



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

Managing Identity and Authorization Policies for Linux-Based
Infrastructures

Aneta Petrová
Ella Deon Ballard

Marc Muehlfeld

Tomáš Čapek

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

Managing Identity and Authorization Policies for Linux-Based Infrastructures

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

Marc Muehlfeld
Red Hat Customer Content Services
mmuehlfeld@redhat.com

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

Legal Notice

Copyright © 2016 Red Hat.

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

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

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, 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. 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: The System-Level Authentication

Guide documents different applications and services available to configure authentication on local systems, including the authconfig utility, the System Security Services Daemon (SSSD) service, the Pluggable Authentication Module (PAM) framework, Kerberos, the certmonger utility, and single-sign on (SSO) for applications. 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, using SSSD to access a Common Internet File System (CIFS), and the realmd system.

Table of Contents

Chapter 1. Introduction to Red Hat Identity Management	6
1.1. The Goal of Red Hat Identity Management	6
1.2. The Identity Management Domain	8
Part I. Installing Identity Management Servers and Services	13
Chapter 2. Installing and Uninstalling an Identity Management Server	14
2.1. Prerequisites for Installing a Server	14
2.2. Packages Required to Install an IdM Server	20
2.3. Installing an IdM Server	20
2.4. Uninstalling an IdM Server	35
2.5. Renaming an IdM Server	35
Chapter 3. Setting up IdM Replicas	36
3.1. Explaining IdM Replicas	36
3.2. Recommendations for Planning the Replica Topologies	37
3.3. Prerequisites for Installing a Replica Server	38
3.4. Creating the Replica	39
3.5. Adding Additional Replication Agreements	46
3.6. Uninstalling an IdM Replica	46
3.7. Renaming an IdM Replica	46
3.8. Changing Load Balancing for IdM Servers and Replicas	46
Chapter 4. Setting up Systems as IdM Clients	48
4.1. What Happens in Client Setup	48
4.2. Prerequisites for Installing a Client	49
4.3. Configuring a Linux System as an IdM Client	49
4.4. Manually Configuring a Linux Client	53
4.5. Setting up a Linux Client Through Kickstart	61
4.6. Re-enrolling a Host	62
4.7. Renaming Machines and Reconfiguring IdM Client Configuration	63
4.8. Performing a Two-Administrator Enrollment	64
4.9. Removing Clients from the Domain	65
4.10. Manually Unconfiguring Client Machines	65
Chapter 5. Upgrading Identity Management	67
5.1. Migrating the IdM Server to Red Hat Enterprise Linux 7	68
Chapter 6. The Basics of Managing the IdM Server and Services	76
6.1. Starting and Stopping the IdM Server	76
6.2. Logging into IdM Using Kerberos	76
6.3. The IdM Command-Line Utilities	78
6.4. The IdM Web UI	80
Chapter 7. Backing Up and Restoring Identity Management	86
7.1. Full-Server Backup and Data-Only Backup	87
7.2. Restoring a Backup	91
Part II. Managing User and System Identities in a Linux Domain	93
Chapter 8. Managing User Accounts	94
8.1. Setting up User Home Directories	94
8.2. User Lifecycle	95
8.3. Editing Users	105

8.4. Enabling and Disabling User Accounts	107
8.5. Allowing Non-admin Users to Manage User Entries	108
8.6. Using an External Provisioning System for Users and Groups	112
Chapter 9. User Authentication	120
9.1. Managing Public SSH Keys for Users	120
9.2. Defining User Authentication Methods	123
9.3. Changing and Resetting User Passwords	125
9.4. Unlocking User Accounts After Password Failures	127
9.5. One-Time Passwords	128
9.6. Smart Cards	134
9.7. User Certificates	136
Chapter 10. User Groups	137
10.1. Managing User Private Groups	137
10.2. Managing User Groups	138
Chapter 11. Unique UID and GID Number Assignments	157
11.1. ID Ranges	157
11.2. ID Range Assignments During Installation	157
11.3. Displaying Currently Assigned ID Ranges	158
11.4. Automatic ID Range Extension After Deleting a Replica	158
11.5. Manual ID Range Extension and Assigning a New ID Range	159
11.6. Ensuring That ID Values Are Unique	160
11.7. Repairing Changed UID and GID Numbers	160
Chapter 12. User and Group Schema	162
12.1. About Changing the Default User and Group Schema	163
12.2. Applying Custom Object Classes to New User Entries	164
12.3. Applying Custom Object Classes to New Group Entries	165
12.4. Specifying Default User and Group Attributes	167
Chapter 13. ID Views	172
13.1. User Overrides and Group Overrides	172
13.2. ID Views and SSSD	172
13.3. Managing ID Views from the Web UI	173
13.4. Managing ID Views from the command line	178
Chapter 14. Managing Hosts	180
14.1. About Hosts, Services, and Machine Identity and Authentication	180
14.2. About Host Entry Configuration Properties	181
14.3. Disabling and Re-enabling Host Entries	182
14.4. Managing Public SSH Keys for Hosts	183
14.5. Setting Ethers Information for a Host	190
14.6. Managing Host Groups	190
Chapter 15. Managing Services	194
15.1. Adding and Editing Service Entries and Keytabs	194
15.2. Configuring Clustered Services	196
15.3. Using the Same Service Principal for Multiple Services	197
15.4. Retrieve Existing Keytabs for Multiple Servers	197
15.5. Disabling and Re-enabling Service Entries	199
Chapter 16. Delegating User Access to Hosts and Services	200
16.1. Delegating Service Management	200
16.2. Delegating Host Management	201

16.3. Delegating Host or Service Management in the Web UI	202
16.4. Accessing Delegated Services	203
Chapter 17. Managing Certificates for Users, Hosts, and Services	204
17.1. Managing Certificates with the Integrated IdM CA	204
17.2. Managing Certificates Issued by External CAs	208
17.3. Listing and Displaying Certificates	209
17.4. Certificate Profiles	211
17.5. Certificate Authority ACL Rules	216
17.6. Using Certificate Profiles and ACLs to Issue User Certificates with the IdM CA	222
Chapter 18. Storing Authentication Secrets with Vaults	228
18.1. How Vaults Work	228
18.2. Prerequisites for Using Vaults	230
18.3. Getting Help for Vault Commands	230
18.4. Storing a User's Personal Secret	231
18.5. Storing a Service Secret in a Vault	232
18.6. Storing a Common Secret for Multiple Users	236
Chapter 19. Integrating with NIS Domains and Netgroups	238
19.1. About NIS and Identity Management	238
19.2. Setting the NIS Port for Identity Management	239
19.3. Creating Netgroups	240
19.4. Exposing Automount Maps to NIS Clients	245
19.5. Migrating from NIS to IdM	246
Chapter 20. Managing DNS	253
20.1. Installing DNS Services Into an Existing Server	253
20.2. BIND in Identity Management	253
20.3. Supported DNS Zone Types	254
20.4. DNS Configuration Priorities	255
20.5. Managing Master DNS Zones	256
20.6. Managing Dynamic DNS Updates	269
20.7. Managing DNS Forwarding	276
20.8. Managing Reverse DNS Zones	283
20.9. Defining DNS Query Policy	285
Part III. Defining Domain-wide System Policies	287
Chapter 21. Using Automount	288
21.1. About Automount and IdM	288
21.2. Configuring Automount	289
21.3. Setting up a Kerberized NFS Server	294
21.4. Configuring Locations	297
21.5. Configuring Maps	299
Chapter 22. Defining Password Policies	306
22.1. About Password Policies and Policy Attributes	306
22.2. Viewing Password Policies	308
22.3. Creating and Editing Password Policies	314
22.4. Managing Password Expiration Limits	317
22.5. Changing the Priority of Group Password Policies	318
22.6. Setting Account Lockout Policies	318
22.7. Enabling a Password Change Dialog	321
Chapter 23. Managing the Kerberos Domain	323

Chapter 23. Managing the Kerberos Domain	322
23.1. About Kerberos	322
23.2. Setting Kerberos Ticket Policies	323
23.3. Refreshing Kerberos Tickets	325
23.4. Kerberos Flags for Services and Hosts	327
23.5. Caching Kerberos Passwords	329
23.6. Removing Keytabs	330
Chapter 24. Using sudo	331
24.1. The sudo Utility in Identity Management	331
24.2. sudo Rules in Identity Management	331
24.3. Configuring the Location for Looking up sudo Policies	332
24.4. Adding sudo Commands, Command Groups, and Rules	334
24.5. Modifying sudo Commands and Command Groups	338
24.6. Modifying sudo Rules	338
24.7. Listing and Displaying sudo Commands, Command Groups, and Rules	350
24.8. Disabling and Enabling sudo Rules	350
24.9. Removing sudo Commands, Command Groups, and Rules	351
Chapter 25. Configuring Host-Based Access Control	353
25.1. About Host-Based Access Control	353
25.2. Creating Host-Based Access Control Entries for Services and Service Groups	354
25.3. Defining Host-Based Access Control Rules	359
25.4. Testing Host-Based Access Control Rules	366
Chapter 26. Defining SELinux User Maps	371
26.1. About Identity Management, SELinux, and Mapping Users	371
26.2. Configuring SELinux User Map Order and Defaults	373
26.3. Mapping SELinux Users and IdM Users	376
Chapter 27. Defining Automatic Group Membership for Users and Hosts	382
27.1. About Automembership	382
27.2. Defining Automembership Rules (Basic Procedure)	383
27.3. Examples of Using Automember Groups	386
Part IV. Configuring the Identity Management Server	389
Chapter 28. Defining Access Control for IdM Users	390
28.1. Access Controls for IdM Entries	390
28.2. Defining Self-Service Settings	391
28.3. Delegating Permissions over Users	395
28.4. Defining Role-Based Access Controls	397
Chapter 29. Identity Management Files and Logs	414
29.1. A Reference of IdM Server Configuration Files and Directories	414
29.2. IdM Domain Services and Log Rotation	416
29.3. About default.conf and Context Configuration Files	417
29.4. Checking IdM Server Logs	418
Chapter 30. Managing Certificates and Certificate Authorities	425
30.1. Renewal Messages	425
30.2. Automatic CA Certificate Renewal	425
30.3. Manual CA Certificate Renewal	425
30.4. Manual CA Certificate Installation	426
30.5. Changing Certificate Chaining	427
30.6. Starting IdM with Expired Certificates	427

30.7. Configuring Alternate Certificate Authorities	428
30.8. Promoting a Replica to a Master CA Server	429
30.9. Configuring OCSP Responders	432
Chapter 31. Disabling Anonymous Binds	434
Chapter 32. Managing Replicas and Replication Agreements	435
32.1. Explaining Replication Agreements	435
32.2. Listing Replication Agreements	435
32.3. Creating and Removing Replication Agreements	436
32.4. Initiating a Manual Replication Update	437
32.5. Re-initializing a Replica	437
32.6. Removing a Replica	437
32.7. Renaming a Server Host System	438
Chapter 33. Migrating from an LDAP Directory to IdM	439
33.1. An Overview of LDAP to IdM Migration	439
33.2. Examples for Using migrate-ds	447
33.3. Scenario 1: Using SSSD as Part of Migration	450
33.4. Scenario 2: Migrating an LDAP Server Directly to Identity Management	451
33.5. Scenario 3: Migrating over SSL	453
Appendix A. Troubleshooting Identity Management	455
A.1. Identity Management Servers	455
A.2. Identity Management Replicas	456
A.3. Identity Management Clients	460
A.4. Logging In and Authentication Problems	461
Appendix B. Revision History	464

Chapter 1. Introduction to Red Hat Identity Management

This chapter explains the purpose of Red Hat Identity Management. It also provides basic information about the Identity Management domain, including the client and server machines that are part of the domain.

1.1. The Goal of Red Hat Identity Management

Red Hat Identity Management (IdM) provides a centralized and unified way to manage identity stores as well as authentication and authorization policies in a Linux-based domain. IdM significantly reduces the administrative overhead of managing different services individually and having to use different tools on different machines.

IdM is one of the few centralized identity, policy, and authorization software solutions that support:

- » advanced features of Linux operating system environments
- » unifying large groups of Linux machines
- » native integration with Active Directory

IdM creates a Linux-based and Linux-controlled domain:

- » IdM builds on existing, native Linux tools and protocols. While it has its own processes and configuration, its underlying technologies are familiar and trusted by Linux administrators and are well-established on Linux systems.
- » IdM servers and clients are Red Hat Enterprise Linux machines. However, even though IdM does not support Windows clients directly, it allows integration with Active Directory environment.

Note

This guide describes using IdM in Linux environments only. For more information on integration with Active Directory, see the [Windows Integration Guide](#).

For information on the Samba suite, which allows integrating Linux machines into Active Directory environment, see the [Using Samba, Kerberos, and Winbind](#) chapter in the *Windows Integration Guide*.

1.1.1. Examples of Benefits Brought by IdM

Managing identities and policies with several Linux servers

Without IdM: Each server is administered separately. All passwords are saved on the local machines. The IT administrator manages users on every machine, sets authentication and authorization policies separately, and maintains local passwords.

With IdM: The IT administrator is able to:

- » maintain the identities in one central place: the IdM server
- » apply policies uniformly to multiples of machines at the same time
- » set different access levels for users by using host-based access control, delegation, and other rules
- » centrally manage privilege escalation rules
- » define how home directories are mounted

Enterprise single sign-on

Without IdM: Users log in to the system and are prompted for password every single time they access a service or application. These passwords might be different, and the users have to remember which credential to use for which application.

With IdM: After users log in to the system, they can access multiple services and applications without being repeatedly asked for their credentials. This helps:

- » improve usability
- » reduce the security risk of passwords put on sticky notes
- » boost user productivity

Managing a mixed Linux and Windows environment

Without IdM: Windows systems are managed in an Active Directory forest. However, development, production, and other teams have many Linux systems, which are excluded from the Active Directory environment.

