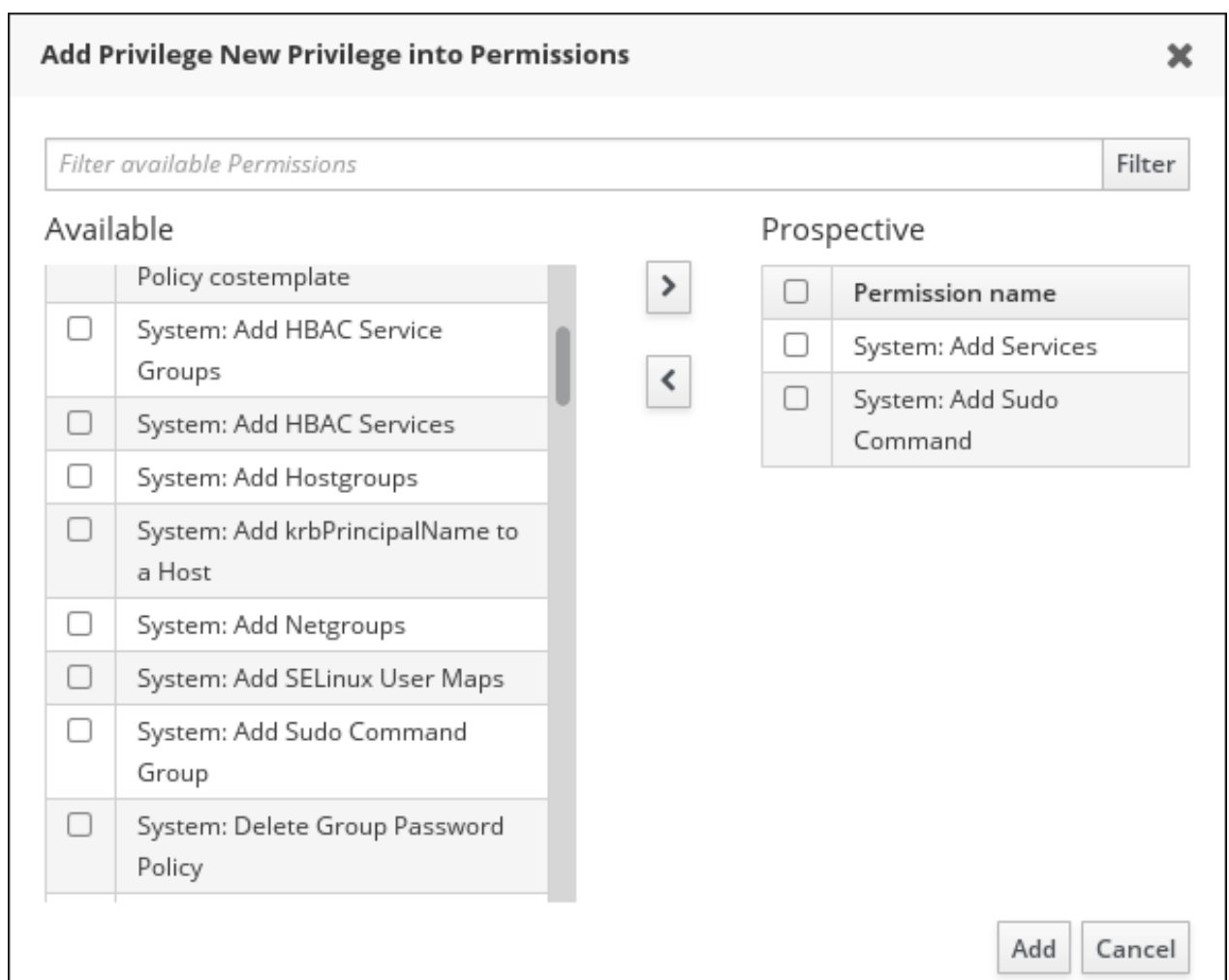
**Figure 24.18. Form for Adding a Privilege**

- Click the **Add and Edit** button to go to the privilege configuration page to add permissions.
- Select the **Permissions** tab.
- Click **Add** at the top of the list of permissions to add permission to the privilege.



**Figure 24.19. Adding Permissions**

- Click the checkbox by the names of the permissions to add, and use the > button to move the permissions to the **Prospective** column.



**Figure 24.20. Selecting Permissions**

9. Click the **Add** button to save.

**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.

1. Create the privilege entry.

```
[jsmith@server ~]$ ipa privilege-add "managing filesystems" --  
desc="for filesystems"
```

2. Assign the desired permissions. For example:

```
[jsmith@server ~]$ ipa privilege-add-permission "managing  
filesystems" --permissions="managing automount" --  
permissions="managing ftp services"
```



## Chapter 25. 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 chapter 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. A Reference of IdM Server Configuration Files and Directories

**Table 25.1. IdM Server Configuration Files and Directories**

Directory or File	Description
<b>Server Configuration</b>	
/etc/ipa/	The main IdM configuration directory.
/etc/ipa/default.conf	The primary configuration file for IdM.
/etc/ipa/server.conf	An optional configuration file for IdM. This does not exist by default, but can be created to load custom configuration when the IdM server is started.
/etc/ipa/cli.conf	An optional configuration file for IdM command-line tools. This does not exist by default, but can be created to apply custom configuration when the <b>ipa</b> is used.
/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.
<b>IdM Logs</b>	
~/ipa/log/cli.log	The log file for errors returned by 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.
/etc/logrotate.d/	The log rotation policies for DNS, SSSD, Apache, Tomcat, and Kerberos.
<b>System Services</b>	
/etc/rc.d/init.d/ipa/	The IdM server init script.
<b>Web UI</b>	
/etc/ipa/html/	A symlink directory in the main configuration directory for the HTML files used by the IdM web UI.

Directory or File	Description
/etc/httpd/conf.d/ipa.conf /etc/httpd/conf.d/ipa-rewrite.conf	The configuration files used by the Apache host for the web UI application.
/etc/httpd/conf/ipa.keytab /usr/share/ipa/	The keytab file used by the web UI service. The main directory for all of the HTML files, scripts, and stylesheets used by the web UI.
/usr/share/ipa/ipa-rewrite.conf /usr/share/ipa/ipa.conf	The configuration files used by the Apache host for the web UI application.
/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.
<b>Kerberos</b>	
/etc/krb5.conf	The Kerberos service configuration file.
<b>SSSD</b>	
/usr/share/sss/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.
<b>389 Directory Server</b>	
/var/lib/dirsrv/slapd- <i>REALM_NAME</i> /	All of the database associated with the Directory Server instance used by the IdM server.
/etc/dirsrv/slapd- <i>REALM_NAME</i> /	All of the configuration and schema 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.
<b>Dogtag Certificate System</b>	
/etc/pki-ca/ /var/lib/pki/pki-tomcat/conf/ca/CS.cfg	The main directory for the IdM CA instance. The main configuration file for the IdM CA instance.
/var/log/dirsrv/slapd- <i>REALM</i> /	Log files associated with the Directory Server instance used by the IdM CA.
<b>Cache Files</b>	

Directory or File	Description
/var/cache/ipa/	Cache files for the IdM server and the IdM Kerberos password daemon.
<b>System Backups</b>	
/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 <b>.conf</b> files for NSS, Kerberos (both <b>krb5.conf</b> and <b>kdc.conf</b> ), 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 <b>sssd.conf</b> file for SSSD authentication services.

## 25.2. IdM Domain Services and Log Rotation

The 389 Directory Server instances used by IdM as a backend and by the Dogtag Certificate System have their own internal log rotation policies. Log rotation settings such as the size of the file, the period between log rotation, and how long log files are preserved can all be configured by editing the 389 Directory Server configuration. This is covered in the [Red Hat Directory Server Administrator's Guide](#).

Several IdM domain services use the system **logrotate** service to handle log rotation and compression:

- ✧ named (DNS)
- ✧ httpd (Apache)
- ✧ tomcat6
- ✧ sssd
- ✧ krb5kdc (Kerberos domain controller)

Most of these policies use the **logrotate** defaults for the rotation schedule (weekly) and the archive of logs (four, for four weeks' worth of logs).

The individual policies set post-rotation commands to restart the service after log rotation, that a missing log file is acceptable, and compression settings.

### Example 25.1. Default httpd Log Rotation File

```
[root@server ~]# cat /etc/logrotate.d/httpd
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    delaycompress
    postrotate
        /sbin/service httpd reload > /dev/null 2>/dev/null || true
    endscript
}
```

There are other potential log settings, like compress settings and the size of the log file, which can be edited in either the global **logrotate** configuration or in the individual policies. The **logrotate** settings are covered in the [logrotate](#) manual page.



## Warning

Two policies set special **create** rules. All of the services create a new log file with the same name, default owner, and default permissions as the previous log. For the **named** and **tomcat6** logs, the **create** is set with explicit permissions and user/group ownership.

```
[root@server ~]# cat /etc/logrotate.d/named
/var/named/data/named.run {
    missingok
    create 0644 named named
    postrotate
        /sbin/service named reload 2> /dev/null > /dev/null ||
true
    endscript
}
```

**Do not change the permissions or the user and group which own the log files.** This is required for both IdM operations and SELinux settings. Changing the ownership of the log rotation policy or of the files can cause the IdM domains services to fail or to be unable to start.

## 25.3. 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.

## 25.4. 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	<code>/var/log/ipaserver-install.log</code>	Server installation log	
IdM server	<code>~/ipa/log/cli.log</code>	Command-line tool log	
IdM client	<code>/var/log/ipaclient-install.log</code>	Client installation log	
Apache server	<code>/var/log/httpd/access_log</code> <code>/var/log/httpd/error_log</code>	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.	<a href="#">Apache log chapter</a>
Dogtag Certificate System	<code>/var/log/pki-ca-install.log</code>	The installation log for the IdM CA.	
Dogtag Certificate System	<code>/var/log/pki-ca/debug</code> <code>/var/log/pki-ca/system</code> <code>/var/log/pki-ca/transactions</code> <code>/var/log/pki-ca/signedAudit</code>	These logs mainly relate to certificate operations. In IdM, this is used for service principals, hosts, and other entities which use certificates.	<a href="#">Logging chapter</a>

Service	Log File	Description	Additional Information
389 Directory Server	/var/log/dirsrv/slapd- <i>REALM</i> /access  /var/log/dirsrv/slapd- <i>REALM</i> /audit  /var/log/dirsrv/slapd- <i>REALM</i> /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.  » <a href="#">Monitoring servers and databases</a> » <a href="#">Log entries explained</a>
389 Directory Server	/var/log/dirsrv/slapd- <i>REALM</i> /access  /var/log/dirsrv/slapd- <i>REALM</i> /audit  /var/log/dirsrv/slapd- <i>REALM</i> /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.  » <a href="#">Monitoring servers and databases</a> » <a href="#">Log entries explained</a>
Kerberos	/var/log/krb5libs.log	This is the primary log file for Kerberos connections.	This location is configured in the <b>krb5.conf</b> file, so it could be different on some systems.
Kerberos	/var/log/krb5kdc.log	This is the primary log file for the Kerberos KDC server.	This location is configured in the <b>krb5.conf</b> file, so it could be different on some systems.
Kerberos	/var/log/kadmind.log	This is the primary log file for the Kerberos administration server.	This location is configured in the <b>krb5.conf</b> file, so it could be different on some systems.

Service	Log File	Description	Additional Information
DNS	/var/log/messages	DNS error messages are included with other system messages.	<p>DNS logging is <i>not</i> enabled by default. DNS logging is enabled by running the <b>querylog</b> command:</p> <pre>/usr/sbin/rndc querylog</pre> <p>This begins writing log messages to the system's <b>/var/log/messages</b> file. To turn off logging, run the <b>querylog</b> command again.</p>

### 25.4.1. Enabling Server Debug Logging

Debug logging for the IdM server is set in the **server.conf** file.



#### Note

Editing the **default.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
```

### 25.4.2. 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 [jrussell]:
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
YIIFgQYJKoZIHvcSAQICAQBuggVwMIIFbKADAgEFoQMCAQ6iBwMFACAAAACjggGHYYIBgzCC
AX+gAwIBBaEZGxdSSFRTLkVORy5CT1MuUKVESEFULkNPTaI5MDegAwIBA6EwMC4bBEhUVFAB
JmRlbGwtcGUxODUwLTaxLnJodHMuZW5nLmJvcy5yZWRoYXQuY29to4IBIDCCARygAwIBEqED
AgECooIBDgSCAQpV2YEWv03X+SEndUBf0hMFGc3Fvnd51nELV0rIB1tfGVjpNlkuQxXKSfFK
VD3vyAUqkii255T0mnXyTwayE93W1U4s0shetmG50zeU4KDmhuupzQZSCb5xB0KPU4HMDvP1
UnDFJUGCK9tcqDJiYE+lJrEcZ8H+Vxvvl+nP6yIdUQKqoEuNhJaLWIi8ieAzk8zvmDlDzpF
YtInGWe9D5ko1Bb7Npu0SEpdVJB2gnB5vszCIdLlzM4JUqX8p21AZV0UYA6QZ0WX90XhqHd
ElKcuHCN2s9FBRoFYK83gf1voS7xSFlzZaFsEGHNmdA0qXbzREKGqUr8fmWmNvBGpDiR2ILQ
ep09lL56JqSCA8owggPGoAMCARKigg09BIIDuarbB67zjmBu9Ax2K+0klSD99pNv97h9yxol
8c6NGLB4CmE8Mo39rL4MMXHe0S00Cbn+TD97XVGLu+cgkfVcuIG4PMMBoajuSnPmIf7qDvfa
8IYDFLDDnRB7I//IXtCc/Z4rBbaxk0SMIRLrsKf5wha7aWtN1JbizBMQw+J0ULN8JjsWxu0L
s75hBtIDbPf3fva3vwBf7kTBChBsheewSAlck9qUglyNxAOdgFVvRrXbfkw51Uo++9qHnhh+
zFSWepfv7U57RYK0Kx0KFd+uauY1ES+xlnMvK18ap2pcy0odBkKu1kwJDND0JXUdSY08MxK2
zb/UGWrVeF6GiSBgu122UGiHp6C+0fEu+nRrvt0RY65Bgi8E1vm55fbb/9dQWncQheL9m6QJ
WPw0rgc+E5S0990N6x3Vv2+Zk17EmbZXinPd2tDe7fJ9cS9o/z7Qjw8z8vvSzHL4GX7FKi2H
JdBST3nEg0C8Pq046UnAJcA8pf1ZkwCK9xDWH+5PSph6WnvpRqugqf/6cq+3jk3MEjCrX+JB
J8QL6AgN3oEB4kvjZpAC+FfTKdX59VLDwfL/r0gMw3ZNk0nLLCLkiiYUMTEHZBzJw9kFbsX3
LmS8qQQA6rQ2L782DYisElywfZ/0Sax8J0/G62Zvy7BHy7SQSGIvcdA0afeNyfxaWM1vTpvS
h0Grnllyfs3FhZAKnVcLYrLPTapR23uLgRMv+0c9wAbwuUDfKg00l5vAd1j55VUyapgDEzi/
URsLdVdcF4zigt4KrTByCwU2/pI6FmEPqB2tsjM2A8JmqA+9Nt8bjaNdNwCOWE0dF50zeL9P

```



```

8oodWGkbRZLk4DLIurpCWld6IyTBhPQ5qZqHJWeoGiFa5y94zBpp27goMPmE0BskXT0JQmve
Yfl0eKEMSzyiWPL2mwi7KEMtfGcPwTIGP2LRE/QxNvPGkwFf0+PDjZGVw+APKkMKqclVXxht
JA/2NmBr01pZIIJ9R+41sR/QoACcXIUXJnhrTwWR1viKCB5Tec87gN+e0Cf0g+fmZuXNRscw
JfhYQJYwJqdYzGtZW+h8jDWqa2EPcDwIQwyFAGXNQ/aMvh1yNTECPLEgrMhYmFAUDLQzI2BD
nfbDftIs0rXjSC0oZn/Uaoqdr4F5sy0rYAXH47bS6MW8CxyylreH8nT2qQXjenakLFHcNjt4
M1n0c/igzNSeZ28qW9WSr4bCdkH+ra3BVpT/AF0WHWkxGF4vWr/iNHCjq8fLF+DsAEx0Zs69
6Rg0fWZy079A\r\nUser-Agent: xmlrpclib.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><array><data>\n<value><string>jrussell</string></value>\n</data>
</array></value>\n</param>\n<param>\n<value>
<struct>\n<member>\n<name>all</name>\n<value><boolean>0</boolean>
</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>Jane
Russell</string>
</value>\n</member>\n<member>\n<name>noprivate</name>\n<value>
<boolean>0</boolean>
</value>\n</member>\n<member>\n<name>uidnumber</name>\n<value>
<int>999</int></value>\n</member>\n<member>\n<name>raw</name>\n<value>
<boolean>0</boolean>
</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>Russell</string>
</value>\n</member>\n<member>\n<name>krbprincipalname</name>\n<value>
<string>jrussell@EXAMPLE.COM</string>
</value>\n</member>\n<member>\n<name>givenname</name>\n<value>
<string>Jane</string>
</value>\n</member>\n<member>\n<name>initials</name>\n<value>
<string>JR</string></value>\n</member>\n</struct>
</value>\n</param>\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+iejB4oAMCARKicQRvVl5x6Zt9PbWN
zvPEWkdu+3PTCq/ZVKjGHM+1zDBz81GL/f+/Pr75zTuveLYn9de0C3k27vz96fn2HQsy9qVH
7sfqn0RWGQWzl+kDkuD6bJ/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</name>\n<value>
<struct>\n<member>\n<name>dn</name>\n<value>
<string>uid=jrussell,cn=users,cn=accounts,dc=example,dc=com</string>
</value>\n</member>\n<member>\n<name>has_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></array>
</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<value>
<string>organizationalperson</string></value>\n<value>
<string>inetorgperson</string></value>\n<value><string>inetuser</string>
</value>\n<value><string>posixaccount</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<member>\n<name>loginshell</name>\n<value><array>
<data>\n<value><string>/bin/sh</string></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<value><string>JR</string></value>\n</data></array>
</value>\n</member>\n<member>\n<name>gidnumber</name>\n<value><array>
<data>\n<value><string>1966800004</string></value>\n</data></array>
</value>\n</member>\n<member>\n<name>gecos</name>\n<value><array>
<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></array>
</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</member>\n<member>\n<name>has_password</name>\n<value>
<boolean>0</'
body: 'boolean>
</value>\n</member>\n<member>\n<name>krbprincipalname</name>\n<value>
<array><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<name>cn</name>\n<value><array>
<data>\n<value><string>Jane Russell</string></value>\n</data></array>
</value>\n</member>\n<member>\n<name>ipauniqueid</name>\n<value><array>
<data>\n<value><string>bba27e6e-df34-11e0-a5f4-001143d2c060</string>
</value>\n</data></array></value>\n</member>\n</struct>
</value>\n</member>\n<member>\n<name>value</name>\n<value>
<string>jrussell</string>
</value>\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**.

## Chapter 26. Managing Certificates and Certificate Authorities

Almost every IdM topology includes an integrated Dogtag Certificate System to manage certificates for servers, replicas, hosts, users, and services within the IdM domain. The Dogtag Certificate System configuration itself may require changes as the domain and the physical machines change.

### 26.1. Renewal Messages

All certificates issued by the IdM servers, such as host and user certificates or subsystem and server certificates used by internal IdM services, are tracked by **certmonger** and automatically renewed as they near expiration.

As a certificate nears its expiration, **certmonger** logs messages in `/var/log/message`, for example:

```
certmonger: Certificate named "NSS Certificate DB" in token
"auditSigningCert cert-pki-ca" in database "/var/lib/pki-ca/alias" will
not be valid after 20160204065136.
```

Once a certificate is renewed, **certmonger** records another message to indicate that the renewal operation has succeeded (or failed), for example:

```
Certificate named "NSS Certificate DB" in token "auditSigningCert cert-
pki-ca" in database "/var/lib/pki-ca/alias" renew success
```

### 26.2. Automatic CA Certificate Renewal

If you are using a root CA certificate managed internally by Dogtag, the **certmonger** utility automatically renews the CA certificate when it is nearing expiration. For more information on how **certmonger** monitors certificate expiration dates, see [the corresponding chapter in the System-Level Authentication Guide](#).

Certificates signed by an external CA cannot be automatically renewed by **certmonger**. You have to renew these certificates manually.

### 26.3. Manual CA Certificate Renewal

You can use the **ipa-cacert-manage** utility to manually renew:

- the self-signed Dogtag CA certificate
- the Dogtag CA certificate signed by an external CA

The renewed certificates created with the **ipa-cacert-manage renew** command use the same key pair and subject name as the old certificates. Renewing a certificate does not remove its previous version to enable certificate rollover.

To manually renew the self-signed Dogtag CA certificate:

1. Run the **ipa-cacert-manage renew** command. The command does not require you to specify the path to the certificate.
2. The renewed certificate is now present in the LDAP certificate store and in the **/etc/pki/pki-tomcat/alias** NSS database.
3. Run the **ipa-certupdate** utility on all clients to update them with the information about the new certificate from LDAP. You have to run **ipa-certupdate** on every client separately.

To manually renew the Dogtag CA certificate signed by an external CA:

1. Run the **ipa-cacert-manage renew** command.
2. The command creates the **/var/lib/ipa/ca.crt** CSR file. Sign the CSR file with the external CA to get the renewed CA certificate. For information about signing the CSR file with an external CA, see [Section 3.2.3.2, “Installing Using an External CA”](#).
3. Run **ipa-cacert-manage renew** again, and this time specify the renewed CA certificate and the external CA certificate chain files using the **--external-cert-file** option. For example:

```
[root@server ~]# ipa-cacert-manage renew --external-cert-file
path/to/signed/certificate
```

4. The renewed CA certificate and the external CA certificate chain are now present in the LDAP certificate store and in the **/etc/pki/pki-tomcat/alias** NSS database.
5. Run the **ipa-certupdate** utility on all clients to update them with the information about the new certificate from LDAP. You have to run **ipa-certupdate** on every client separately.



### Important

If you do not run **ipa-certupdate** after renewing a certificate manually, the renewed certificate will not be distributed to clients.

You can make sure the renewed certificate is properly installed and present in the NSS database by using the **certutil** utility to list the certificates in the database. For example:

```
[root@server ~]# certutil -L -d /etc/pki/pki-tomcat/alias
```

## 26.4. Changing Certificate Chaining

When renewing a certificate with the **ipa-cacert-manage renew** command, you can also modify the certificate chaining. It is possible to:

- ✧ renew the self-signed Dogtag CA certificate as a CA certificate signed by an external CA
- ✧ renew the Dogtag CA certificate signed by an external CA as a self-signed CA certificate

To renew the self-signed Dogtag CA certificate as a CA certificate signed by an external CA, add the **--external-ca** option to **ipa-cacert-manage renew**. The rest of the procedure is the same as manually renewing an externally-signed certificate, which is described in [Section 26.3, “Manual CA Certificate Renewal”](#).

To renew the Dogtag CA certificate signed by an external CA as a self-signed Dogtag CA certificate, add the **--self-signed** option to **ipa-cacert-manage renew**.

## 26.5. 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 ~]# systemctl restart httpd.service
```

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 ~]# systemctl restart dirsrv.target
```

## 26.6. 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.



### Note

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
```

## 26.7. Promoting a Replica to a Master CA Server

The only difference between a master server and a replica is that only the master CA manages renewal of CA subsystem certificates and generates CRLs which are distributed among the other servers and replicas in the topology. Otherwise, servers and replicas are equal peers in the server topology.

If the original server is going to be taken offline or decommissioned, a replica needs to be configured to take its place because there always must be one instance somewhere in the IdM topology which issues CRLs. *Promoting* a replica to a master server changes its configuration and enables it to function as the root CA.

The first IdM server installed owns the master CA in the PKI hierarchy. Servers are almost always created to host CA services. These are the *original* CA services.



## Note

The only exception to this is if system certificates are manually loaded during the installation for a CA-less installation. Otherwise, a Certificate System instance is installed and configured.

A replica *can* host CA services, but this is not required. Servers and replicas which host a CA are also equal peers in the topology. They can all issue certificates and keys to IdM clients, and they all replicate information amongst themselves.

When the first server is installed, it is configured to issue CRLs. In its CA configuration file at `/etc/pki/pki-tomcat/ca/CS.cfg`, it has CRL generation enabled:

```
ca.crl.issuingPointId.enableCRLCache=true
ca.crl.issuingPointId.enableCRLUpdates=true
ca.listenToCloneModifications=false
```

All replicas point to that master CA as the source for CRL information and disable the CRL settings:

```
ca.crl.issuingPointId.enableCRLUpdates=false
```

To promote a replica to a master CA, you must change which server handles certificate renewal and which server generates CRLs.

## Changing Which Server Handles Certificate Renewal

1. To determine the host name of the current renewal master, use the `ldapsearch` utility. In the following example, it is `server.example.com`:

```
$ ldapsearch -H ldap://$HOSTNAME -D 'cn=Directory Manager' -W -b
'cn=masters,cn=ipa,cn=etc,dc=example,dc=com' '(0(cn=CA)
(ipaConfigString=caRenewalMaster))' dn
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <cn=masters,cn=ipa,cn=etc,dc=example,dc=com> with scope
subtree
# filter: (0(cn=CA)(ipaConfigString=caRenewalMaster))
# requesting: dn
#
# CA, server.example.com, masters, ipa, etc, example.com
dn:
```



```
cn=CA,cn=server.example.com,cn=masters,cn=ipa,cn=etc,dc=example,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

2. Configure CA renewal on the new master using the **ipa-csreplica-manage** utility:

```
# ipa-csreplica-manage set-renewal-master
```



## Note

You are not required to reconfigure the current CA as a clone to manually decommission it. Clone renewal is configured automatically when you set up another CA as the renewal master server.

## Changing Which Server Generates CRLs

1. To identify the current CRL generation master, examine the **CS.cfg** on each CA. For example:

```
# grep ca.crl.MasterCRL.enableCRLUpdates /etc/pki/pki-
tomcat/ca/CS.cfg
ca.crl.MasterCRL.enableCRLUpdates=true
```

The **ca.crl.MasterCRL.enableCRLUpdates** parameter is set to **true** on the CRL generation master. On clones, it is set to **false**.

2. Stop CRL generation on the current CRL generation master.

- a. Stop the CA service:

```
# systemctl stop pki-tomcatd@pki-tomcat.service
```

- b. Set the values of the **ca.crl.MasterCRL.enableCRLCache** and **ca.crl.MasterCRL.enableCRLUpdates** parameters to **false** in the **/etc/pki/pki-tomcat/ca/CS.cfg** file to disable CRL generation:

```
ca.crl.MasterCRL.enableCRLCache=false
ca.crl.MasterCRL.enableCRLUpdates=false
```

- c. Start the CA service:

```
# systemctl start pki-tomcatd@pki-tomcat.service
```

- d. Configure Apache to redirect CRL requests to the new master by uncommenting the **RewriteRule** on the last line of the **/etc/httpd/conf.d/ipa-pki-proxy.conf** file:

```
# Only enable this on servers that are not generating a CRL
RewriteRule ^/ipa/crl/MasterCRL.bin
https://<hostname>/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL [L,R=301,NC]
```

- e. Restart Apache:

```
# systemctl restart httpd.service
```

3. Configure a replica to generate CRLs as the new master:

- a. Stop the CA service:

```
# systemctl stop pki-tomcatd@pki-tomcat.service
```

- b. Set the values of the **ca.crl.MasterCRL.enableCRLCache** and **ca.crl.MasterCRL.enableCRLUpdates** parameters to **true** in **/etc/pki/pki-tomcat/ca/CS.cfg** to enable CRL generation:

```
ca.crl.MasterCRL.enableCRLCache=true
ca.crl.MasterCRL.enableCRLUpdates=true
```

- c. Start the CA service:

```
# systemctl start pki-tomcatd@pki-tomcat.service
```

- d. Configure Apache to disable redirecting CRL requests by commenting out the **RewriteRule** argument on the last line of the **/etc/httpd/conf.d/ipa-pki-proxy.conf** file:

```
#RewriteRule ^/ipa/crl/MasterCRL.bin
https://server.example.com/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL [L,R=301,NC]
```

As a clone, all CRL requests were routed to the original master. As the new master, this instance will respond to CRL requests.

- e. Restart Apache:

```
# systemctl restart httpd.service
```

## 26.8. Configuring 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.

## 26.8.1. 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
```

## 26.8.2. 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.

```
[root@server ~]# systemctl stop pki-tomcatd@pki-tomcat.service
```

2. Open the **CS.cfg** file.

```
[root@server ~]# vim /var/lib/pki/pki-tomcat/conf/ca/CS.cfg
```

3. Change the **`ca.crl.MasterCRL.autoUpdateInterval`** to the new interval setting.
4. Restart the CA server.

```
[root@server ~]# systemctl start pki-tomcatd@pki-tomcat.service
```

## 26.9. Certificate Profiles

A certificate profile defines the content of certificates belonging to the particular profile, as well as constraints for issuing the certificates, enrollment method, and input and output forms for enrollment. A single certificate profile is associated with issuing a particular type of certificate. Different certificate profiles can be defined for users, services, and hosts in IdM.

The CA uses certificate profiles in signing of certificates to determine:

- ✧ whether the CA can accept a certificate signing request (CSR)
- ✧ what features and extensions should be present on the certificate

IdM includes the following two certificate profiles by default: **`caIPAserviceCert`** and **`IECUserRoles`**. In addition, custom profiles can be imported.

Custom certificate profiles allow issuing certificates for specific, unrelated purposes. For example, it is possible to restrict use of a particular profile to only one user or one group, preventing other users and groups from using that profile to issue a certificate for authentication.



### Note

By combining certificate profiles and CA ACLs, [Section 26.10, “Certificate Authority ACL Rules”](#), the administrator can define and control access to custom certificate profiles. For a description of using profiles and CA ACLs to issue user certificates, see [Section 10.11, “Issuing User Certificates with the IdM CA”](#).

### 26.9.1. Certificate Profile Management from the Command Line

The **`certprofile`** plug-in for management of IdM profiles allows privileged users to import, modify, or remove IdM certificate profiles. To display all commands supported by the plug-in, run the **`ipa certprofile`** command:

```
$ ipa certprofile
Manage Certificate Profiles
```

```
...
```

#### EXAMPLES:

```
Import a profile that will not store issued certificates:
ipa certprofile-import ShortLivedUserCert \
  --file UserCert.profile --desc "User Certificates" \
  --store=false
```

```
Delete a certificate profile:
ipa certprofile-del ShortLivedUserCert
...
```

Note that to perform the **certprofile** operations, you must be operating as a user who has the required permissions. IdM includes the following certificate profile-related permissions by default:

#### System: Read Certificate Profiles

Enables users to read all profile attributes.

#### System: Import Certificate Profile

Enables users to import a certificate profile into IdM.