With IdM: The IT administrator is able to:

- » manage the Linux systems using native Linux tools
- » integrate the Linux systems with the Windows systems, thus preserving a centralized user store
- » expand the Linux base easily
- » separate management of Linux and Active Directory machines and allow Linux and Windows admins to control their environment directly

1.1.2. Contrasting Identity Management with a Standard LDAP Directory

A standard LDAP directory, such as Red Hat Directory Server, is a general-purpose directory: it can be customized to fit a broad range of use cases.

- » Schema: a flexible schema that can be customized for a vast array of entries, such as users, machines, network entities, physical equipment, or buildings.
- » Typically used as: a back-end directory to store data for other applications, such as business applications that provide services on the Internet.

Identity Management (IdM) has a specific purpose: managing identities as well as authentication and authorization policies that relate to these identities.

- » Schema: a specific schema that defines a particular set of entries relevant to its purpose, such as entries for user or machine identities.
- » Typically used as: the identity and authentication server to manage identities within the boundaries of an enterprise or a project.

The underlying directory server technology is the same for both Red Hat Directory Server and IdM. However, IdM is optimized to manage identities. This limits its general extensibility, but also brings certain benefits: simpler configuration, better automation of resource management, and increased efficiency in managing identities.

1.2. The Identity Management Domain

The Identity Management (IdM) domain consists of a group of machines that share the same configuration, policies, and identity stores. The shared properties allow the machines within the domain to be aware of each other and operate together.

From the perspective of IdM, the domain includes the following types of machines:

- » IdM servers, which work as domain controllers
- » IdM clients, which are enrolled with the servers

Additionally, IdM servers are also IdM clients enrolled with themselves: server machines provide the same functionality as clients.

IdM supports Red Hat Enterprise Linux machines as the IdM servers and clients.



Note

This guide describes using IdM in Linux environments. For more information on integration with Active Directory, see the [Windows Integration Guide](#).

1.2.1. Identity Management Servers

The IdM servers act as central repositories for identity and policy information. They also host the services used by domain members. IdM provides a set of management tools to manage all the IdM-associated services centrally: the IdM web UI and command-line utilities.

For information on installing IdM servers, see [Chapter 2, Installing and Uninstalling an Identity Management Server](#).

To support redundancy and load-balancing, the data and configuration can be replicated from one IdM server to another: a *replica* of the initial server. You can configure servers and their replicas to provide different services to clients. For more details on IdM replicas, see [Chapter 3, Setting up IdM Replicas](#).

1.2.1.1. Services Hosted by IdM Servers

Most of the following services are not strictly required to be installed on the IdM server. For example, services such as a certificate authority (CA), a DNS server, or a Network Time Protocol (NTP) server can be installed on an external server outside the IdM domain.

Kerberos KDC

IdM uses the Kerberos protocol to support single sign-on. With Kerberos, the user only needs to present the correct user name and password once. Then the user can access IdM services without the system prompting for the credentials again.

- For details on how Kerberos works, see the [System-Level Authentication Guide](#).
- For information on how to authenticate using Kerberos in IdM, see [Section 6.2, “Logging into IdM Using Kerberos”](#).
- For information on managing Kerberos in IdM, see [Chapter 23, Managing the Kerberos Domain](#).

LDAP directory server

IdM includes an internal LDAP directory server instance where it stores all the IdM information, such as information related to Kerberos, user accounts, host entries, services, policies, DNS, and others.

The LDAP directory server instance is based on the same technology as Red Hat Directory Server. However, it is tuned to IdM-specific tasks.

Note

This guide refers to this component as Directory Server.

Certificate authority

In most deployments, an integrated certificate authority (CA) is installed with the IdM server. You can also install the server without the integrated CA, as long as you create and provide all required certificates independently.

- For more details on installing an IdM server with the different CA configurations, see [Section 2.3.2, “Determining What CA Configuration to Use”](#).

Note

This guide refers to this component as Certificate System when addressing the implementation and as certificate authority when addressing the services provided by the implementation.

Domain Name System (DNS)

IdM uses 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](#).
- For information on using DNS with IdM and important prerequisites, see [Section 2.1.3, “Host Name and DNS Configuration”](#).

- For details on installing an IdM server with or without integrated DNS, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#).

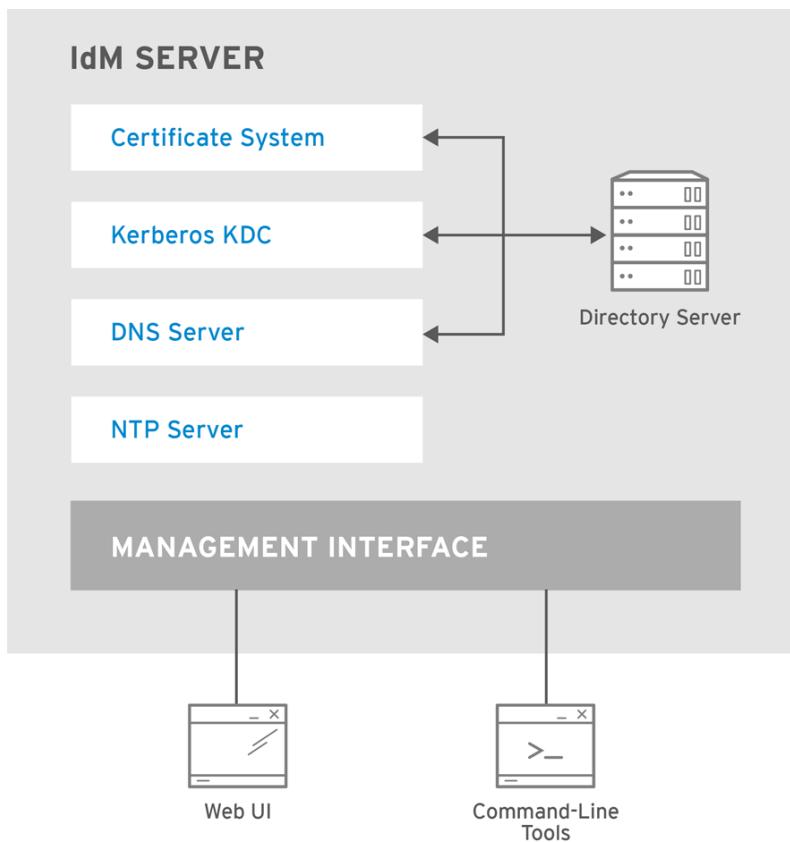
Network Time Protocol

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 and to prevent replay attacks. If the times between the server and client skew outside the allowed range, the Kerberos tickets are invalidated.

By default, IdM uses the Network Time Protocol (NTP) to synchronize clocks over a network. With NTP, a central server acts as an authoritative clock and the clients synchronize their times to match the server clock. The IdM server is configured as the NTP server for the IdM domain during the server installation process.

Note

Running an NTP server on an IdM server installed on a virtual machine can lead to inaccurate time synchronization in some environments. To avoid potential problems, do not run NTP on IdM servers installed on virtual machines. For more information on the reliability of an NTP server on a virtual machine, see [this Knowledgebase solution](#).



RHEL_404973_0516

Figure 1.1. The Identity Management Server: Unifying Services

1.2.2. Identity Management Clients

IdM clients are machines configured to operate within the IdM domain. They interact with the IdM servers to access domain resources. For example, they belong to the Kerberos domains configured on the servers, receive certificates and tickets issued by the servers, and use other centralized services for authentication and authorization.

An IdM client does not require dedicated client software to interact as a part of the domain. It only requires proper system configuration of certain services and libraries, such as Kerberos or DNS. This configuration directs the client machine to use IdM services.

For information on installing IdM clients, see [Chapter 4, Setting up Systems as IdM Clients](#).

1.2.2.1. Services Hosted by IdM Clients

System Security Services Daemon

The System Security Services Daemon (SSSD) is a client-side application for caching credentials. Using SSSD on client machines is recommended because it simplifies the required client configuration. SSSD also provides additional features, for example:

- ✖ offline client authentication, ensured by caching credentials from centralized identity and authentication stores locally
- ✖ improved consistency of the authentication process, because it is not necessary to maintain both a central account and a local user account for offline authentication
- ✖ integration with other services, such as `sudo`
- ✖ host-based access control (HBAC) authorization

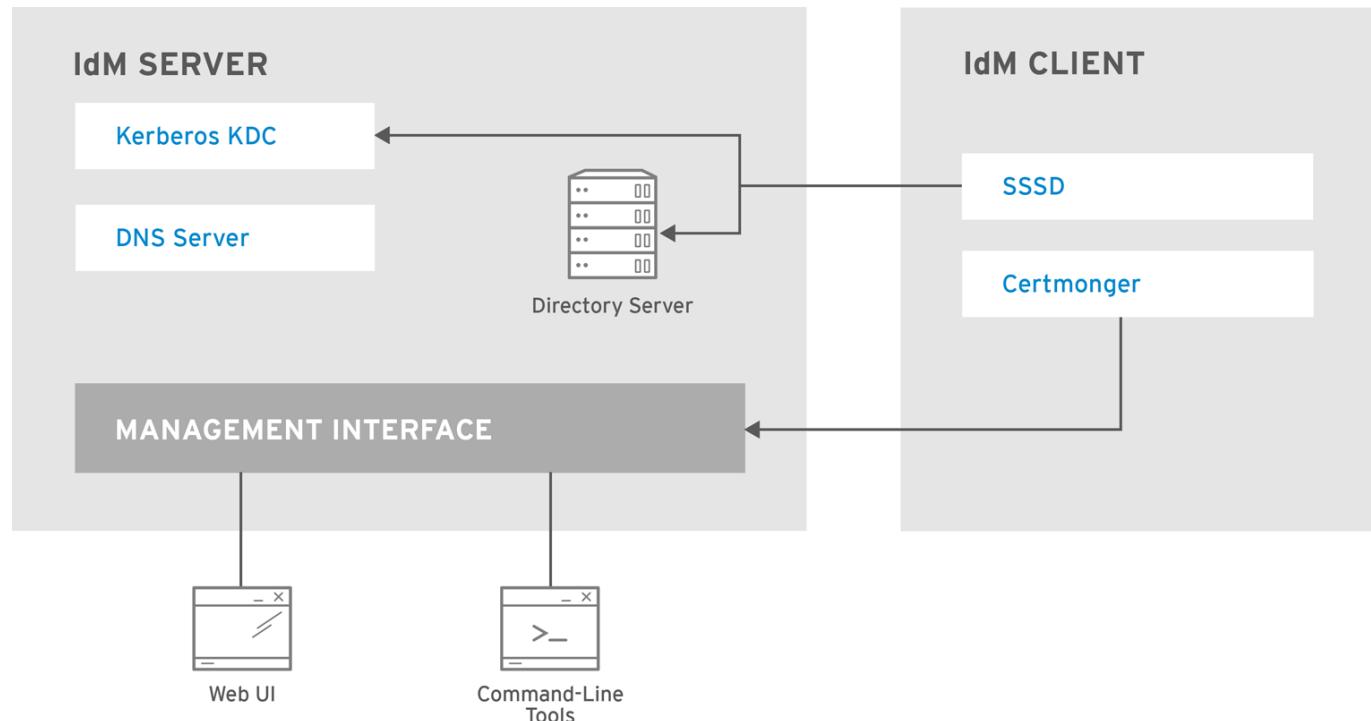
With SSSD, the IdM administrators can define all identity configuration centrally in the IdM server, while caching allows the local system to continue normal authentication operations if the IdM server becomes unavailable or if the client becomes offline.

For more information about SSSD, see the [System-Level Authentication Guide](#). SSSD also supports Windows Active Directory (AD). For more information about using SSSD with AD, see the [Windows Integration Guide](#).

`certmonger`

The `certmonger` service monitors and renews the certificates on the client. It can request new certificates for the services on the system.

For more information about `certmonger`, see the [System-Level Authentication Guide](#).



RHEL_404973_0516

Figure 1.2. Interactions Between IdM Services

Part I. Installing Identity Management Servers and Services

Chapter 2. Installing and Uninstalling an Identity Management Server

An *Identity Management* (IdM) server is a domain controller: it defines and manages the IdM domain. To set up an IdM server, you must:

1. Install the necessary packages
2. Configure the machine using setup scripts

You can set up multiple domain controllers within your domain for load-balancing and failover tolerance. These additional servers are *replicas* of the initial master IdM server. This chapter describes installing an IdM server. For information on installing a replica, see [Chapter 3, Setting up IdM Replicas](#).

2.1. Prerequisites for Installing a Server



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.

2.1.1. Hardware Recommendations

RAM is the most important hardware feature to size properly. To determine how much RAM you require, consider these recommendations:

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



Note

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

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

To increase performance, you can tune the underlying Directory Server to increase performance. For details, see [“Optimizing System Performance”](#) in the Directory Server Performance Tuning Guide.

2.1.2. System Requirements

Identity Management 4.2 is supported on Red Hat Enterprise Linux 7. Install an IdM server on a clean system without any custom configuration for services such as DNS, Kerberos, or Directory Server.

The following limitations also apply to installing an IdM server:

- ▶ Installing and running IdM in the Federal Information Processing Standard (FIPS) mode is not supported. Disable FIPS on your system before installing an IdM server, replica, or client, and do not enable it after the installation.
- ▶ Red Hat recommends to disable the Name Service Cache Daemon (NSCD) on Identity Management machines. 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 conflicts between NSCD and SSSD.

The IdM server installation overwrites system files to set up the IdM domain. IdM backs up the original system files to `/var/lib/ipa/sysrestore/`.

2.1.3. Host Name and DNS Configuration



Warning

Be extremely cautious and ensure that:

- ▶ you have a tested and functional DNS service available
- ▶ the service is properly configured

This requirement applies to IdM servers with integrated DNS services as well as to IdM servers installed without DNS. DNS records are vital for nearly all IdM domain functions, including running LDAP directory services, Kerberos, and Active Directory integration. Note that the primary DNS domain and Kerberos realm cannot be changed after the installation.

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 the DNS level, the *primary DNS domain used for IdM* cannot be shared with any other system.