#### System: Delete Certificate Profile

Enables users to delete an existing certificate profile.

#### System: Modify Certificate Profile

Enables users to modify the profile attributes and to disable or enable the profile.

All these permissions are included in the default **CA Administrator** privilege. For more information on IdM role-based access controls and managing permissions, see [Section 24.4, “Defining Role-Based Access Controls”](#).



### Note

When requesting a certificate, the **--profile-id** option can be added to the **ipa cert-request** command to specify which profile to use. If no profile ID is specified, the default **caIPAServiceCert** profile is used for the certificate.

This section only describes the most important aspects of using the **ipa certprofile** commands for profile management. For complete information about a command, run it with the **--help** option added, for example:

```
$ ipa certprofile-mod --help
Usage: ipa [global-options] certprofile-mod ID [options]

Modify Certificate Profile configuration.
Options:
  -h, --help          show this help message and exit
  --desc=STR          Brief description of this profile
  --store=BOOL        Whether to store certs issued using this profile
...
```

## Importing Certificate Profiles

To import a new certificate profile to IdM, use the **ipa certprofile-import** command. Running the command without any options starts an interactive session in which the **certprofile-import** script prompts you for the information required to import the certificate.

```
$ ipa certprofile-import

Profile ID: smime
Profile description: S/MIME certificates
Store issued certificates [True]: TRUE
Filename of a raw profile. The XML format is not supported.: smime.cfg
-----
Imported profile "smime"
-----
Profile ID: smime
Profile description: S/MIME certificates
Store issued certificates: TRUE
```

The **ipa certprofile-import** command accepts several command-line options. Most notably:

#### **--file**

This option passes the file containing the profile configuration directly to **ipa certprofile-import**. For example:

```
$ ipa certprofile-import --file=smime.cfg
```

#### **--store**

This option sets the **Store issued certificates** attribute. It accepts two values:

- **True**, which delivers the issued certificates to the client and stores them in the target IdM principal's **userCertificate** attribute.
- **False**, which delivers the issued certificates to the client, but does not store them in IdM. This option is most commonly-used when issuing multiple short-term certificates is required.

Import fails if the profile ID specified with **ipa certprofile-import** is already in use or if the profile content is incorrect. For example, the import fails if a required attribute is missing or if the profile ID value defined in the supplied file does not match the profile ID specified with **ipa certprofile-import**.

To obtain a template for a new profile, you can run the **ipa certprofile-show** command with the **--out** option, which exports a specified existing profile to a file. For example:

```
$ ipa certprofile-show caIPAServiceCert --out=file_name
```

You can then edit the exported file as required and import it as a new profile.

## Displaying Certificate Profiles

To display all certificate profiles currently stored in IdM, use the **ipa certprofile-find** command:

```
$ ipa certprofile-find
-----
3 profiles matched
-----
Profile ID: caIPAServiceCert
```

```
Profile description: Standard profile for network services
Store issued certificates: TRUE
```

```
Profile ID: IECUserRoles
```

```
...
```

To display information about a particular profile, use the **ipa certprofile-show** command:

```
$ ipa certprofile-show profile_ID
Profile ID: profile_ID
Profile description: S/MIME certificates
Store issued certificates: TRUE
```

## Modifying Certificate Profiles

To modify an existing certificate profile, use the **ipa certprofile-mod** command. Pass the required modifications with the command using the command-line options accepted by **ipa certprofile-mod**. For example, to modify the description of a profile and change whether IdM stores the issued certificates:

```
$ ipa certprofile-mod profile_ID --desc="New description" --store=False
-----
Modified Certificate Profile "profile_ID"
-----
Profile ID: profile_ID
Profile description: New description
Store issued certificates: FALSE
```

To update the certificate profile configuration, import the file containing the updated configuration using the **--file** option. For example:

```
$ ipa certprofile-import profile_ID --file=new_configuration.cfg
```

## Deleting Certificate Profiles

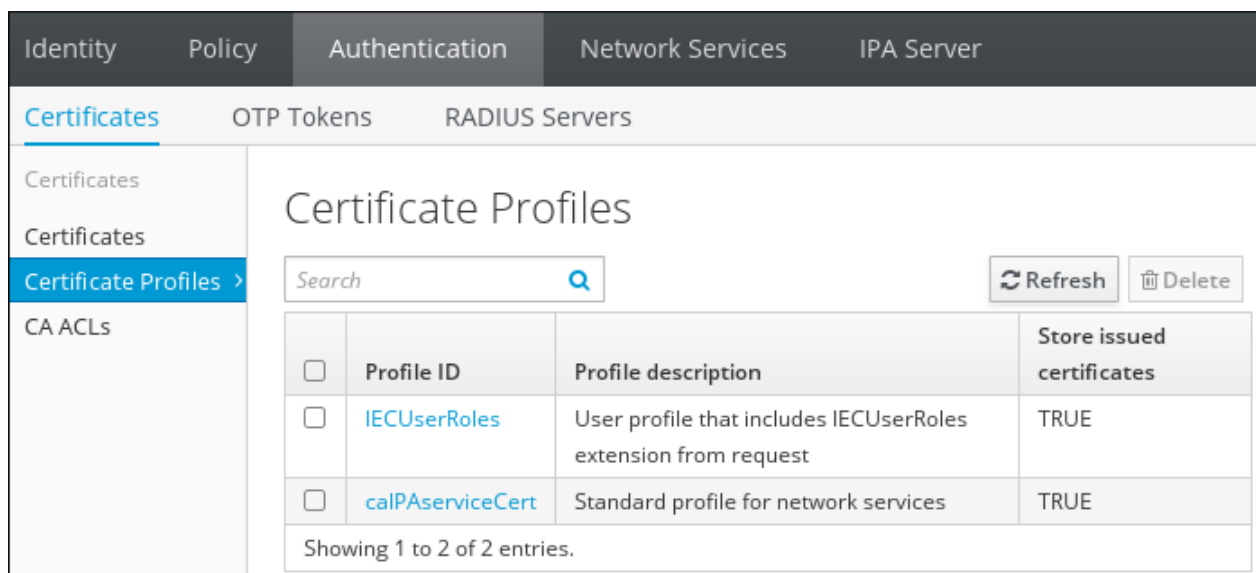
To remove an existing certificate profile from IdM, use the **ipa certprofile-del** command:

```
$ ipa certprofile-del profile_ID
-----
Deleted profile "profile_ID"
-----
```

### 26.9.2. Certificate Profile Management from the Web UI

To manage certificate profiles from the IdM web UI:

1. Open the **Authentication** tab and the **Certificates** subtab.
2. Open the **Certificate Profiles** section.

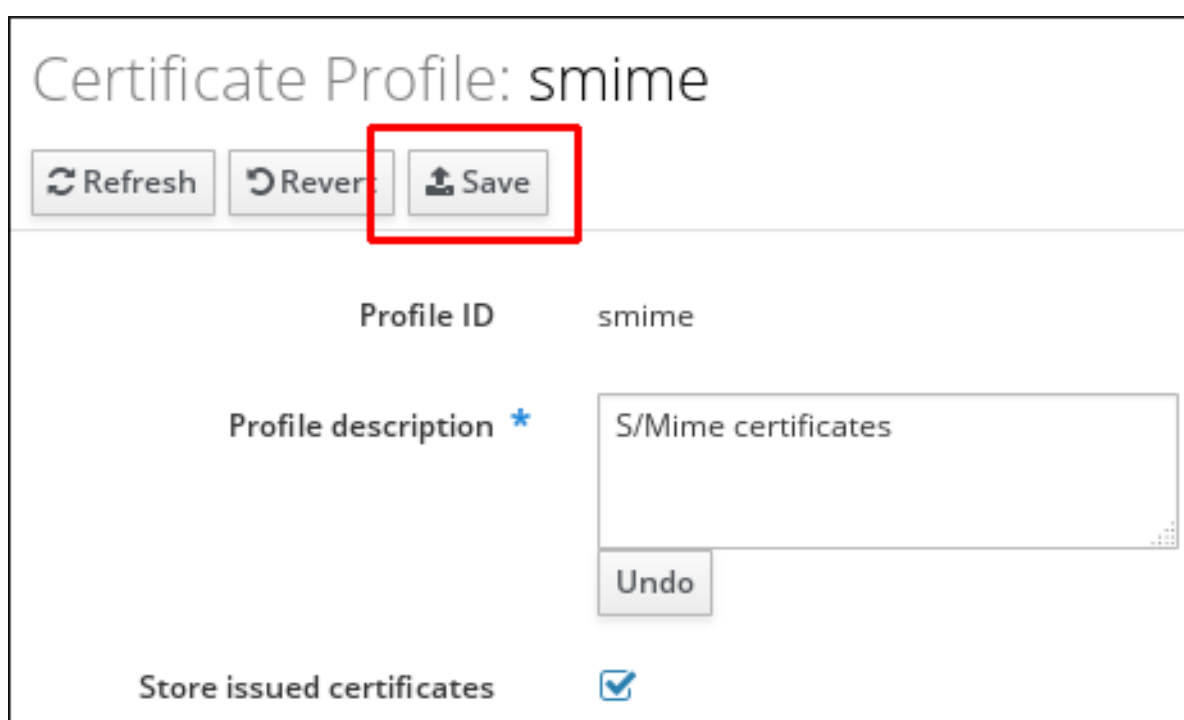


**Figure 26.1. Certificate Profile Management in the Web UI**

In the **Certificate Profiles** section, you can display information about existing profiles, modify their attributes, or delete selected profiles.

For example, to modify an existing certificate profile:

1. Click on the name of the profile to open the profile configuration page.
2. In the profile configuration page, fill in the required information.
3. Click **Save** to confirm the new configuration.



**Figure 26.2. Modifying a Certificate Profile in the Web UI**

If you enable the **Store issued certificates** option, the issued certificates are delivered to the client as well as stored in the target IdM principal's **userCertificate**



attribute. If the option is disabled, the issued certificates are delivered to the client, but not stored in IdM. Storing certificates is often disabled when issuing multiple short-lived certificates is required.

Note that some certificate profile management operations are currently unavailable in the web UI:

- ✧ It is not possible to import a certificate profile in the web UI. To import a certificate, use the **ipa certprofile-import** command.
- ✧ It is not possible to set, add, or delete attribute and value pairs. To modify the attribute and value pairs, use the **ipa certprofile-mod** command.
- ✧ It is not possible to import updated certificate profile configuration. To import a file containing updated profile configuration, use the **ipa certprofile-mod --file=file\_name** command.

For more information about the commands used to manage certificate profiles, see [Section 26.9.1, “Certificate Profile Management from the Command Line”](#).

### 26.9.3. Upgrading IdM Servers with Certificate Profiles

When upgrading an IdM server, the profiles included in the server are all imported and enabled.

If you upgrade multiple server replicas, the profiles of the first upgraded replica are imported. On the other replicas, IdM detects the presence of other profiles and does not import them or resolve any conflicts between the two sets of profiles. If you have custom profiles defined on replicas, make sure the profiles on all replicas are consistent before upgrading.

## 26.10. Certificate Authority ACL Rules

Certificate Authority access control list (CA ACL) rules define which profiles can be used to issue certificates to which users, services, or hosts. By associating profiles, principals, and groups, CA ACLs permit principals or groups to request certificates using particular profiles:

- ✧ an ACL can permit access to multiple profiles
- ✧ an ACL can have multiple users, services, hosts, user groups, and host groups associated with it

For example, using CA ACLs, the administrator can restrict use of a profile intended for employees working from an office located in London only to hosts that are members of the London office-related group.



### Note

By combining certificate profiles, described in [Section 26.9, “Certificate Profiles”](#), and CA ACLs, the administrator can define and control access to custom certificate profiles. For a description of using profiles and CA ACLs to issue user certificates, see [Section 10.11, “Issuing User Certificates with the IdM CA”](#).

## 26.10.1. CA ACL Management from the Command Line

The **caacl** plug-in for management of CA ACL rules allows privileged users to add, display, modify, or delete a specified CA ACL. To display all commands supported by the plug-in, run the **ipa caacl** command:

```
$ ipa caacl
Manage CA ACL rules.

...

EXAMPLES:

Create a CA ACL "test" that grants all users access to the
"UserCert" profile:
    ipa caacl-add test --usercat=all
    ipa caacl-add-profile test --certprofiles UserCert

Display the properties of a named CA ACL:
    ipa caacl-show test

...
```

Note that to perform the **caacl** operations, you must be operating as a user who has the required permissions. IdM includes the following CA ACL-related permissions by default:

### System: Read CA ACLs

Enables the user to read all attributes of the CA ACL.

### System: Add CA ACL

Enables the user to add a new CA ACL.

### System: Delete CA ACL

Enables the user to delete an existing CA ACL.

### System: Modify CA ACL

Enables the user to modify an attribute of the CA ACL and to disable or enable the CA ACL.

### System: Manage CA ACL membership

Enables the user to manage the CA, profile, user, host, and service membership in the CA ACL.

All these permissions are included in the default **CA Administrator** privilege. For more information on IdM role-based access controls and managing permissions, see [Section 24.4, “Defining Role-Based Access Controls”](#).

This section describes only the most important aspects of using the **ipa caacl** commands for CA ACL management. For complete information about a command, run it with the **--help** option added, for example:

```
$ ipa caacl-mod --help
Usage: ipa [global-options] caacl-mod NAME [options]
```

Modify a CA ACL.

Options:

-h, --help	show this help message and exit
--desc=STR	Description
--profilecat=['all']	Profile category the ACL applies to
...	

## Creating CA ACLs

To create a new CA ACL, use the **ipa caacl-add** command. Running the command without any options starts an interactive session in which the **ipa caacl-add** script prompts your for the required information about the new CA ACL.

```
$ ipa caacl-add
ACL name: smime_acl
-----
Added CA ACL "smime_acl"
-----
ACL name: smime_acl
Enabled: TRUE
```

New CA ACLs are enabled by default.

The most notable options accepted by **ipa caacl-add** are the options that associate a CA ACL with a certificate profile, user, host, or service category:

- » **--profilecat**
- » **--usercat**
- » **--hostcat**
- » **--servicecat**

IdM only accepts the **all** value with these options, which associates the CA ACL with all profiles, users, hosts, or services. For example, to associate the CA ACL with all users and user groups:

```
$ ipa caacl-add ca_acl_name --usercat=all
```

Profile, user, host, and service categories are an alternative to adding particular objects or groups of objects to a CA ACL, which is described in [Section 26.10.1, “Adding Entries to CA ACLs and Removing Entries from CA ACLs”](#). Note that it is not possible to use a category and also add objects or groups of the same type; for example, you cannot use the **--usercat=all** option and then add a user to the CA ACL with the **ipa caacl-add-user --users=user\_name** command.



## Note

Requesting a certificate for a user or group using a certificate profile fails if the user or group are not added to the corresponding CA ACL. For example:

```
$ ipa cert-request CSR-FILE --principal user --profile-id
profile_id
ipa: ERROR Insufficient access: Principal 'user' is not permitted
to use CA '.' with profile 'profile_id' for certificate issuance.
```

You must either add the user or group to the CA ACL, as described in [Section 26.10.1, “Adding Entries to CA ACLs and Removing Entries from CA ACLs”](#), or associate the CA ACL with the **all** user category.

## Displaying CA ACLs

To display all CA ACLs, use the **ipa caacl-find** command:

```
$ ipa caacl-find
-----
2 CA ACLs matched
-----
ACL name: hosts_services_caIPAserviceCert
Enabled: TRUE
...
```

Note that **ipa caacl-find** accepts the **--profilecat**, **--usercat**, **--hostcat**, and **--servicecat** options, which can be used to filter the results of the search to CA ACLs with the corresponding certificate profile, user, host, or service category. Note that IdM only accepts the **all** category with these options. For more information about the options, see [Section 26.10.1, “Creating CA ACLs”](#).

To display information about a particular CA ACL, use the **ipa caacl-show** command:

```
$ ipa caacl-show ca_acl_name
ACL name: ca_acl_name
Enabled: TRUE
Host category: all
...
```

## Modifying CA ACLs

To modify an existing CA ACL, use the **ipa caacl-mod** command. Pass the required modifications using the command-line options accepted by **ipa caacl-mod**. For example, to modify the description of a CA ACL and associate the CA ACL with all certificate profiles:

```
$ ipa caacl-mod ca_acl_name --desc="New description" --profilecat=all
-----
Modified CA ACL "ca_acl_name"
-----
```

```
ACL name: smime_acl
Description: New description
Enabled: TRUE
Profile category: all
```

The most notable options accepted by **ipa caacl-mod** are the **--profilecat**, **--usercat**, **--hostcat**, and **--servicecat** options. For a description of these options, see [Section 26.10.1, “Creating CA ACLs”](#).

## Disabling and Enabling CA ACLs

To disable a CA ACL, use the **ipa caacl-disable** command:

```
$ ipa caacl-disable ca_acl_name
-----
Disabled CA ACL "ca_acl_name"
-----
```

A disabled CA ACL is not applied and cannot be used to request a certificate. Disabling a CA ACL does not remove it from IdM.

To enable a disabled CA ACL, use the **ipa caacl-enable** command:

```
$ ipa caacl-enable ca_acl_name
-----
Enabled CA ACL "ca_acl_name"
-----
```

## Deleting CA ACLs

To remove an existing CA ACL, use the **ipa caacl-del** command:

```
$ ipa caacl-del ca_acl_name
```

## Adding Entries to CA ACLs and Removing Entries from CA ACLs

Using the **ipa caacl-add-\*** and **ipa caacl-remove-\*** commands, you can add new entries to a CA ACL or remove existing entries.

### **ipa caacl-add-host** and **ipa caacl-remove-host**

Adds or removes a host or host group.

### **ipa caacl-add-profile** and **ipa caacl-remove-profile**

Adds or removes a profile.

### **ipa caacl-add-service** and **ipa caacl-remove-service**

Adds or removes a service.

### **ipa caacl-add-user** and **ipa caacl-remove-user**

Adds or removes a user or group.

For example:

```
$ ipa caacl-add-user ca_acl_name --groups=group_name
```

Note that it is not possible to add an object or a group of objects to a CA ACL and also use a category of the same object, as described in [Section 26.10.1, “Creating CA ACLs”](#); these settings are mutually exclusive. For example, if you attempt to run the **ipa caacl-add-user --users=user\_name** command on a CA ACL specified with the **--usercat=all** option, the command fails:

```
$ ipa caacl-add-user ca_acl_name --users=user_name
ipa: ERROR: users cannot be added when user category='all'
```



## Note

Requesting a certificate for a user or group using a certificate profile fails if the user or group are not added to the corresponding CA ACL. For example:

```
$ ipa cert-request CSR-FILE --principal user --profile-id
profile_id
ipa: ERROR Insufficient access: Principal 'user' is not permitted
to use CA '.' with profile 'profile_id' for certificate issuance.
```

You must either add the user or group to the CA ACL, or associate the CA ACL with the **all** user category, as described in [Section 26.10.1, “Creating CA ACLs”](#).

For detailed information on the required syntax for these commands and the available options, run the commands with the **--help** option added. For example:

```
$ ipa caacl-add-user --help
```

## 26.10.2. CA ACL Management from the Web UI

To manage CA ACLs from the IdM web UI:

1. Open the **Authentication** tab and the **Certificates** subtab.
2. Open the **CA ACLs** section.

ACL name	Status	Description
<input type="checkbox"/> hosts_services_calPAserviceCert	✓ Enabled	

Showing 1 to 1 of 1 entries.

**Figure 26.3. CA ACL Rules Management in the Web UI**

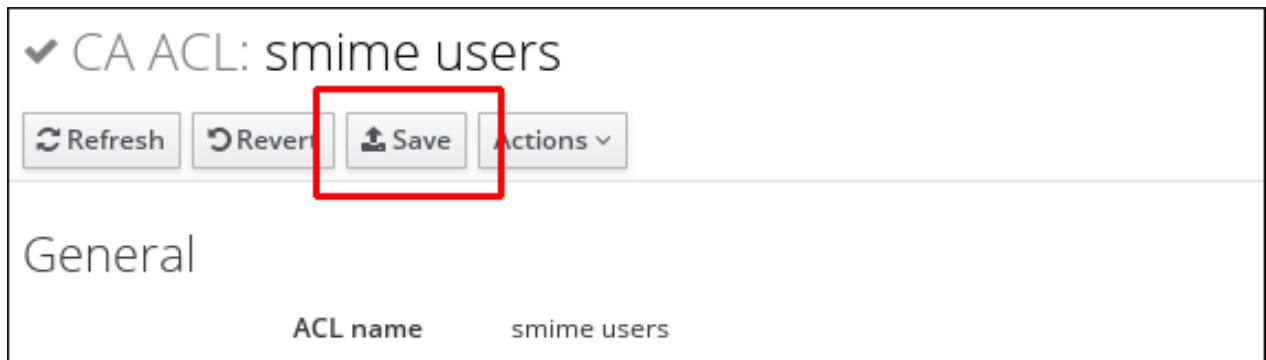
In the **CA ACLs** section, you can add new CA ACLs, display information about existing CA ACLs, modify their attributes, as well as enable, disable, or delete selected CA ACLs.