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

Verifying the Server Host Name

The host name must be a fully-qualified domain name, such as `server.example.com`. To verify your machine's host name, use the `hostname` utility:

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

The output of `hostname` must not be `localhost` or `localhost6`.



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 must not resolve to the loopback address. It must resolve to the machine's public IP address, not to `127.0.0.1`.

Verifying the Forward and Reverse DNS Configuration

- 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
```

- Verify the forward DNS configuration by using the `dig` utility and adding the host name.
 - 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 server
2001:DB8::1111 server.example.com server
```

2.1.4. Port Requirements

IdM uses a number of ports to communicate with its services. These ports, listed in [Table 2.1, “Identity Management 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. Identity Management Ports

Service	Ports	Protocol
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 ports 80 and 389.

- » Port 80 (HTTP) is used to provide Online Certificate Status Protocol (OCSP) responses and Certificate Revocation Lists (CRL). Both are digitally signed and therefore secured against man-in-the-middle attacks.
- » Port 389 (LDAP) uses STARTTLS and GSSAPI for encryption.

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.

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.

2.2. Packages Required to Install an IdM Server

To install the packages required for a server without integrated DNS services:

```
# yum install ipa-server
```

To install the packages required for a server with integrated DNS services:

```
# yum install ipa-server ipa-server-dns
```

Note

To determine whether DNS is right for your use case, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#).

The *ipa-server* packages automatically installs other required packages as dependencies, such as:

- » *389-ds-base* for the Directory Server LDAP service
- » *krb5-server* package for the Kerberos service
- » various IdM-specific tools

2.3. Installing an IdM Server

Note

The installation procedures and examples in the following sections are not mutually exclusive: you can combine them to achieve the required result. For example, you can install a server with integrated DNS and with an externally-hosted root CA.

The **ipa-server-install** utility installs and configures an IdM server.

Before installing a server, see these sections:

- » [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#)

» [Section 2.3.2, “Determining What CA Configuration to Use”](#)

The **ipa-server-install** utility provides a non-interactive installation mode which allows automated and unattended server setup. For details, see [Section 2.3.7, “Installing a Server Non-Interactively”](#).

The **ipa-server-install** installation script creates a log file at **/var/log/ipaserver-install.log**. If the installation fails, the log can help you identify the problem.

2.3.1. Determining Whether to Use Integrated DNS

IdM supports installing a server with integrated DNS or without integrated DNS.

An IdM server with integrated DNS services

The integrated DNS server provided by IdM is not designed to be used as a general-purpose DNS server. It only supports features related to IdM deployment and maintenance. It does not support some of the advanced DNS features.

Red Hat strongly recommends IdM-integrated DNS for basic usage within the IdM deployment: When the IdM server also manages DNS, there is tight integration between DNS and native IdM tools which enables automating some of the DNS record management.

Note that even if an IdM server is used as a master DNS server, other external DNS servers can still be used as slave servers.

For example, if your environment is already using another DNS server, such as an Active Directory-integrated DNS server, you can delegate only the IdM primary domain to the IdM-integrated DNS. You are not required to migrate DNS zones over to the IdM-integrated DNS.

To install a server with integrated DNS, see [Section 2.3.3, “Installing a Server with Integrated DNS”](#).

An IdM server without integrated DNS services

An external DNS server is used to provide the DNS services. Consider installing an IdM server without DNS in these situations:

- » If you require advanced DNS features beyond the scope of the IdM DNS
- » In environments with a well-established DNS infrastructure which allows you to use external DNS servers

To install a server without integrated DNS, see [Section 2.3.4, “Installing a Server Without Integrated DNS”](#).



Important

Make sure your system meets the DNS requirements described in [Section 2.1.3, “Host Name and DNS Configuration”](#).

Maintenance Requirements for Integrated or External DNS

When using an integrated DNS server, most of the DNS record maintenance is automated. You only must:

- » 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 following command:

```
# dig @IP_address +norecurse +short ipa.example.com. NS
```

`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.

When using an external DNS server, you must:

- » manually create the new domain on the DNS server
- » fill the new domain manually with records from the zone file that is generated by the IdM installer
- » manually update the records 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

Preventing DNS Amplification Attacks

The default configuration of the IdM-integrated DNS server allows all clients to issue recursive queries to the DNS server. If your server is deployed in a network with an untrusted client, change the server's configuration to limit recursion to authorized clients only. [1]

To ensure that only authorized clients are allowed to issue recursive queries, add the appropriate access control list (ACL) statements to the `/etc/named.conf` file on your server. For example:

```
acl authorized { 192.0.2.0/24; 198.51.100.0/24; };
options {
    allow-query { any; };
    allow-recursion { authorized; };
};
```

2.3.2. Determining What CA Configuration to Use

IdM supports installing a server with an integrated Red Hat Certificate System certificate authority (CA) or without a CA.

Server with an integrated Certificate System CA

This is the default configuration suitable for most deployments. Certificate System uses a *CA signing certificate* to create and sign the certificates in the IdM domain.



Warning

Red Hat strongly recommends to install the CA on more than one server in the IdM domain. For information on installing a replica of the initial server including the CA services, see [Section 3.4.3, “Installing a replica from a server with a Certificate System CA installed”](#).

If you install the CA on only one server, you risk losing the CA configuration without a chance of recovery if the server fails. See [Section A.2.5, “Recovering a Lost CA Server”](#) for details.

The CA signing certificate must be signed by a *root CA*, which is the highest CA in the CA hierarchy. The root CA can be the Certificate System CA itself or an externally-hosted CA.

The Certificate System CA is the root CA

This is the default configuration.

To install a server with this configuration, see [Section 2.3.3, “Installing a Server with Integrated DNS”](#) and [Section 2.3.4, “Installing a Server Without Integrated DNS”](#).

An external CA is the root CA

The Certificate System CA is subordinate to an external CA. However, all certificates for the IdM domain are still issued by the Certificate System instance.

The external CA can be a corporate CA or a third-party CA, such as Verisign or Thawte. The certificates issued within the IdM domain are potentially subject to restrictions set by the external root CA for attributes like the validity period.

To install a server with an externally-hosted root CA, see [Section 2.3.5, “Installing a Server with an External CA as the Root CA”](#).

Server without a CA

This configuration option is suitable for very rare cases when restrictions within the infrastructure do not allow to install certificate services with the server.

You must request these certificates from a third-party authority prior to the installation:

- ✖ 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

Managing certificates without the integrated Certificate System CA presents a significant maintenance burden. Most notably:

- Creating, uploading, and renewing certificates is a manual process.
- The **certmonger** service is not used to track certificates. Therefore, it does not warn you of impending certificate expiration.

To install a server without an integrated CA, see [Section 2.3.6, “Installing Without a CA”](#)

2.3.3. Installing a Server with Integrated DNS

Note

If you are unsure what DNS or CA configuration is appropriate for you, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) and [Section 2.3.2, “Determining What CA Configuration to Use”](#).

To install a server with integrated DNS, you must provide the following information during the installation process:

DNS forwarders

The following DNS forwarder settings are supported:

- one or more forwarders (the **--forwarder** option in non-interactive installation mode)
- no forwarders (the **--no-forwarders** option in non-interactive installation mode)

If you are unsure whether to use DNS forwarding, see [Section 20.7, “Managing DNS Forwarding”](#).

Reverse DNS zones

The following reverse DNS zone settings are supported:

- a default value for the reverse DNS zone (the default automatic behavior, no command-line option required)
- no reverse DNS zone (the **--no-reverse** option in non-interactive installation mode)

For non-interactive installation, add the **--setup-dns** option as well.

Example 2.1. Installing a Server with Integrated DNS

This procedure installs a server:

- with integrated DNS
- with integrated Certificate System CA as the root CA, which is the default CA configuration

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

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

2. The script prompts to configure an integrated DNS service. Enter **yes**.

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

3. Allow the installation process to overwrite existing BIND configuration.

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

4. The script prompts for several required settings.

- » To accept the default values in brackets, press **Enter**.
- » To provide a value different than the proposed default value, enter the required value.

```
Server host name [server.example.com]:  
Please confirm the domain name [example.com]:  
Please provide a realm name [EXAMPLE.COM]:
```



Warning

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 the primary DNS domain is **ipa.example.com**, use **IPA.EXAMPLE.COM** for the Kerberos realm name.

Different naming practices will prevent you from using Active Directory trusts and can have other negative consequences.

5. Enter the passwords for the Directory Server superuser, **cn=Directory Manager**, and for the **admin** IdM system user account.

```
Directory Manager password:  
IPA admin password:
```

6. The script prompts for DNS forwarders.

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

- » To configure DNS forwarders, enter **yes**.

To provide the forwarders, open the **/etc/resolv.conf** file, and supply the IP addresses in the file as DNS forwarders. The installation process will add the forwarder IP addresses 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**.

7. The script 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.
```

- » If you do not want to create a reverse zone, enter **no**.

8. Enter **yes** to confirm the server configuration.

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

9. The installation script now configures the server. Wait for the operation to complete.
10. The installation script produces a DNS zone file with records: the **/tmp/sample.zone.2yv_RI.db** file in the example output below. These records must be added to the existing DNS servers.
- » If you are using an IdM-integrated DNS server, IdM updates the records automatically.
 - » If you are using an external DNS server, the process of updating the DNS records varies depending on the particular DNS solution.

```
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
```



Important

The server installation is not complete until you add the DNS records to the existing DNS servers.

11. Add DNS delegation from the parent domain to the IdM DNS domain. For example, if the IdM DNS domain is **ipa.example.com**, add a name server (NS) record to the **example.com** parent domain.



Important

This step must be repeated each time an IdM DNS server is installed.

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 2.1.4, “Port Requirements”](#).

To test the new server:

1. Authenticate to the Kerberos realm using the admin credentials. This verifies that **admin** is properly configured and the Kerberos realm is accessible.

```
# kinit admin
```

2. Run a command such as **ipa user-find**. On a new server, the command prints the only configured user: **admin**.

```
# 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
-----
```

2.3.4. Installing a Server Without Integrated DNS

Note

If you are unsure what DNS or CA configuration is appropriate for you, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) and [Section 2.3.2, “Determining What CA Configuration to Use”](#).

To install a server without integrated DNS, run the **ipa-server-install** utility without any DNS-related options.

Example 2.2. Installing a Server Without Integrated DNS

This procedure installs a server:

- » without integrated DNS
 - » with integrated Certificate System CA as the root CA, which is the default CA configuration
1. Run the **ipa-server-install** utility.

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

2. The script prompts to configure an integrated DNS service. Press **Enter** to select the default **no** option.

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

3. The script prompts for several required settings.

- » To accept the default values in brackets, press **Enter**.
- » To provide a value different than the proposed default value, enter the required value.

Server host name [server.example.com]:
 Please confirm the domain name [example.com]:
 Please provide a realm name [EXAMPLE.COM]:



Warning

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 the primary DNS domain is **ipa.example.com**, use **IPA.EXAMPLE.COM** for the Kerberos realm name.

Different naming practices will prevent you from using Active Directory trusts and can have other negative consequences.

4. Enter the passwords for the Directory Server superuser, **cn=Directory Manager**, and for the **admin** IdM system user account.

Directory Manager password:
 IPA admin password:

5. Enter **yes** to confirm the server configuration.

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

6. The installation script now configures the server. Wait for the operation to complete.
7. The installation script produces a DNS zone file with records: the **/tmp/sample.zone.2yv_RI.db** file in the example output below. These records must be added to the existing DNS servers.
- » If you are using an IdM-integrated DNS server, IdM updates the records automatically.
 - » If you are using an external DNS server, the process of updating the DNS records varies depending on the particular DNS solution.

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
```



Important

The server installation is not complete until you add the DNS records to the existing DNS servers.

8. Add DNS delegation from the parent domain to the IdM DNS domain. For example, if the IdM DNS domain is **ipa.example.com**, add a name server (NS) record to the **example.com** parent domain.



Important

This step must be repeated each time an IdM DNS server is installed.

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 2.1.4, “Port Requirements”](#).

To test the new server:

1. Authenticate to the Kerberos realm using the admin credentials. This verifies that **admin** is properly configured and the Kerberos realm is accessible.

```
# kinit admin
```

2. Run a command such as **ipa user-find**. On a new server, the command prints the only configured user: **admin**.

```
# 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
```

2.3.5. Installing a Server with an External CA as the Root CA



Note

If you are unsure what DNS or CA configuration is appropriate for you, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) and [Section 2.3.2, “Determining What CA Configuration to Use”](#).

To install a server that uses an external CA as the root CA, add the `--external-ca` option to the `ipa-server-install` utility. Other than that, most of the installation procedure is the same as in [Section 2.3.3, “Installing a Server with Integrated DNS”](#) or [Section 2.3.4, “Installing a Server Without Integrated DNS”](#).

During the configuration of the Certificate System instance, the utility will print the location of the certificate signing request (CSR): `/root/ipa.csr`:

```
...
```

```
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
```

When this happens:

1. 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.

2. 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.

- 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:

```
# ipa-server-install --external-cert-
file=/tmp/servercert20110601.pem --external-cert-
file=/tmp/cacert.pem
```

Note

The **ipa-server-install --external-ca** command 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 A.1.1, “External CA Installation Fails”](#).

2.3.6. Installing Without a CA

Note

If you are unsure what DNS or CA configuration is appropriate for you, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) and [Section 2.3.2, “Determining What CA Configuration to Use”](#).

To install a server without a CA, you must provide the required certificates manually by adding options to the **ipa-server-install** utility. Other than that, most of the installation procedure is the same as in [Section 2.3.3, “Installing a Server with Integrated DNS”](#) or [Section 2.3.4, “Installing a Server Without Integrated DNS”](#).

Note

The command-line options in this section are incompatible with the **--external-ca** option.

To provide the LDAP server certificate and private key:

- ▶ **--dirsrv-cert-file** for the certificate and private key files for the LDAP server certificate

- » **--dirsrv-pin** for the password to access the private key in the files specified in **--dirsrv-cert-file**

To provide the Apache server certificate and private key:

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

To provide the full CA certificate chain of the CA that issued the LDAP and Apache server certificates:

- » **--dirsrv-cert-file** and **--http-cert-file** for the certificate files with the full CA certificate chain or a part of it

You can add these options 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 files provided using **--dirsrv-cert-file** and **--http-cert-file** must contain exactly one server certificate and exactly one private key.



Note

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

- » If necessary, add **--ca-cert-file** for the certificate files to complete the full CA certificate chain.

You can add this option 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** 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.

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 used the **--root-ca-file** option to specify the PEM file of the root CA certificate. 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**.

2.3.7. Installing a Server Non-Interactively



Note

If you are unsure what DNS or CA configuration is appropriate for you, see [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) and [Section 2.3.2, “Determining What CA Configuration to Use”](#).

The minimum required options for a non-interactive installation are:

- » **--ds-password** to provide the password for the Directory Manager (DM), the Directory Server super user
- » **--admin-password** to provide the password for **admin**, the IdM administrator
- » **--realm** to provide the Kerberos realm name
- » **--unattended** to let the installation process select default options for the host name and domain name

Optionally, you can provide custom values for these settings:

- **--hostname** for the server host name
- **--domain** for the domain name



Warning

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 the primary DNS domain is **ipa.example.com**, use **IPA.EXAMPLE.COM** for the Kerberos realm name.

Different naming practices will prevent you from using Active Directory trusts and can have other negative consequences.

For a complete list of options accepted by **ipa-server-install**, run the **ipa-server-install --help** command.

Example 2.3. Basic Installation without Interaction

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

```
# ipa-server-install --realm EXAMPLE.COM --ds-password
DM_password --admin-password admin_password --unattended
```

2. The setup script now configures the server. Wait for the operation to complete.
3. The installation script produces a DNS zone file with records: the `/tmp/sample.zone.2yv_RI.db` file in the example output below. These records must be added to the existing DNS servers.
 - » If you are using an IdM-integrated DNS server, IdM updates the records automatically.
 - » If you are using an external DNS server, the process of updating the DNS records varies depending on the particular DNS solution.

```
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
```



Important

The server installation is not complete until you add the DNS records to the existing DNS servers.

4. Add DNS delegation from the parent domain to the IdM DNS domain. For example, if the IdM DNS domain is `ipa.example.com`, add a name server (NS) record to the `example.com` parent domain.



Important

This step must be repeated each time an IdM DNS server is installed.

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 2.1.4, “Port Requirements”](#).

To test the new server:

1. Authenticate to the Kerberos realm using the admin credentials. This verifies that `admin` is properly configured and the Kerberos realm is accessible.

```
# kinit admin
```

2. Run a command such as `ipa user-find`. On a new server, the command prints the only configured user: `admin`.

```
# 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
-----
```

2.4. Uninstalling an IdM Server

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

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

If the server included integrated DNS, update the name server (NS) records in the parent domain to ensure they do not point to the uninstalled server.



Note

The procedure for uninstalling an IdM replica is different from uninstalling a server. For information about uninstalling a replica, see [Section 32.6, “Removing a Replica”](#).

2.5. Renaming an IdM Server

It is not possible to change the host name of an IdM server machine after the server was set up. However, you can replace the original server with a new one. For more information, see [Section 32.7, “Renaming a Server Host System”](#).

[1] For details, see the [DNS Amplification Attacks](#) page.

Chapter 3. 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 7, Backing Up and Restoring Identity Management](#).

3.1. Explaining IdM Replicas

Replicas are created as clones of the initial master servers. Once a replica is created, it is functionally identical to the master server: servers and replicas created from these servers share the same internal information about users, machines, certificates, and configured policies.

Note

For more information on the types of machines in the IdM topology, see [Section 1.2, "The Identity Management Domain"](#).

Replication is the process of copying data between replicas. The information between replicas is shared using *multi-master replication*: all replicas joined through a replication agreement receive updates and are therefore considered data masters.

Replication agreements for directory data and for certificate data are managed separately. See [Section 32.1, "Types of Replication Agreements"](#).

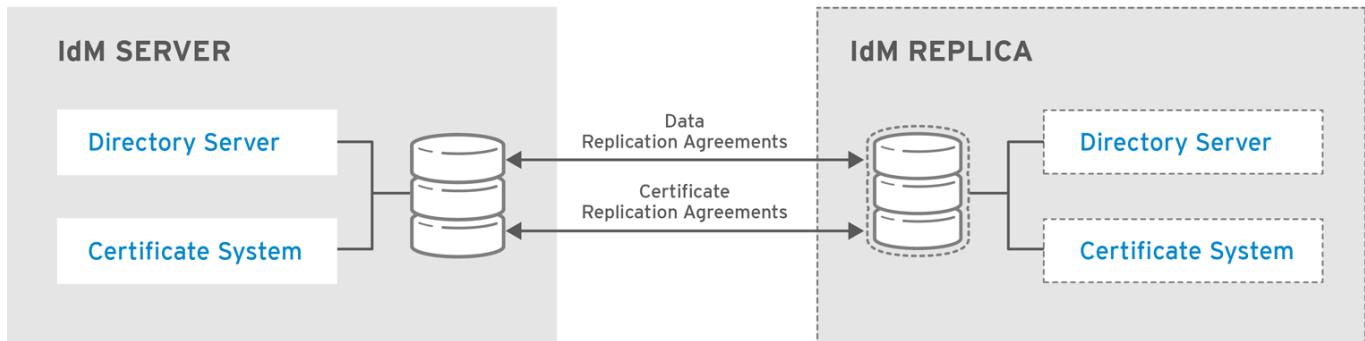


Figure 3.1. Server and Replica Agreements

3.2. Recommendations for Planning the Replica Topologies

Red Hat recommends to follow these guidelines:

Configure no more than 20 replicas in a single IdM domain

Red Hat guarantees to support environments with 20 replicas or less.

Configure at least two, but no more than four replication agreements per each replica

Configuring additional replication agreements ensures that information is replicated not just between the initial replica and the master server, but between other replicas as well.

- » If you create replica B from server A and then replica C from server A, replicas B and C are not directly joined, so data from replica B must first be replicated to server A before propagating to replica C.

Setting up an additional replication agreement between replica B and replica C ensures the data is replicated directly, which improves data availability, consistency, failover tolerance, and performance.

However, configuring too many replication agreements can have a serious negative impact on overall performance. Overloading replicas with a large number of replication agreements can potentially negate any performance improvements brought by setting up additional replication agreements.

See [Chapter 32, Managing Replicas and Replication Agreements](#) for details on managing replication agreements.

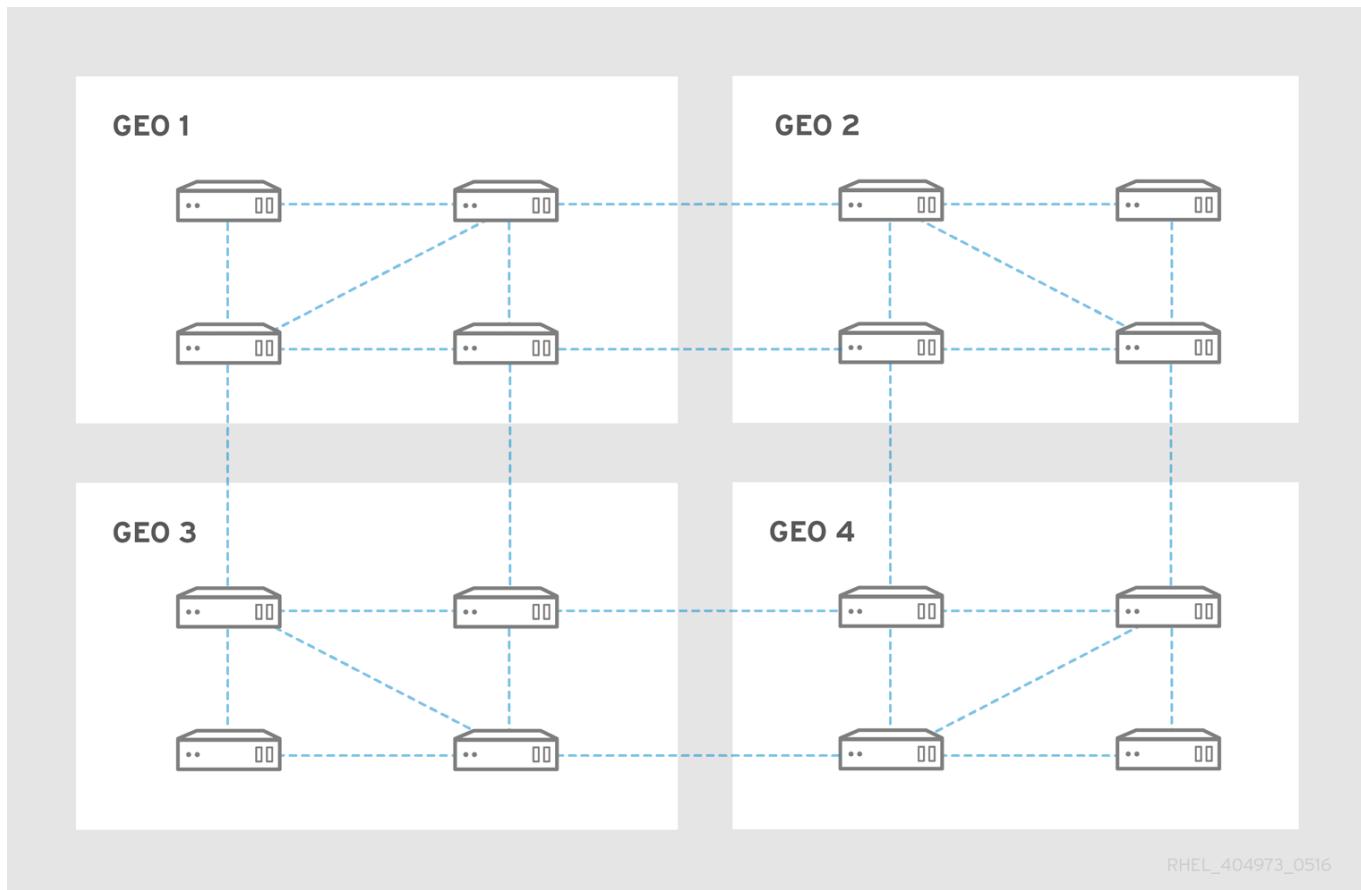


Figure 3.2. An Example Topology

3.2.1. Tight Cell Topology

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 the cells is a *tight cell*, where all servers have replication agreements with each other.
- » Each server has one replication agreement with another server *outside* the cell. This ensures that every cell is loosely coupled to every other cell in the domain.

To accomplish a tight cell topology:

- » 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.
- » In small offices, rather than using a replica, use SSSD to cache credentials and an off-site IdM server as the data back end.

3.3. 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 the replica is to be installed meets all of the prerequisites listed in [Section 2.1, “Prerequisites for Installing a Server”](#).

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 4.2 packages, then the replica must also run on Red Hat Enterprise Linux 7 or later and use IdM version 4.2 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 2.1.4, “Port Requirements”](#), 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 2.1.4, “Port Requirements”](#).

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.

3.4. 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.

3.4.1. 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 192.0.2.2
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.

3.4.2. Installing a Replica with DNS

To install a replica with integrated DNS, follow the procedure for installing without DNS described in [Section 3.4.1, “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 20.1, “Installing DNS Services Into an Existing Server”](#), and add the DNS records manually, as described in [Section 20.5.4, “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
@$NAME SERVER ${i}.${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.
...
...
```

3.4.2.1. Setting up Additional Name Servers

IdM adds the newly-configured IdM DNS server to the list of DNS servers in the **/etc/resolv.conf** file. It is recommended to manually add other DNS servers as backup servers in case the IdM server becomes unavailable. For example:

```
search example.com

; the IdM server
nameserver 192.0.2.1

; backup DNS servers
nameserver 198.51.100.1
nameserver 198.51.100.2
```

For more details about configuring **/etc/resolv.conf**, see the `resolv.conf(5)` man page.

3.4.3. 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.

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.



Warning

Red Hat strongly recommends to install the CA on more than one server in the IdM domain. For information on installing a replica of the initial server including the CA services, see [Section 3.4.3, “Installing a replica from a server with a Certificate System CA installed”](#).

If you install the CA on only one server, you risk losing the CA configuration without a chance of recovery if the server fails. See [Section A.2.5, “Recovering a Lost CA Server”](#) for details.

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 3.4.1, “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 3.4.1, “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 2.3.6, “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
```

3.4.4. 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.

3.4.5. 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
```

3.5. 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-csreplica-manage** utility.

For more information on adding additional replication agreements, see [Chapter 32, Managing Replicas and Replication Agreements](#).

3.6. Uninstalling an IdM Replica

For information on how to uninstall an IdM Replica, see [Section 32.6, “Removing a Replica”](#).

3.7. Renaming an IdM Replica

It is not possible to change the host name of an IdM replica machine after the replica was set up. However, you can replace the original replica with a new one. For more information, see [Section 32.7, “Renaming a Server Host System”](#).

3.8. Changing Load Balancing for IdM Servers and Replicas

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 4. 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.

4.1. 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**.
- » 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.

4.2. Prerequisites for Installing a Client

The following limitations apply to installing an IdM client:

- » Installing and running IdM in the Federal Information Processing Standard (FIPS) mode is not supported. Disable FIPS on your system before installing an IdM server, replica, or client, and do not enable it after the installation.
- » Red Hat recommends to disable the Name Service Cache Daemon (NSCD) on Identity Management machines. 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 conflicts between NSCD and SSSD.

4.2.1. Opening the IdM Required System Ports

IdM clients connect to a number of ports on IdM servers to communicate with their services. These ports must be open **on the IdM servers** to work. For more information on which ports IdM requires, see [Section 2.1.4, “Port Requirements”](#).

On a client, open these ports in the outgoing direction. If you are using a firewall that does not filter outgoing packets, such as **firewalld**, no further action is required.

4.3. 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.

4.3.1. 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.

- 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](#).

- 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,0=EXAMPLE.COM
Issuer: CN=Certificate Authority,0=EXAMPLE.COM
Valid From: Tue Aug 13 09:29:07 2016 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/sssd/sssd.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.
```

- 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
```

- Run the **ipa-client-automount** command which automatically configures NFS for IdM. For more information on **ipa-client-automount**, see [Section 21.2.1, “Configuring NFS Automatically”](#).

4.3.2. 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 4.1. 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 4.2. 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 4.3. 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 4.4. 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 4.5. 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
```

4.4. 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.