For example, to modify an existing CA ACL:

1. Click on the name of the CA ACL to open the CA ACL configuration page.
2. In the CA ACL configuration page, fill in the required information.

The **Profiles** and **Permitted to have certificates issued** sections allow you to associate the CA ACL with certificate profiles, users or user groups, hosts or host groups, or services. You can either add these objects using the **Add** buttons, or select the **Anyone** option to associate the CA ACL with all users, hosts, or services.

3. Click **Save** to confirm the new configuration.



**Figure 26.4. Modifying a CA ACL Rule in the Web UI**

## Chapter 27. 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 -h server.example.com
-p 389 -ZZ
Enter LDAP Password:
dn: cn=config
changetype: modify
replace: nsslapd-allow-anonymous-access
nsslapd-allow-anonymous-access: rootdse

modifying entry "cn=config"
```



### 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.

```
# systemctl restart dirsrv.target
```



## Chapter 28. Changing Domain DNS Configuration

### 28.1. 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
```

### 28.2. 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-pkcs11` 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.

### 28.3. Changing Load Balancing for IdM Servers and Replicas

As [Section 1.3.1, “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](#).

## Chapter 29. Managing the Server-Replica Relationships

[Section 1.3.1, “IdM Servers and Replicas”](#) describes the relationship between servers (original instances) and replicas (copied instances) in Identity Management. This network of related servers and replicas is the **topology** of the Identity Management domain.

The topology is defined by a series of agreements set between IdM servers and replicas that copy data between instances. These replication agreements identify what servers and replicas are active in the topology (meaning, recognized by other servers and sending and updating information).

Changing the IdM topology by adding or removing replicas and servers is done by managing the replication agreements between instances. These replication agreements are created between the master server and the replicas automatically by the **ipa-replica-install** command as replicas are created. When replicas are removed or when two new replicas need to communicate with each other, those replication agreements must be managed manually.

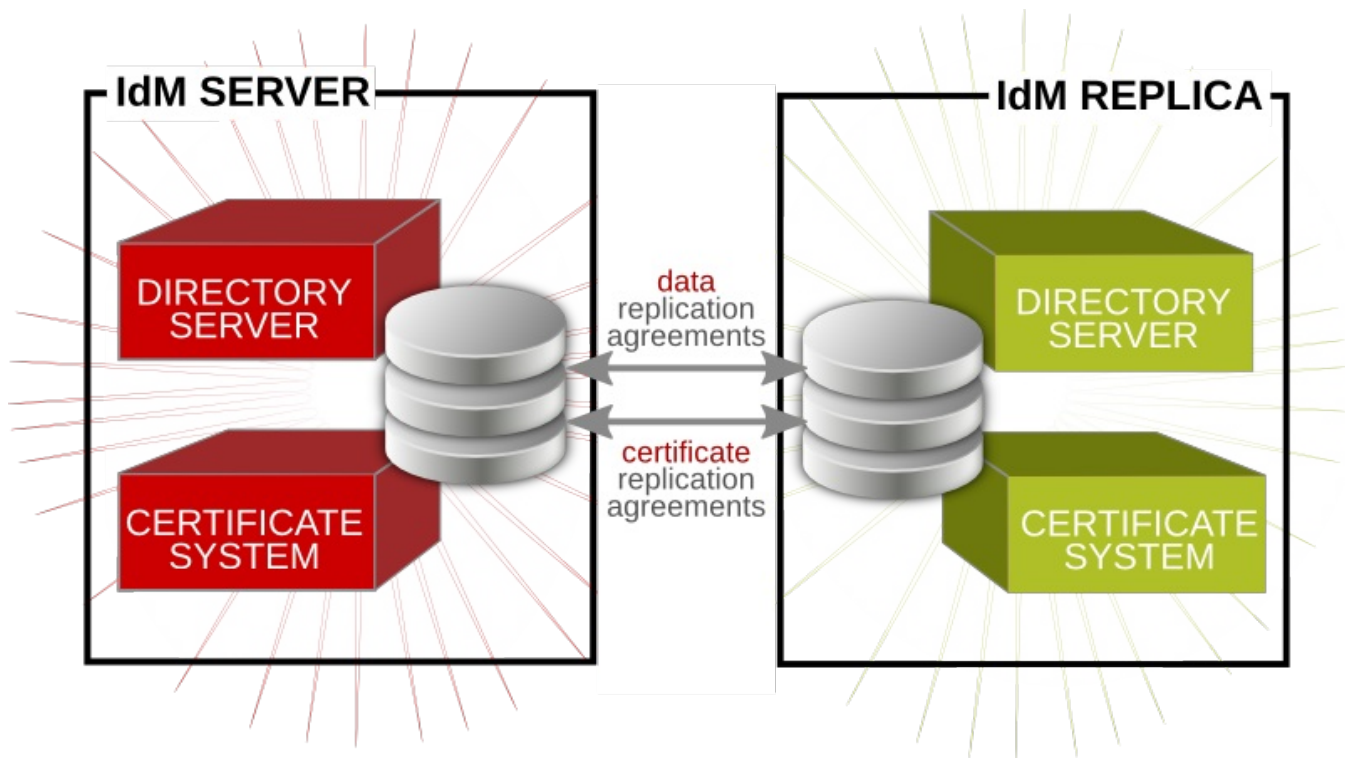
### 29.1. Managing Replication Agreements Between IdM Servers

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*.

#### 29.1.1. The Topology of Replication Agreements

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 29.1. 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.

### 29.1.2. Types of Replication Agreements

There are three types of replication agreements for IdM servers:

- ✧ Ones to replicate directory data (such as users, groups, and policies)
- ✧ Ones to replicate user information with an Active Directory server (a synchronization agreement)
- ✧ Ones to replicate certificate and key data

### 29.1.3. Commands to Manage Replication Agreements

Agreements for both the directory data and the synchronized user data are managed using the **ipa-replica-manage** command. Agreements for the certificate and key data are managed using the **ipa-csreplica-manage** command.

These tools have the same commands, arguments, and format. The differences relate to which subtree within the IdM directory is being replicated.

**Table 29.1. Replica Management Commands**

Command	Description
connect	Create a new replication agreement between the two specified servers.
disconnect	Removes a replication agreement between the two specified servers.
del	Removes all replication agreements for the given server and removes it entirely from the replication topology. This is used to decommission a server/replica, not simply to change the replication agreements for it.
list	Lists the replication agreements. If no server is given, then it lists all servers involved in the replication topology. If a server is specified, then it lists all of the servers with which it has a replication agreement.
re-initialize	Essentially restarts replication for the given server. It retrieves all of the replicated data from the original source.
force-sync	Forces an immediate, incremental update (replication event) for the specified server.
list-ruv	Lists the replication ID (a backend identifier) for each server within the replication topology. <i>For the ipa-replica-manage command only.</i>
clean-ruv	Runs a special task to remove all outstanding updates associated with a given replication ID. <i>For the ipa-replica-manage command only.</i>

### 29.1.4. Listing Replication Agreements

The **ipa-replica-manage** command can list all of the servers and replicas in the replication topology, using the **list** command:

```
[root@server ~]# ipa-replica-manage list
srv1.example.com: master
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.

```
[root@server ~]# ipa-replica-manage list srv1.example.com
srv2.example.com
srv3.example.com
```

The same thing can be done for certificate replication agreements by using the **ipa-csreplica-manage** command.

### 29.1.5. Creating Replication Agreements

Replication agreements are created by *connecting* one server to another server. When a replica is created from a master server, those two servers have a replication agreement between them. However, other servers within the topology do not have a replication agreement with the new replica. While data will most likely be replicated across the topology eventually, adding additional replication agreements can improve performance and provide additional failover. (In some topologies, and depending on how replicas are cloned from a master, some changes could still be missed without additional replication agreements.)

A new replication agreement is created using the **connect** command.

```
ipa-replica-manage connect server1 server2
```

If only one server is given, the replication agreements are created between the local host and the specified server.

For example:

```
[root@server ~]# 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:

```
[root@server ~]# ipa-replica-manage connect --cacert=/etc/ipa/ca.crt  
srv2.example.com srv4.example.com
```

The same thing can be done for certificate replication agreements by using the **ipa-csreplica-manage** command.

### 29.1.6. Removing Replication Agreements

To remove a replication agreement between specific servers/replicas, use the **disconnect** command:

```
[root@server ~]# 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** command:

```
[root@server ~]# ipa-replica-manage del srv2.example.com
```

The same thing can be done for certificate replication agreements by using the **ipa-csreplica-manage** command.

### 29.1.7. 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.

```
[root@server ~]# ipa-replica-manage force-sync --from srv1.example.com
```

The same thing can be done for certificate replication agreements by using the **ipa-csreplica-manage** command.

### 29.1.8. 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:

```
[root@server ~]# ipa-replica-manage re-initialize --from  
srv1.example.com
```

The same thing can be done for certificate replication agreements by using the **ipa-csreplica-manage** command.

### 29.1.9. Resolving Replication Problems

#### 29.1.9.1. Serial Numbers Not Found Errors

The 389 Directory Server and Dogtag Certificate System instances share a single directory database for data. Replication agreements are set up for different suffixes within that directory.

The directory and certificate replication agreements are managed through different tools and are created and removed independently. If a certificate replication agreement is removed, but a data replication agreement is not, there can be problems with using certificates with some directory entries.

For example, both data and certificate replication agreements exist between Server A and Server B. If the certificate agreement is removed, both Server A and Server B still have certificate authorities and are still issuing certificates, but that information is no longer being replicated. If Server A issues a certificate to Host 1, and then someone attempts to use Server B to manage Host 1, Server B returns an error that it cannot verify Host 1's certificate serial number.