4.4.1. 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 4.4.2, “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/sssd/sssd.conf** file to point to the IdM domain.

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

[sssd]
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
```

- 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
...
```

- 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
```

- 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
```

- 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_authok
...
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_authok
...
session     optional      pam_sss.so

```

» For **/etc/pam.d/smardcard-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.

- Obtain the certificate from the server.

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

- 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.

- 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 24.1.2, “NIS Domain Name Requirements”](#).

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

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

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

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

- c. Enable SSSD as a source for `sudo` rules by adding the following `sudoers` 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 21.2.1, “Configuring NFS Automatically”](#).

4.4.2. Other Examples of Adding a Host Entry

[Section 4.4.1, “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.

4.4.2.1. Adding Host Entries from the Web UI

1. Open the **Identity** tab, and select the **Hosts** subtab.
2. Click **Add** at the top of the hosts list.

<input type="checkbox"/>	Host name	Description	Enrolled
<input type="checkbox"/>	server.example.com		True

Showing 1 to 1 of 1 entries.

Figure 4.1. Adding Host Entries

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

Host Name *	DNS Zone *
server	zone.example.com.
Class	
IP Address	192.0.2.1
Force	<input checked="" type="checkbox"/>
* Required field	
Add Add and Add Another Add and Edit Cancel	

Figure 4.2. Add Host Wizard

DNS zones can be created in IdM, which is described in [Section 20.5.1, “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.

4. Click the **Add and Edit** button to go directly to the expanded entry page and fill in more attribute information. Information about the host hardware and physical location can be included with the host entry.

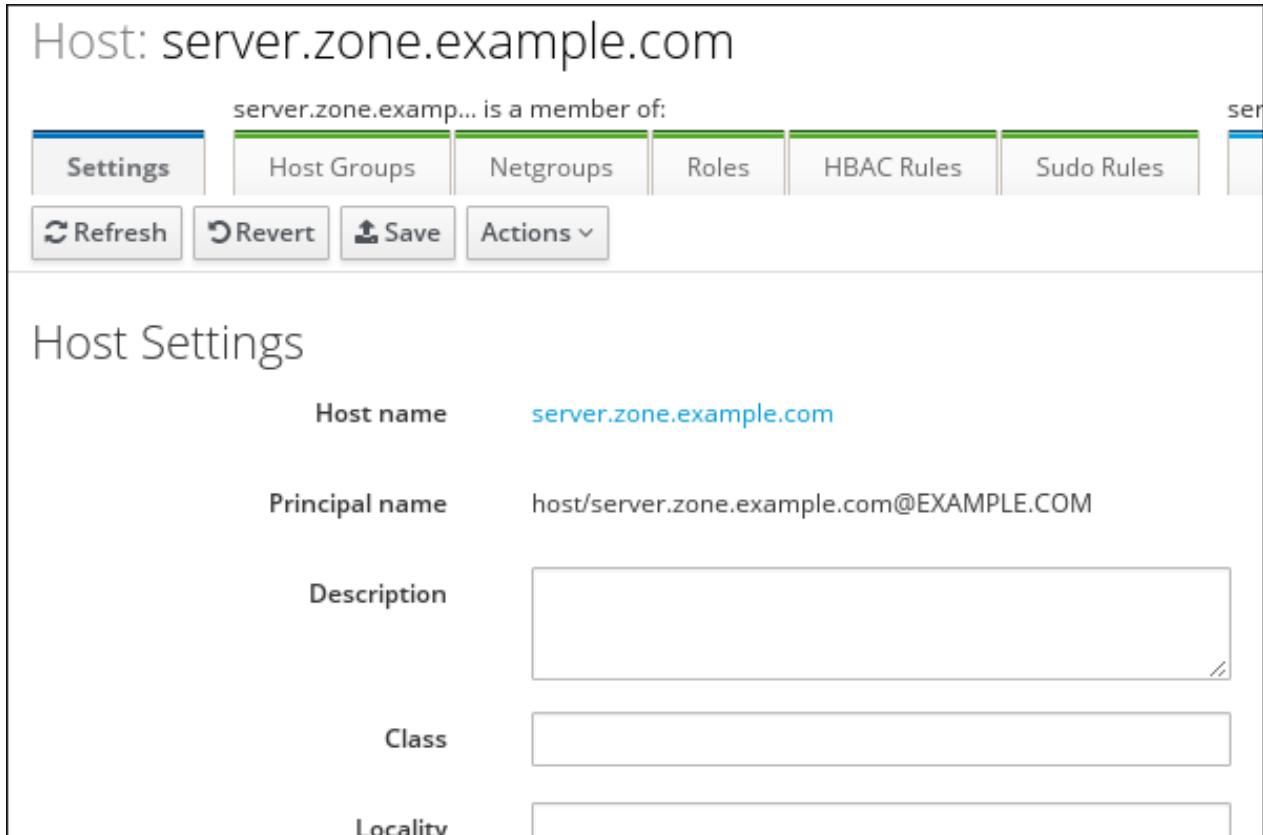


Figure 4.3. Expanded Entry Page

4.4.2.2. 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 4.6. 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 4.7. 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
```

4.5. 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.

- Run the kickstart script.

4.6. 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 re-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.

4.7. 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 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.

4.8. 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 4.4.2, “Other Examples of Adding a Host Entry”](#).
2. The second administrator installs the IdM client packages on the machine, as in [Section 4.3, “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.

4.9. 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
```



Warning

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

The DNS entries for the host are not deleted automatically during the uninstallation. To remove them manually, see [Section 20.5.6, “Deleting Records from DNS Zones”](#).

4.10. 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 5. 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.

5.1. Migrating the IdM Server to Red Hat Enterprise Linux 7

As is covered in [Section 30.8, “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 3.3, “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-*
```

- 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 2.1.4, “Port Requirements”](#).

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

- 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}
```

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

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

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

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

- 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.

- 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 "20161127184547" removed.
[root@rhel6 ~]# getcert stop-tracking -d /var/lib/pki-
ca/alias -n "ocspSigningCert cert-pki-ca"
Request "20161127184548" removed.
[root@rhel6 ~]# getcert stop-tracking -d /var/lib/pki-
ca/alias -n "subsystemCert cert-pki-ca"
Request "20161127184549" removed.
[root@rhel6 ~]# getcert stop-tracking -d /etc/httpd/alias -n
ipaCert
Request "20161127184550" 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 "20161127184743" 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 "20161127184744" 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 "20161127184745" 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/pwdfile.txt
New tracking request "20161127184746" 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 ~]# service httpd restart
```

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 3, 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 6. The Basics of Managing the IdM Server and Services

This chapter describes the Identity Management command-line and UI tools that are available to manage the IdM server and services, including methods for authenticating to IdM.

6.1. Starting and Stopping the IdM Server

A number of different services are installed together with an IdM server, including Directory Server, Certificate Authority (CA), DNS, Kerberos, and others. Use the `ipactl` utility to stop, start, or restart the entire IdM server along with all the installed services.

To start the entire IdM server:

```
# ipactl start
```

To stop the entire IdM server:

```
# ipactl stop
```

To restart the entire IdM server:

```
# ipactl restart
```

If you only want to stop, start, or restart an individual service, use the `systemctl` utility, described in the [System Administrator's Guide](#). For example, using `systemctl` to manage individual services is useful when customizing the Directory Server behavior: the configuration changes require restarting the Directory Server instance, but it is not necessary to restart all the IdM services.



Important

To restart multiple IdM domain services, Red Hat always recommends to use `ipactl`. Because of dependencies between the services installed with the IdM server, the order in which they are started and stopped is critical. The `ipactl` utility ensures that the services are started and stopped in the appropriate order.

6.2. Logging into IdM Using Kerberos

IdM uses the Kerberos protocol to support single sign-on. With Kerberos, the user only needs to present the correct user name and password once. Then the user can access IdM services without the system prompting for the credentials again.

By default, only machines that are members of the IdM domain can use Kerberos to authenticate to IdM. However, it is possible to configure external systems for Kerberos authentication as well; for more information, see [Section 6.4.3, “Configuring an External System for Kerberos Authentication to the Web UI”](#).

Using kinit

To log in to IdM from the command line, use the **kinit** utility.

Note

To use **kinit**, the *krb5-workstation* package must be installed.

When run without specifying a user name, **kinit** logs into IdM under the user name of the user that is currently logged-in on the local system. For example, if you are logged-in as **local_user** on the local system, running **kinit** attempts to authenticate you as the **local_user** IdM user:

```
[local_user@server ~]$ kinit
Password for local_user@EXAMPLE.COM:
```

Note

If the user name of the local user does not match any user entry in IdM, the authentication attempt fails.

To log in as a different IdM user, pass the required user name as a parameter to the **kinit** utility. For example, to log in as the **admin** user:

```
[local_user@server ~]$ kinit admin
Password for admin@EXAMPLE.COM:
```

Obtaining Kerberos Tickets Automatically

The **pam_krb5** pluggable authentication module (PAM) and SSSD can be configured to automatically obtain a TGT for a user after a successful login into the desktop environment on an IdM client machine. This ensures that after logging in, the user is not required to run **kinit**.

On IdM systems that have IdM configured in SSSD as the identity and authentication provider, SSSD obtains the TGT automatically after the user logs in with the corresponding Kerberos principal name.

For information on configuring **pam_krb5**, see the **pam_krb5(8)** man page. For general information about PAM, see the [System-Level Authentication Guide](#).

Storing Multiple Kerberos Tickets

By default, Kerberos only stores one ticket per logged-in user in the credential cache. Whenever a user runs **kinit**, Kerberos overwrites the currently-stored ticket with the new ticket. For example, if you use **kinit** to authenticate as **user_A**, the ticket for **user_A** will be lost after you authenticate again as **user_B**.

To obtain and store another TGT for a user, set a different credential cache, which ensures the contents of the previous cache are not overwritten. You can do this in one of the following two ways:

- » Run the **export KRB5CCNAME=path_to_different_cache** command, and then use **kinit** to obtain the ticket.
- » Run the **kinit -c path_to_different_cache** command, and then reset the **KRB5CCNAME** variable.

To restore the original TGT stored in the default credential cache:

1. Run the **kdestroy** command.
2. Restore the default credential cache location using the **unset \$KRB5CCNAME** command.

Checking the Current Logged-in User

To verify what TGT is currently stored and used for authentication, use the **klist** utility to list cached tickets. In the following example, the cache contains a ticket for **user_A**, which means that only **user_A** is currently allowed to access IdM services:

```
$ klist
Ticket cache: KEYRING:persistent:0:0
Default principal: user_A@EXAMPLE.COM

Valid starting     Expires            Service principal
11/10/2015 08:35:45  11/10/2015 18:35:45
krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

6.3. The IdM Command-Line Utilities

The basic command-line script for IdM is named **ipa**. The **ipa** script is a parent script for a number of subcommands. These subcommands are then used to manage IdM. For example, the **ipa user-add** command adds a new user:

```
$ ipa user-add user_name
```

Command-line management has certain benefits over management in UI; for example, the command-line utilities allow management tasks to be automated and performed repeatedly in a consistent way without manual intervention. Additionally, while most management operations are available both from the command line and in the web UI, some tasks can only be performed from the command line.

Note

This section only provides a general overview of the **ipa** subcommands. More information is available in the other sections dedicated to specific areas of managing IdM. For example, for information about managing user entries using the **ipa** subcommands, see [Chapter 8, Managing User Accounts](#).

The **ipa** script can display help about a particular set of subcommands: a *topic*. To display the list of available topics, use the **ipa help topics** command:

```
$ ipa help topics
```

<code>automember</code>	Auto Membership Rule.
<code>automount</code>	Automount
<code>caacl</code>	Manage CA ACL rules.
...	

To display help for a particular topic, use the `ipa help topic_name` command. For example, to display information about the `automember` topic:

```
$ ipa help automember
```

Auto Membership Rule.

Bring clarity to the membership of hosts and users by configuring inclusive or exclusive regex patterns, you can automatically assign a new entries into a group or hostgroup based upon attribute information.

...

EXAMPLES:

Add the initial group or hostgroup:

```
ipa hostgroup-add --desc="Web Servers" webservers
ipa group-add --desc="Developers" devel
```

...

The `ipa` script can also display a list of available `ipa` commands. To do this, use the `ipa help commands` command:

<code>\$ ipa help commands</code>	
<code>automember-add</code>	Add an automember rule.
<code>automember-add-condition</code>	Add conditions to an automember rule.
...	

For detailed help on the individual `ipa` commands, add the `--help` option to a command. For example:

```
$ ipa automember-add --help
```

Usage: ipa [global-options] automember-add AUTOMEMBER-RULE [options]

Add an automember rule.

Options:

<code>-h, --help</code>	show this help message and exit
-------------------------	---------------------------------

<code>--desc=STR</code>	A description of this auto member rule
-------------------------	--

...

For more information about the `ipa` utility, see the `ipa(1)` man page.

6.3.1. Setting a List of Values

IdM stores entry attributes in lists. For example:

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

Any update to a list of attributes overwrites the previous list. For example, an attempt to add a single attribute by only specifying this attribute replaces the whole previously-defined list with the single new attribute. Therefore, when changing a list of attributes, you must specify the whole updated list.

IdM supports the following methods of supplying a list of attributes:

- » Using the same command-line argument multiple times within the same command invocation. For example:

```
$ ipa permission-add --permissions=read --permissions=write --
permissions=delete
```

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