```
Certificate operation cannot be completed: EXCEPTION (Certificate serial  
number 0x2d not found)
```

This is because Server B has information about Host 1 in its data directory, but it does not have the host certificate in its certificate directory.

To work around this, enable replication between the two IdM servers.

### 29.1.9.2. 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:

```
[jsmith@ server ~]$ ldapsearch -x -D "cn=directory manager" -w password
-b "dc=example,dc=com" "nsds5ReplConflict=*" \* nsds5ReplConflict
```

#### 29.1.9.2.1. 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**.

#### 29.1.9.2.2. 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.

### 29.1.9.3. Cleaning RUV Errors

Each server records changes to its database in a changelog; each change is assigned an identifier called a *replica update vector* (RUV). The RUVs are a way of identifying where changes come from (the replica) and the order to apply them (through the change state number), as changes are made across multiple servers.

When a server is removed from replication, all of the metadata associated with that server is removed from the other servers' replication configuration. However, if one server is offline when the replication topology is updated, then the metadata (RUVs) for the replica remain in that server's configuration. When replication occurs, that server returns an error because it expects information for a given server (based on the RUVs in its configuration), and that one server is not sending updates any more.

```
[09/Sep/2011:09:03:43 -0600] NSMMReplicationPlugin - ruv_compare_ruv:
RUV [changelog max RUV] does not
  contain element [{replica 55 ldap://localhost.localdomain:9389}
4e6a27ca000000370000 4e6a27e8000000370000]
  which is present in RUV [database RUV]
...
[09/Sep/2011:09:03:43 -0600] NSMMReplicationPlugin -
replica_check_for_data_reload: Warning: for replica
dc=example,dc=com there were some differences between the changelog max
RUV and the database RUV.
```

To resolve those errors, run a **clean-ruv** task to remove any RUVs associated with removed replica. This is run against the replica ID, which would be listed in the 389 Directory Server error logs:

```
...
contain element [{replica 55 ldap://localhost.localdomain:9389}
4e6a27ca000000370000 4e6a27e8000000370000]
...
```

For example:

```
[root@server ~]# ipa-replica-manage clean-ruv 55
```



## Warning

Running a **clean-ruv** task against the wrong replica ID will corrupt all of the data associated with that replica in the replication database. In that case, the replica must be reinitialized to correct the errors; reinitializing a replica is in [Section 29.1.8, “Reinitializing IdM Servers”](#).

## 29.2. 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
```

## 29.3. Renaming a Server or Replica Host System

There is no way to change the hostname for an IdM server or replica machine. The Kerberos keys and certificate management is too complex to allow the hostname to change.

Rather, if a server or replica needs to be renamed, it is easier to replace the instance.

1. Create a new replica, with a CA, with the new hostname or IP address. This is described in [Chapter 4, Setting up IdM Replicas](#).
2. Stop the original IdM server instance.

```
[root@oldserver ~]# ipactl stop
```

3. Verify that all other servers/replicas and clients are working as before.
4. Uninstall the IdM server, as in [Chapter 7, Uninstalling IdM Servers and Replicas](#)

## Chapter 30. 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.

### 30.1. 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 30.1.2, “Planning Password Migration”](#).

#### 30.1.1. 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.



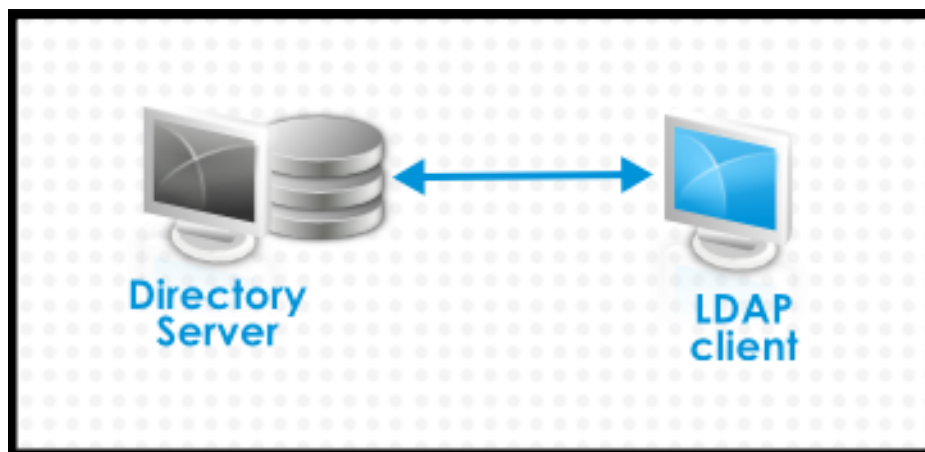
#### Important

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.

##### 30.1.1.1. 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 30.1. 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.

#### **30.1.1.2. Recommended Configuration for Red Hat Enterprise Linux Clients**

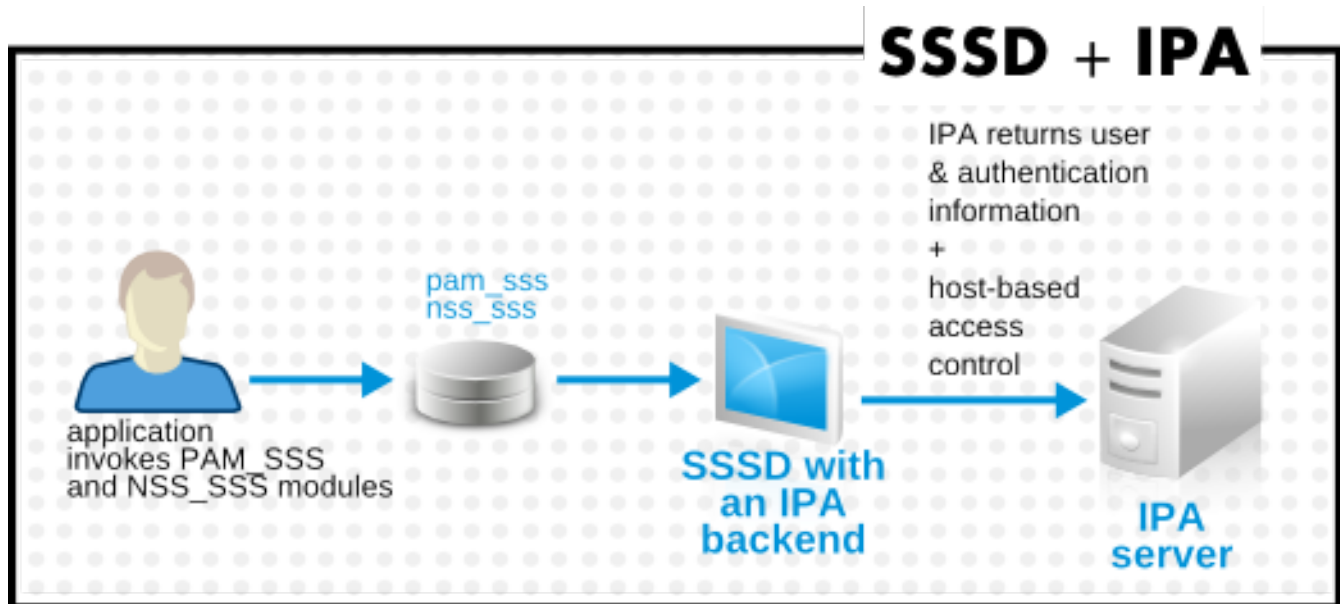
Red Hat Enterprise Linux has a service called the *System Security Services Daemon* (SSSD). SSSD uses special PAM and NSS libraries (**pam\_sss** and **nss\_sss**, respectively) which allow SSSD to be integrated very closely with Identity Management and leverage the full authentication and identity features 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 *System-Level Authentication 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.

## Note

During the migration process from an LDAP directory to Identity Management, SSSD can seamlessly migrate user passwords without additional user interaction.



**Figure 30.2. 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.

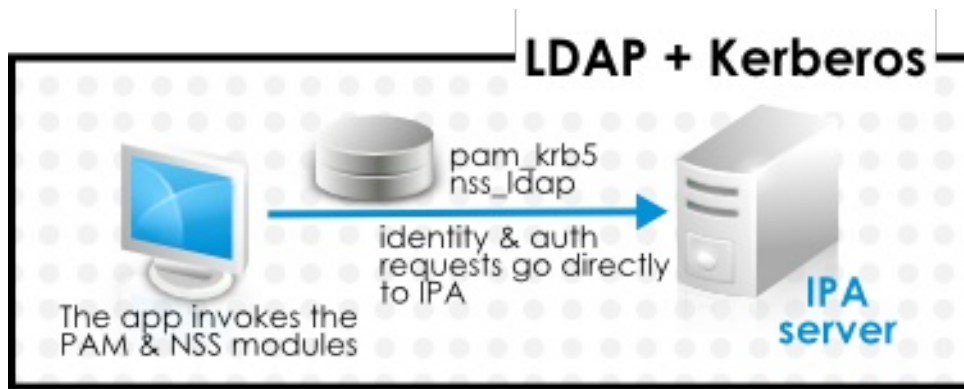
## Note

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 30.1.1.3, “Alternative Supported Configuration”](#).

### 30.1.1.3. 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 30.3. 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 *System-Level Authentication 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).

### 30.1.2. 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 has 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 <sup>[8]</sup>, 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.



### 30.1.2.1. 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.

### 30.1.2.2. 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.

### 30.1.2.3. 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 plain text 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.

### 30.1.2.4. 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 30.1.2.3, “Method 3: Using SSSD \(Recommended\)”](#).

### 30.1.2.5. 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:
```

### 30.1.3. 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.

#### 30.1.3.1. LDAP Servers Supported for Migration

The migration process from an LDAP server to Identity Management uses a special script, **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.

#### 30.1.3.2. 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.

### 30.1.3.3. Migration — IdM System Requirements

With a moderately-sized directory (around 10,000 users and 10 groups), it is necessary to have a powerful enough target system (the IdM system) to allow the migration to proceed. The minimum requirements for a migration are:

- 4 cores
- 4GB of RAM
- 30GB of disk space
- A SASL buffer size of 2MB

This is set in the **nsslapd-sasl-max-buffer-size** attribute in the 389 Directory Server instance for the IdM server. This attribute value is set using the **ldapmodify** command in the **cn=config** subtree.

### 30.1.3.4. 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. When using **ipa migrate-ds**, the remote system user, specified by the **--binddn** option, needs to have read access to the **userPassword** attribute, otherwise passwords will not be migrated.

The Identity Management server must be configured to run in migration mode, and then the migration script can be used.

### 30.1.3.5. Improving Migration Performance

An LDAP migration is essentially a specialized import operation for the 389 Directory Server instance within the IdM server. Tuning the 389 Directory Server instance for better import operation performance can help improve the overall migration performance.

There are two parameters that directly affect import performance:

- The **nsslapd-cachememsize** attribute, which defines the size allowed for the entry cache. This is a buffer, that is automatically set to 80% of the total cache memory size.

For large import operations, this parameter (and possibly the memory cache itself) can be increased to more efficiently handle a large number of entries or entries with larger attributes (such as certificate chains and CRLs).

This can be edited using the **ldapmodify** command; the configuration entries are in **cn=config**.

- ✱ The system **ulimit** setting, which sets the maximum number of allowed processes for the system user. Especially on 32-bit systems, it is possible for the Directory Server user to hit its process limit when trying to process a large database.