```
$ ipa permission-add --permissions={read,write,delete}
```

6.3.2. Using Special Characters

When passing command-line arguments in **ipa** commands that include special characters, such as angle brackets (< and >), ampersand (&), asterisk (*), or vertical bar (|), you must escape these characters by using a backslash (\). For example, to escape an asterisk (*):

```
$ ipa certprofile-show certificate_profile --out=exported\*profile.cfg
```

Commands containing unescaped special characters do not work as expected because the shell cannot properly parse such characters.

6.4. The IdM Web UI

The Identity Management web UI is a web application for IdM administration. It has most of the capabilities of the **ipa** command-line utility. Therefore, the users can choose whether they want to manage IdM from the UI or from the command line.

Note

Management operations available to the logged-in user depend on the user's access rights. For the **admin** user and other users with administrative privileges, all management tasks are available. For regular users, only a limited set of operations related to their own user account is available.

Supported Web Browsers

Identity Management supports the following browsers for connecting to the web UI:

- » Mozilla Firefox 38 and later
- » Google Chrome 46 and later

6.4.1. Accessing the Web UI and Authenticating

The web UI can be accessed both from IdM server and client machines, as well as from machines outside of the IdM domain. However, to access the UI from a non-domain machine, you must first configure the non-IdM system to be able to connect to the IdM Kerberos domain; see [Section 6.4.3, “Configuring an External System for Kerberos Authentication to the Web UI”](#) for more details.

Accessing the Web UI

To access the web UI, type the IdM server URL into the browser address bar:

`https://server.example.com`

This opens the IdM web UI login screen in your browser.

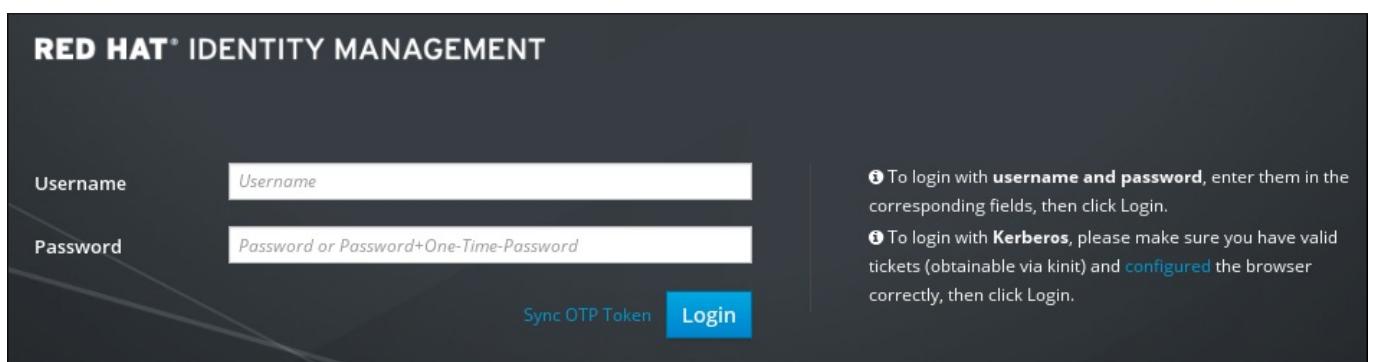


Figure 6.1. Web UI Login Screen

Available Login Methods

The user can authenticate to the web UI in two ways:

With an active Kerberos ticket

If the user has a valid TGT obtained with the `kinit` utility, clicking **Login** automatically authenticates the user. Note that the browser must be configured properly to support Kerberos authentication.

For information on obtaining a Kerberos TGT, see [Section 6.2, “Logging into IdM Using Kerberos”](#). For information on configuring the browser, see [Section 6.4.2, “Configuring the Browser for Kerberos Authentication”](#).

By providing user name and password

To authenticate using a user name and password, enter the user name and password on the web UI login screen.

IdM also supports one-time password (OTP) authentication. For more information, see [Section 9.5, “One-Time Passwords”](#).

After the user authenticates successfully, the IdM management window opens.

The screenshot shows the Red Hat Identity Management (IdM) web interface. At the top, there's a navigation bar with tabs for 'Identity', 'Policy', 'Authentication', 'Network Services', and 'IPA Server'. Below that is a secondary navigation bar with tabs for 'Users', 'User Groups', 'Hosts', 'Host Groups', 'Netgroups', 'Services', and 'Automember'. The 'Users' tab is selected. On the left, a sidebar titled 'User categories' has three items: 'Active users' (selected), 'Stage users', and 'Preserved users'. The main content area is titled 'Active users' and contains a table with one row. The table columns are: User login (checkbox), First name, Last name, Status, UID, and Email address. The single row shows 'admin' as the user login, with the status set to 'Enabled'.

Figure 6.2. The IdM Web UI Layout

Web UI Session Length

The default web UI session expiration period is 20 minutes. If the user does not perform any action for 20 minutes, the web UI logs the user out. However, if the user was logged in using Kerberos, the web UI automatically logs the user in again.

6.4.2. Configuring the Browser for Kerberos Authentication

To enable authentication with Kerberos credentials, you must configure your browser to support Kerberos negotiation for accessing the IdM domain. Note that if your browser is not configured properly for Kerberos authentication, an error message appears after clicking **Login** on the IdM web UI login screen.

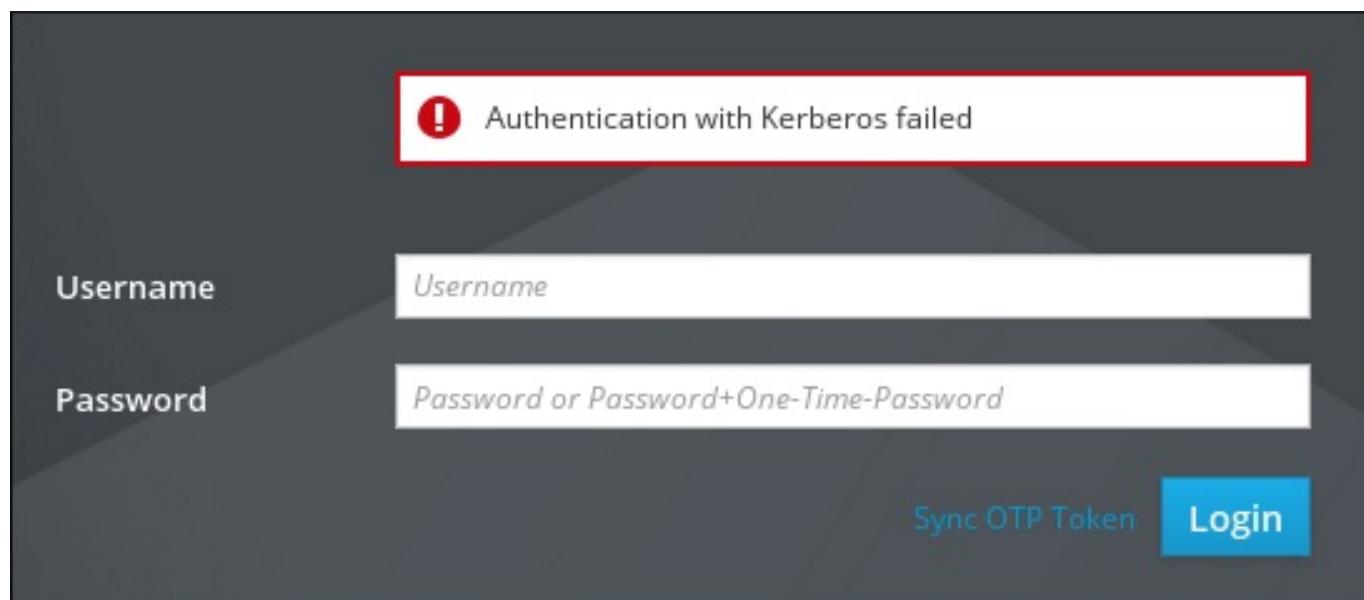


Figure 6.3. Kerberos Authentication Error

You can configure your browser for Kerberos authentication in three ways:

- » Automatically from the IdM web UI. This option is only available for Firefox. See [Section 6.4.2, “Automatic Firefox Configuration in the Web UI”](#) for details.

- » Automatically from the command line during the IdM client installation. This option is only available for Firefox. See [Section 6.4.2, “Automatic Firefox Configuration from the Command Line”](#) for details.
- » Manually in the Firefox configuration settings. This option is available for all supported browsers. See [Section 6.4.2, “Manual Browser Configuration”](#) for details.

Note

The System-Level Authentication Guide includes a [troubleshooting guide for Kerberos authentication in Firefox](#). If Kerberos authentication is not working as expected, see this troubleshooting guide for more advice.

Automatic Firefox Configuration in the Web UI

To automatically configure Firefox from the IdM web UI:

1. Click the link for browser configuration on the web UI login screen.

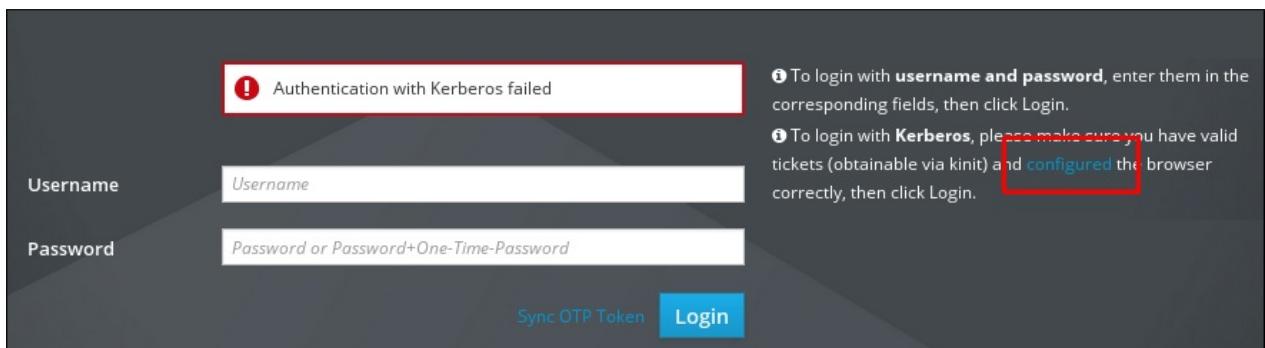


Figure 6.4. Link to Configuring the Browser in the Web UI

2. Choose the link for Firefox configuration to open the Firefox configuration page.

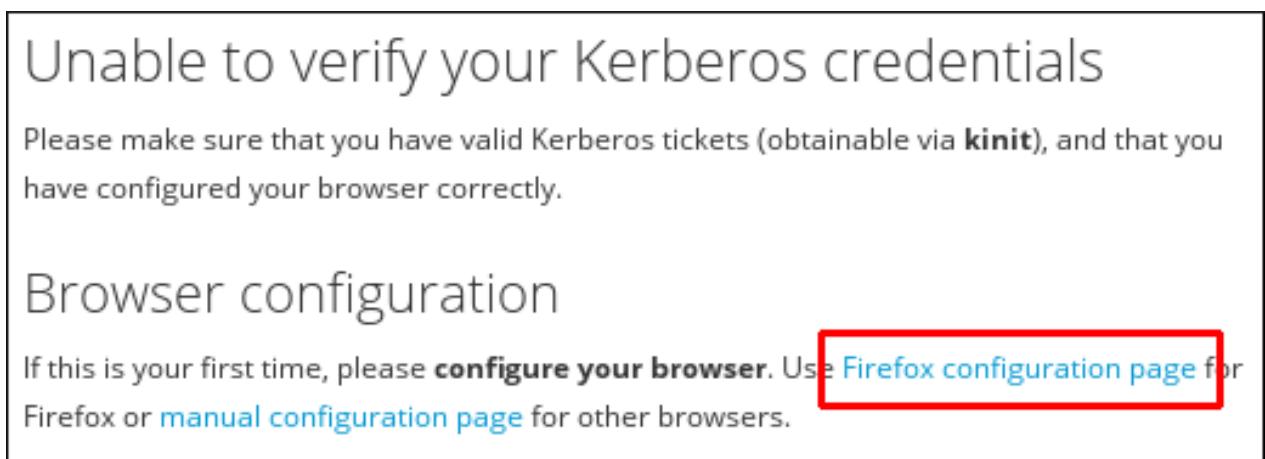


Figure 6.5. Link to the Firefox Configuration Page

3. Follow the steps on the Firefox configuration page.

Automatic Firefox Configuration from the Command Line

Firefox can be configured from the command line during IdM client installation. To do this, use the `--configure-firefox` option when installing the IdM client with the `ipa-client-install` utility:

```
# ipa-client-install --configure-firefox
```

The `--configure-firefox` option creates a global configuration file with default Firefox settings that enable Kerberos for single sign-on (SSO).

Manual Browser Configuration

To manually configure your browser:

1. Click the link for browser configuration on the web UI login screen.

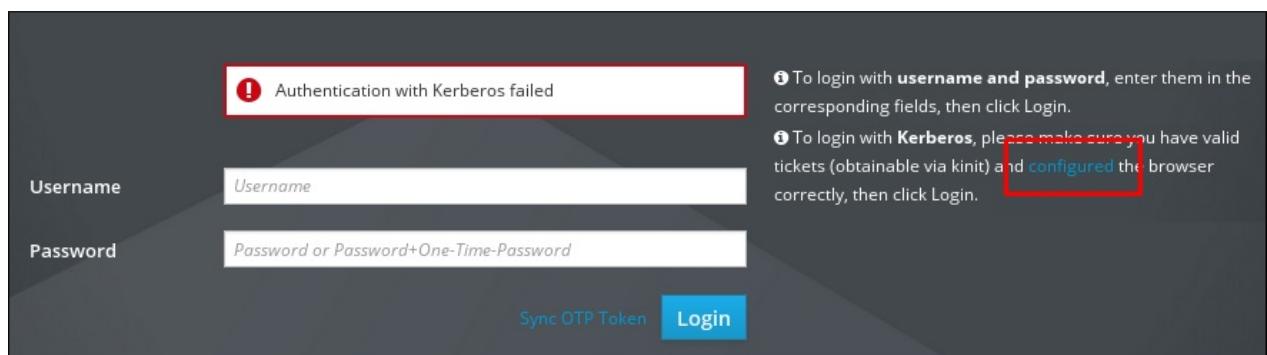


Figure 6.6. Link to Configuring the Browser in the Web UI

2. Choose the link for manual browser configuration.

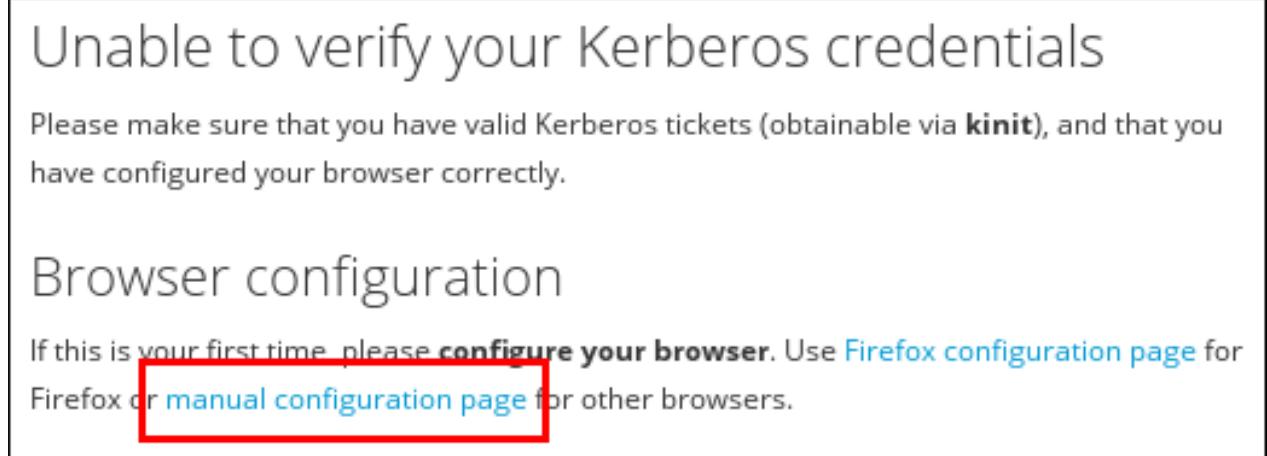


Figure 6.7. Link to the Manual Configuration Page

3. Look for the instructions to configure your browser and follow the steps.

6.4.3. Configuring an External System for Kerberos Authentication to the Web UI

To enable Kerberos authentication to the web UI from a system that is not a member of the IdM domain, you must define an IdM-specific Kerberos configuration file on the external machine. Enabling Kerberos authentication on external systems is especially useful when your infrastructure includes multiple realms or overlapping domains.

To create the Kerberos configuration file:

1. Copy the `/etc/krb5.conf` file from the IdM server to the external machine. For example:

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



Warning

Do not overwrite the existing `krb5.conf` file on the external machine.

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 the browser on the external machine as described in [Section 6.4.2, "Configuring the Browser for Kerberos Authentication"](#).

Users on the external system can now use the `kinit` utility to authenticate against the IdM server domain.

6.4.4. Proxy Servers and Port Forwarding in the Web UI

Using proxy servers to access the web UI does not require any additional configuration in IdM.

Port forwarding is not supported with the IdM server. However, because it is possible to use proxy servers, an operation similar to port forwarding can be configured using proxy forwarding with OpenSSH and the SOCKS option. This can be configured using the `-D` option of the `ssh` utility; for more information on using `-D`, see the `ssh(1)` man page.

Chapter 7. 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.

7.1. 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 7.1.3, “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

7.1.1. 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.

7.1.2. 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-Date: 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.

7.1.3. 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/ntp
/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/pwdfile.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/sssd/sssd.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/sssd.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/opendnssec/conf.xml
/etc/opendnssec/kasp.xml
/etc/ipa/dnssec/softhsm2.conf
/etc/ipa/dnssec/softhsm_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

```

7.2. 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.

7.2.1. 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

To avoid authentication problems on an IdM master, clear the SSSD cache after a restore:

1. Stop the SSSD service:

```
[root@server ~]# systemctl stop sssd
```

2. Remove all cached content from SSSD:

```
[root@server ~]# find /var/lib/sss/ ! -type d | xargs rm -f
```

3. Start the SSSD service:

```
[root@server ~]# systemctl start sssd
```

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.

7.2.2. 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-csreplica-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.

7.2.3. 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 and System Identities in a Linux Domain

Chapter 8. Managing User Accounts

This chapter covers general management and configuration of user accounts.

8.1. Setting up User Home Directories

It is recommended that every user has a home directory configured. The default expected location for user home directories is in the `/home/` directory. For example, IdM expects a user with the `user_login` login to have a home directory set up at `/home/user_login`.



Note

You can change the default expected location for user home directories using the `ipa config-mod` command.

IdM does not automatically create home directories for users. However, you can configure a PAM home directory module to create a home directory automatically when a user logs in. Alternatively, you can add home directories manually using NFS shares and the `automount` utility.

8.1.1. Mounting Home Directories Automatically Using the PAM Home Directory Module

Supported PAM Home Directory Modules

To configure a PAM home directory module to create home directories for users automatically when they log in to the IdM domain, use one of the following PAM modules:

- » `pam_oddjob_mkhomedir`
- » `pam_mkhomedir`

IdM first attempts to use `pam_oddjob_mkhomedir`. If this module is not installed, IdM attempts to use `pam_mkhomedir` instead.

Configuring the PAM Home Directory Module

Enabling the PAM home directory module has local effect. Therefore, you must enable the module individually on each client and server where it is required.

To configure the module during the installation of the server or client, use the `--mkhomedir` option with the `ipa-server-install` or `ipa-client-install` utility when installing the machine.

To configure the module on an already installed server or client, use the `authconfig` utility. For example:

```
# authconfig --enablenmkhomedir --update
```

For more information on using `authconfig` to create home directories, see the [System-Level Authentication Guide](#).

8.1.2. Mounting Home Directories Manually

You can use an NFS file server to provide a `/home/` directory that will be available to all machines in the IdM domain, and then mount the directory on an IdM machine using the `automount` utility.

Potential Problems When Using NFS

Using NFS can potentially have negative impact on performance and security. For example, using NFS can lead to security vulnerabilities resulting from granting root access to the NFS user, performance issues with loading the entire `/home/` directory tree, or network performance issues for using remote servers for home directories.

To reduce the effect of these problems, it is recommended to follow these guidelines:

- » Use `automount` to mount only the user's home directory and only when the user logs in. Do not use it to load 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 this user. Because 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.

Configuring Home Directories Using NFS and `automount`

To manually add home directories to the IdM server from separate locations using NFS shares and `automount`:

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

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

2. Add a direct mapping to the new location's `auto.direct` file. The `auto.direct` file is the `automount` map automatically created by the `ipa-server-install` utility. In the following example, the mount point is `/share`:

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

For more details on using `automount` with IdM, see [Chapter 21, Using Automount](#).

8.2. User Lifecycle

Identity Management supports three user account states: *stage*, *active*, and *preserved*.

- » **Stage** users are not allowed to authenticate. This is an initial state. Some of the user account properties required for active users might not yet be set.
- » **Active** users are allowed to authenticate. All required user account properties must be set in this state.

- ▶ **Preserved** users are former **active** users. They are considered inactive and cannot authenticate to IdM. Preserved users retain most of the account properties they had as active users, but they are not part of any user groups.

Note

The list of users in the **preserved** state can provide a history of past user accounts.

User entries can also be permanently deleted from the IdM database. Deleting a user entry permanently removes the entry itself and all its information from IdM, including group memberships and passwords. Any external configuration for the user, such as the system account and home directory, is not deleted, but is no longer accessible through IdM.



Important

Deleted user accounts cannot be restored. When you delete a user account, all the information associated with the account is lost permanently.

A new administrator user can only be created by another administrator, such as the default **admin** user. If you accidentally delete all administrator accounts, the Directory Manager must create a new administrator manually in the Directory Server.

User Lifecycle Management Operations

To manage user provisioning, the administrator can move user accounts from one state to another. New user accounts can be added as either **active** or **stage**, but not as **preserved**.

IdM supports the following operations for user lifecycle management:

stage → **active**

When an account in the **stage** state is ready to be properly activated, the administrator moves it to the **active** state.

active → **preserved**

After the user leaves the company, the administrator moves the account to the **preserved** state.

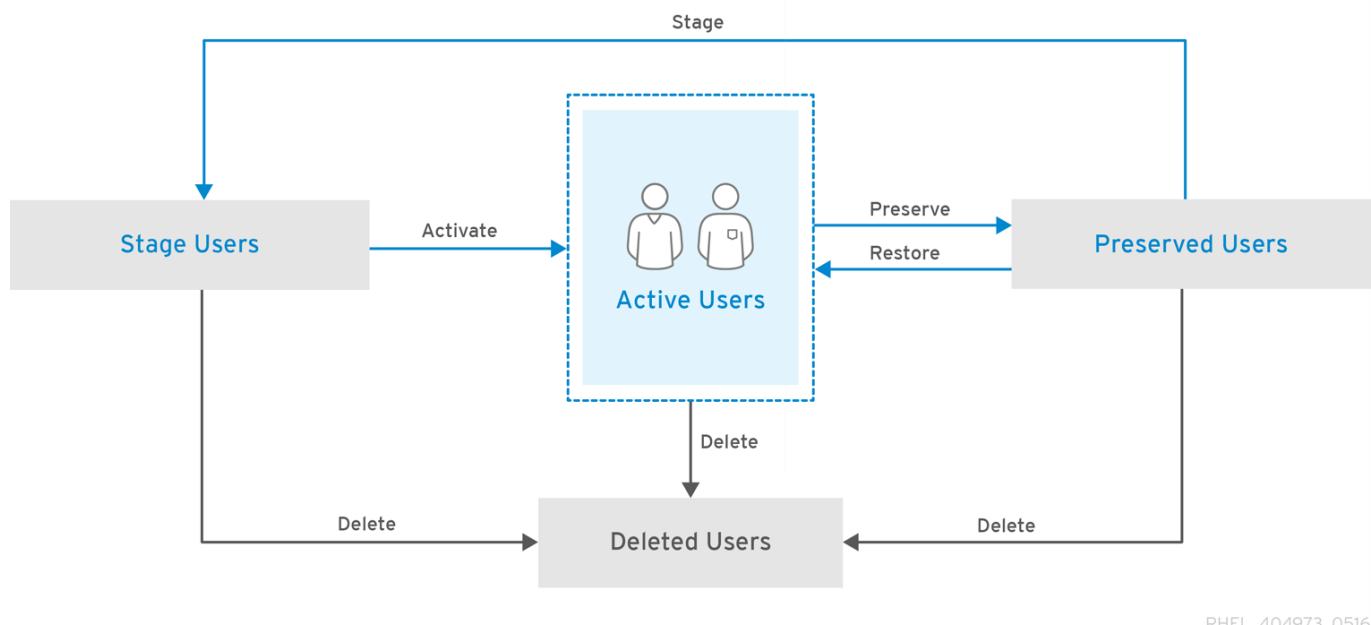
preserved → **active**

A former user joins the company again. The administrator restores the user account by moving it from the **preserved** state back to the **active** state.

preserved → **stage**

A former user is planning to join the company again. The administrator moves the account from the **preserved** state to the **stage** state to prepare the account for later reactivation.

You can also permanently delete active, stage, and preserved users from IdM. Note that you cannot move stage users to the **preserved** state, you can only delete them permanently.



RHEL_404973_0516

Figure 8.1. User Lifecycle Operations

8.2.1. Adding Stage or Active Users

Adding Users in the Web UI

1. Select the **Identity** → **Users** tab.
2. Select the **Active users** or **Stage users** category, depending on whether you want to add a user in the **active** or **stage** state.

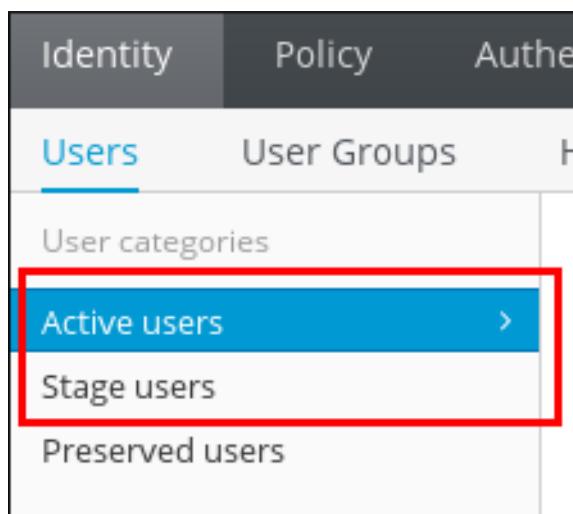
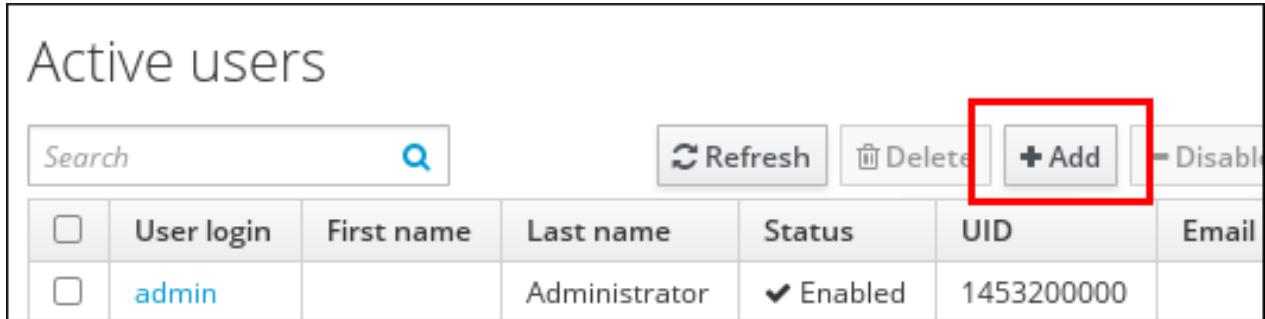


Figure 8.2. Selecting User Category

For more information about the **active** or **stage** user lifecycle states, see [Section 8.2, “User Lifecycle”](#).