```
[root@server ~]# ulimit -u 4096
```

This is covered in the Red Hat Directory Server *Performance Tuning Guide* at [https://access.redhat.com/site/documentation/en-US/Red\\_Hat\\_Directory\\_Server/9.0/html/Performance\\_Tuning\\_Guide/import.html](https://access.redhat.com/site/documentation/en-US/Red_Hat_Directory_Server/9.0/html/Performance_Tuning_Guide/import.html).

### 30.1.3.6. 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.

## 30.2. 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.

### 30.2.1. 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.

### 30.2.2. 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 --group-objectclass=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 --exclude-users=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 --exclude-users=bjensen --exclude-users=mreynolds ldap://ldap.example.com:389
```

### 30.2.3. 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 **ipa 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.

### 30.2.4. Setting the Schema to Use

By default, Identity Management uses [RFC2307bis](#) schema to define user, host, host group, 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
```

## 30.3. 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:

```
[root@server]# 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 <sup>[9]</sup>, on a different machine from the existing LDAP directory.
  3. Enable the IdM server to allow migration:

```
[root@server]# ipa config-mod --enable-migration=TRUE
```

4. Disable the compat plug-in.

```
[root@server]# ipa-compat-manage disable
```

5. Restart the IdM Directory Server instance.

```
[root@server]# systemctl restart dirsrv.target
```

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:

```
[root@server]# 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 30.2, “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.

```
[root@server]# ipa-compat-manage enable
```

8. Restart the IdM Directory Server instance.

```
[root@server]# systemctl restart dirsrv.target
```

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:



```
[root@server ~]# 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.

```
[jsmith@server ~]$ 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.

## 30.4. 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 <sup>[10]</sup>, on a different machine from the existing LDAP directory.
2. Disable the compat plug-in.

```
[root@server ~]# ipa-compat-manage disable
```

3. Restart the IdM Directory Server instance.

```
[root@server ~]# systemctl restart dirsrv.target
```

4. Enable the IdM server to allow migration:

```
[root@server ~]# 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:

```
[root@server ]# 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 30.2, “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.

```
[root@server ]# ipa-compat-manage enable
```

7. Restart the IdM Directory Server instance.

```
[root@server ]# systemctl restart dirsrv.target
```

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 30.1.2, “Planning Password Migration”](#).

- a. Install SSSD on every client machine:

```
[root@server ]# 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.

## 30.5. Scenario 3: Migrating over SSL

Both migrating using SSSD ([Section 30.3, “Scenario 1: Using SSSD as Part of Migration”](#)) and migrating directly from LDAP ([Section 30.4, “Scenario 2: Migrating an LDAP Server Directly to Identity Management”](#)) can be done over SSL. The migration procedure itself is the same, but it requires additional configuration on the IdM server.

IdM uses the OpenLDAP client libraries to connect to the remote LDAP server. This means that the OpenLDAP configuration on the IdM server machine must have the CA certificate configuration for the *LDAP directory's* issuing CA.

1. Download the CA certificate for the CA which issued the LDAP directory's certificates. The location and methods to obtain the CA certificate depend on the CA which issued it or the location of the certificate in the LDAP configuration.

Save the CA certificate as **/etc/ipa/remote.crt** on the IdM system.

2. Update the SELinux labels for the CA certificate file. The label should be **unconfined\_u:object\_r:etc\_t:s0**.

```
[root@server ~]# restorecon /etc/ipa/remote.crt
```

3. Configure the OpenLDAP libraries to use the CA certificate for the old LDAP instance.

- a. Open the OpenLDAP configuration file.

```
[root@server ~]# vim /etc/openldap/ldap.conf
```

- b. The CA certificate needs to be imported into the certificate configuration. There are three ways that this can be done:

- ✧ The **TLS\_CACERT** parameter can be set to the PEM file (**remote.crt**) for the CA of the remote LDAP server.

```
TLS_CACERT=/etc/ipa/remote.crt
```

- ✧ The CA certificate can be loaded into the IdM NSS database, and that can then be referenced in the **TLS\_CACERTDIR** parameter.

```
[root@server ~]# certutil -A -d /etc/dirsrv/slapd-EXAMPLE-COM -n "CA certificate" -t "CT,," -a -i /etc/ipa/remote.crt
[root@server ~]# vim /etc/openldap/ldap.conf

....
TLS_CACERTDIR=/etc/dirsrv/slapd-EXAMPLE-COM
```

- ✧ The CA certificate can be in any directory on the system, and that location can be given in the **TLS\_CACERTDIR** parameter.

```
[root@server ~]# vim /etc/openldap/ldap.conf

....
TLS_CACERTDIR=/etc/ipa/
```

**Only one of those configuration settings is required.**

- c. Restart the IdM Apache instance. The SSL configuration is loaded through the Apache server.

```
[root@server ~]# systemctl restart httpd.service
```

- d. Go through any required migration preparation and run the **ipa migrate-ds** script, as described in [Section 30.3, “Scenario 1: Using SSSD as Part of Migration”](#) and [Section 30.4, “Scenario 2: Migrating an LDAP Server Directly to Identity Management”](#).
- e. Undo any changes that were made to the **ldap.conf** file in step [b](#). This can prevent future problems with trusting the IdM CA or other certificate-related conflicts.
- f. Restart the IdM Apache instance to load the update SSL configuration.

```
[root@server ~]# systemctl restart httpd.service
```

---

[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.

## Appendix A. 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 25.4, “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-pkcs11` 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-pkcs11` 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-pkcs11` 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/2015:22:29:03][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 2.4.2, "Host Name and DNS Configuration"](#).

## 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 25.4, "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 15.8, “Managing Reverse DNS Zones”](#). 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=negotiateauth: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.

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>	<p>There are no Kerberos tickets. Run <b>kinit</b>.</p>

Error Log Message	Description and Fix
<pre>-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken() -1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure Server not found in Kerberos database</pre>	<p>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 <b>[domain_realm]</b> section in the <b>/etc/krb5.conf</b> file. Make sure that the IdM Kerberos domain entry is correct and matches the configuration in the Firefox negotiation parameters. For example:</p> <pre>.example.com = EXAMPLE.COM example.com = EXAMPLE.COM</pre>
<p>Nothing is in the log file.</p>	<p>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.</p>

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 [WRFILe:/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 fail (for example, **getent passwd admin**).

Authentication in IdM is managed with the SSSD daemon, which is described in the *System-Level Authentication 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 25.4, “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 "SSSD and Identity Providers \(Domains\)" section of the System-Level Authentication 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.

## Index

### A

#### attributes

- setting multi-valued, [From the Command Line](#)

### B

#### bind

- DNS and LDAP, [BIND in Identity Management](#)

### C

#### certificates

- automatically renewed, [Renewal Messages](#)

#### 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 and Removing Master DNS Zones](#)
- disabling zones, [Disabling and Enabling Zones](#)
- dynamic updates, [Enabling Dynamic DNS Updates](#)
- hosts with DHCP, [Adding Host Entries from the Command Line](#)
- PTR synchronization
  - requirements, [Synchronizing A/AAAA and PTR Records](#)

### DNS zone records, [Adding Records to DNS Zones](#)

- deleting, [Deleting Records from DNS Zones](#)
- format for adding, [Adding Records to DNS Zones](#)
- IPv4 example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- IPv6 example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- PTR example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- SRV example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- types of records, [Adding Records to DNS Zones](#)

## G

### glue entries, [Solving Orphan Entry Conflicts](#)

## H

#### hosts

- certificate not found errors, [Serial Numbers Not Found Errors](#)
- 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

### log rotation

- policies, [IdM Domain Services and Log Rotation](#)

### logging in

- SELinux problems, [SELinux Login Problems](#)
- separate credentials cache, [Caching User Kerberos Tickets](#)

## **logrotate, [IdM Domain Services and Log Rotation](#)**

## **N**

### **naming conflicts**

- in replication, [Solving Naming Conflicts](#)

## **P**

### **password expiration, [Managing Password Expiration Limits](#)**

#### **password policies**

- expiration, [Managing Password Expiration Limits](#)

#### **policies**

- log rotation, [IdM Domain Services and Log Rotation](#)

#### **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 A/AAAA and PTR Records](#)

## **R**

### **reboot, [Starting and Stopping the IdM Domain](#)**

#### **replicas**

- number in replication, [IdM Servers and Replicas](#)

#### **replication**

- errors, [Serial Numbers Not Found Errors](#)
- size limits, [IdM Servers and Replicas](#)

## **S**

### **SELinux**

- login problems, [SELinux Login Problems](#)

### **servers**

- number in replication, [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 systemctl, [Starting and Stopping the IdM Domain](#)**

**systemctl, [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, [Adding Records to DNS Zones](#)
- IPv4 example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- IPv6 example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- PTR example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- SRV example, [Examples of Adding or Modifying DNS Resource Records from the Command Line](#)
- types, [Adding Records to DNS Zones](#)



## Appendix B. Revision History

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Enterprise Linux.

<b>Revision 7.0-12</b>	<b>Fri Nov 13 2015</b>	<b>Aneta Petrová</b>
Version for 7.2 GA release with updates to DNS and other sections.		
<b>Revision 7.0-11</b>	<b>Thu Nov 12 2015</b>	<b>Aneta Petrová</b>
Version for 7.2 GA release.		
<b>Revision 7.0-10</b>	<b>Fri Mar 13 2015</b>	<b>Tomáš Čapek</b>
Async update with last-minute edits for 7.1.		
<b>Revision 7.0-8</b>	<b>Wed Feb 25 2015</b>	<b>Tomáš Čapek</b>
Version for 7.1 GA release.		
<b>Revision 7.0-6</b>	<b>Fri Dec 05 2014</b>	<b>Tomáš Čapek</b>
Rebuild to update the sort order on the splash page.		
<b>Revision 7.0-4</b>	<b>Wed Jun 11 2014</b>	<b>Ella Deon Ballard</b>
Initial release.		