3. Click **Add** at the top of the users list.



Active users						
	User login	First name	Last name	Status	UID	Email
<input type="checkbox"/>	admin		Administrator	✓ Enabled	1453200000	

Figure 8.3. Adding a User

4. Fill in the **Add User** form.

Note that if you do not set a user login manually, IdM generates the login automatically based on the specified first name and last name.

5. Click **Add**.

Alternatively, click **Add and Add Another** to start adding another user or **Add and Edit** to start editing the new user entry. For information on editing user entries, see [Section 8.3, “Editing Users”](#).

Adding Users from the Command Line

To add a new user in the **active** state, use the **ipa user-add** command. To add a new user in the **stage** state, use the **ipa stageuser-add** command.

Note

For more information about the **active** or **stage** user lifecycle states, see [Section 8.2, “User Lifecycle”](#).

When run without any options, **ipa user-add** and **ipa stageuser-add** prompt you for the minimum required user attributes and use default values for the other attributes.

Alternatively, you can add options specifying various attributes directly to the commands.

In the interactive session, after you run the command without any options, IdM proposes an automatically generated user login based on the provided first name and last name and displays it in brackets ([]). To accept the default login, confirm by pressing **Enter**. To specify a custom login, do not confirm the default and specify the custom login instead.

```
$ ipa user-add
First name: first_name
Last name: last_name
User login [default_login]: custom_login
```

Adding options to **ipa user-add** and **ipa stageuser-add** enables you to define custom values for many of the user attributes. This means that you can specify more information than in the interactive session. For example, to add a stage user:

```
$ ipa stageuser-add stage_user_login --first=first_name --last=last_name
--email=email_address
```

For a complete list of options accepted by **ipa user-add** and **ipa stageuser-add**, run the commands with the **--help** option added.

8.2.1.1. User Name Requirements

IdM supports user names that can be described by the following regular expression:

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

Note

User names ending with the trailing dollar sign (\$) are supported to enable Samba 3.x machine support.

If you add a user whose user name contains uppercase characters, IdM automatically converts the name to lowercase when saving it. Therefore, IdM always requires users to enter their user names all lowercase when logging in. Additionally, it is not possible to add users whose user names only differ in letter casing, such as **user** and **User**.

The default maximum length for user names is 32 characters. To change it, use the **ipa config-mod --maxusername** command. For example, to increase the maximum user name length to 64 characters:

```
$ ipa config-mod --maxusername=64
Maximum username length: 64
...
```

8.2.1.2. Defining a Custom UID or GID Number

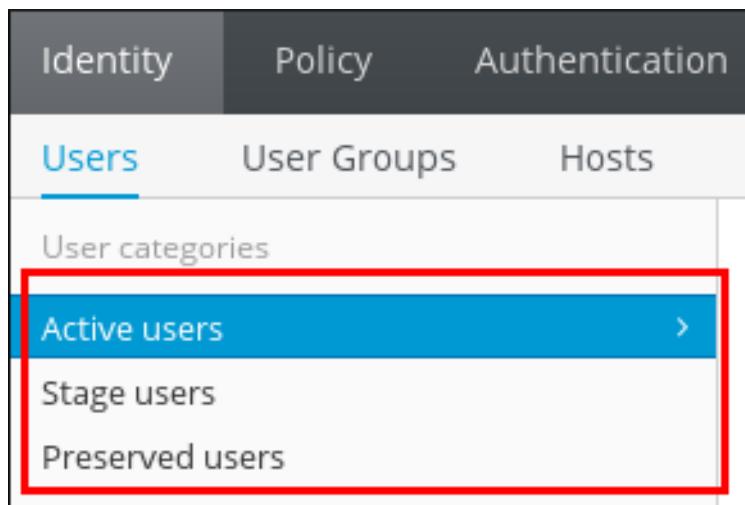
If you add a new user entry without specifying a custom UID or GID number, IdM automatically assigns an ID number that is next available in the ID range. This means that users' ID numbers are always unique. For more information about ID ranges, see [Chapter 11, Unique UID and GID Number Assignments](#).

When you specify a custom ID number, the server does not validate whether the custom ID number is unique. Due to this, multiple user entries might have the same ID number assigned. Red Hat recommends to prevent having multiple entries with the same ID number.

8.2.2. Listing Users and Searching for Users

Listing Users in the Web UI

1. Select the **Identity** → **Users** tab.
2. Select the **Active users**, **Stage users**, or **Preserved users** category.

**Figure 8.4. Listing Users**

Displaying Information About a User in the Web UI

To display detailed information about a user, click the name of the user in the list of users:

Active users						
Search		Refre				
<input type="checkbox"/>	User login	First name	Last name	Status	UID	Email address
<input type="checkbox"/>	admin		Administrator	✓ Enabled	1453200000	
<input type="checkbox"/>	user	User	User	✓ Enabled	1453200006	user1@example.com
<input type="checkbox"/>	user2	User2	User2	✓ Enabled	1453200007	user2@abc.idm.l
<input type="checkbox"/>	user3	User3	User3	✓ Enabled	1453200008	user3@abc.idm.l

Figure 8.5. Displaying User Information

Listing Users from the Command Line

To list all active users run the **ipa user-find** command. To list all stage users, use the **ipa stageuser-find** command. To list preserved users, run the **ipa user-find --preserved=true** command.

For example:

```
$ ipa user-find
-----
23 users matched
-----
User login: admin
Last name: Administrator
Home directory: /home/admin
Login shell: /bin/bash
UID: 1453200000
```

```
GID: 1453200000
Account disabled: False
Password: True
Kerberos keys available: True

User login: user
...
```

By adding options and arguments to **ipa user-find** and **ipa stageuser-find**, you can define the search criteria and filter the search results. For example, to display all active users with a specific title defined:

```
$ ipa user-find --title=user_title
-----
2 users matched
-----
User login: user
...
Job Title: Title
...

User login: user2
...
Job Title: Title
...
```

Similarly, to display all stage users whose login contains **user**:

```
$ ipa user-find user
-----
3 users matched
-----
User login: user
...
User login: user2
...
User login: user3
...
```

For a complete list of options accepted by **ipa user-find** and **ipa stageuser-find**, run the commands with the **--help** option added.

Displaying Information about a User from the Command Line

To display information about an active or preserved user, use the **ipa user-show** command:

```
$ ipa user-show user_login
User login: user_login
First name: first_name
Last name: last_name
...
```

To display information about a stage user, use the `ipa stageuser-show` command:

8.2.3. Activating, Preserving, Deleting, and Restoring Users

This section describes moving user accounts between different user lifecycle states. For details on the lifecycle states in IdM, see [Section 8.2, “User Lifecycle”](#).

Managing User Lifecycle in the Web UI

To activate a stage user:

- In the **Stage users** list, select the user to activate, and click **Activate**.

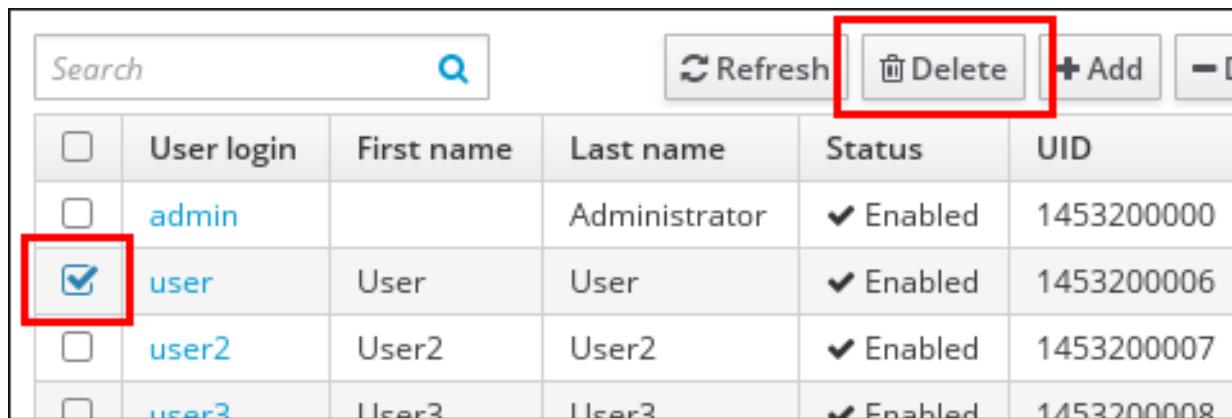


Stage Users					
	User login	First name	Last name	UID	Email address
<input checked="" type="checkbox"/>	user	User	User	-1	
Showing 1 to 1 of 1 entries.					

Figure 8.6. Activating a User

To preserve or delete a user:

- In the **Active users** or **Stage users** lists, select the user. Click **Delete**.



Active users					
	User login	First name	Last name	Status	UID
<input type="checkbox"/>	admin		Administrator	✓ Enabled	1453200000
<input checked="" type="checkbox"/>	user	User	User	✓ Enabled	1453200006
<input type="checkbox"/>	user2	User2	User2	✓ Enabled	1453200007
<input type="checkbox"/>	user3	User3	User3	✓ Enabled	1453200008

Figure 8.7. Deleting a User

- If you selected an active user, select **delete** or **preserve**. If you selected a stage user, you can only delete the user. The default UI option is **delete**.

For example, to preserve an active user:

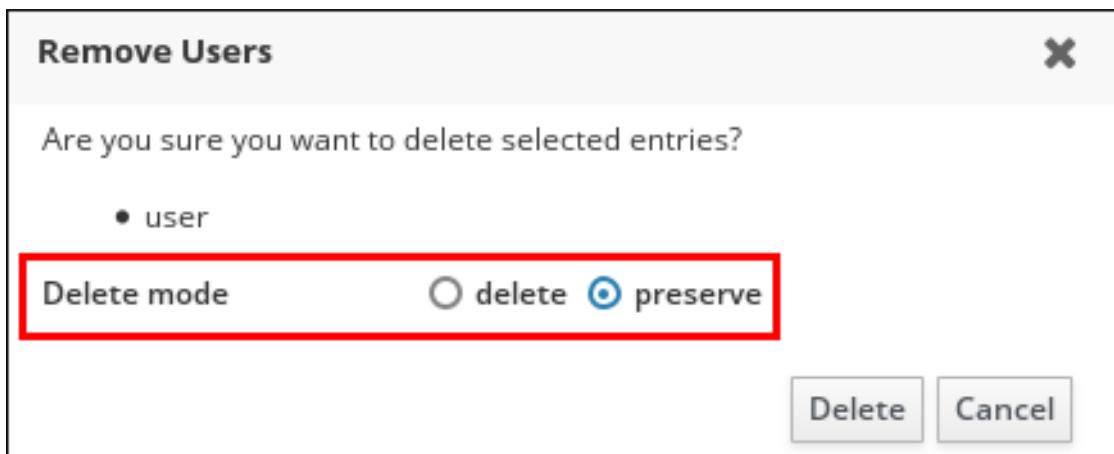


Figure 8.8. Selecting the Delete Mode in the Web UI

To confirm, click the **Delete** button.

To restore a preserved user:

- In the **Preserved users** list, select the user to restore, and click **Restore**.

Preserved users					
	User login	First name	Last name	UID	Email address
<input checked="" type="checkbox"/>	user	User	User	1453200006	
<input type="checkbox"/>	user2	User2	User2	1453200007	

Figure 8.9. Restoring a User

Note

Restoring a user does not restore all of the account's previous attributes. For example, the user's password is not restored and must be defined again.

Note that in the web UI, it is not possible to move a user from the **preserved** state to the **stage** state.

Managing User Lifecycle from the Command Line

To activate a user account by moving it from **stage** to **active**, use the **ipa stageuser-activate** command.

```
$ ipa stageuser-activate user_login
-----
Stage user user_login activated
-----
```

...

To preserve or delete a user account, use the **ipa user-del** or **ipa stageuser-del** commands.

- » To remove an active user permanently from the IdM database, run **ipa user-del** without any options.

```
$ ipa user-del user_login  
-----  
Deleted user "user3"  
-----
```

- » To preserve an active user account, run **ipa user-del** with the **--preserve** option.

```
$ ipa user-del --preserve user_login  
-----  
Deleted user "user_login"  
-----
```

- » To remove a stage user permanently from the IdM database, run **ipa stageuser-del**.

```
$ ipa stageuser-del user_login  
-----  
Deleted stage user "user_login"  
-----
```

Note

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 the **stdout** standard output stream when the command completes.

```
$ ipa user-del --continue user1 user2 user3
```

If **--continue** is not used, the command proceeds with deleting users until it encounters an error, after which it stops and exits.

To restore a preserved user account by moving it from **preserved** to **active**, use the **ipa user-undel** command.

```
$ ipa user-undel user_login  
-----  
Undeleted user account "user_login"  
-----
```

To restore a preserved user account by moving it from **preserved** to **stage**, use the **ipa user-stage** command.

```
$ ipa user-stage user_login  
-----  
Staged user account "user_login"  
-----
```

Note

Restoring a user account does not restore all of the account's previous attributes. For example, the user's password is not restored and must be defined again.

For more information about these commands and the options they accept, run them with the **--help** option added.

8.3. Editing Users

Editing Users in the Web UI

1. Select the **Identity → Users** tab.
2. Search the **Active users**, **Stage users**, or **Preserved users** category to find the user to edit.
3. Click the name of the user to edit.

	User login	First name	Last name	Status	UID
<input type="checkbox"/>	admin		Administrator	<input checked="" type="checkbox"/> Enabled	14532
<input type="checkbox"/>	user	User	User	<input checked="" type="checkbox"/> Enabled	14532

Figure 8.10. Selecting a User to Edit

4. Edit the user attribute fields as required.
5. Click **Save** at the top of the page.

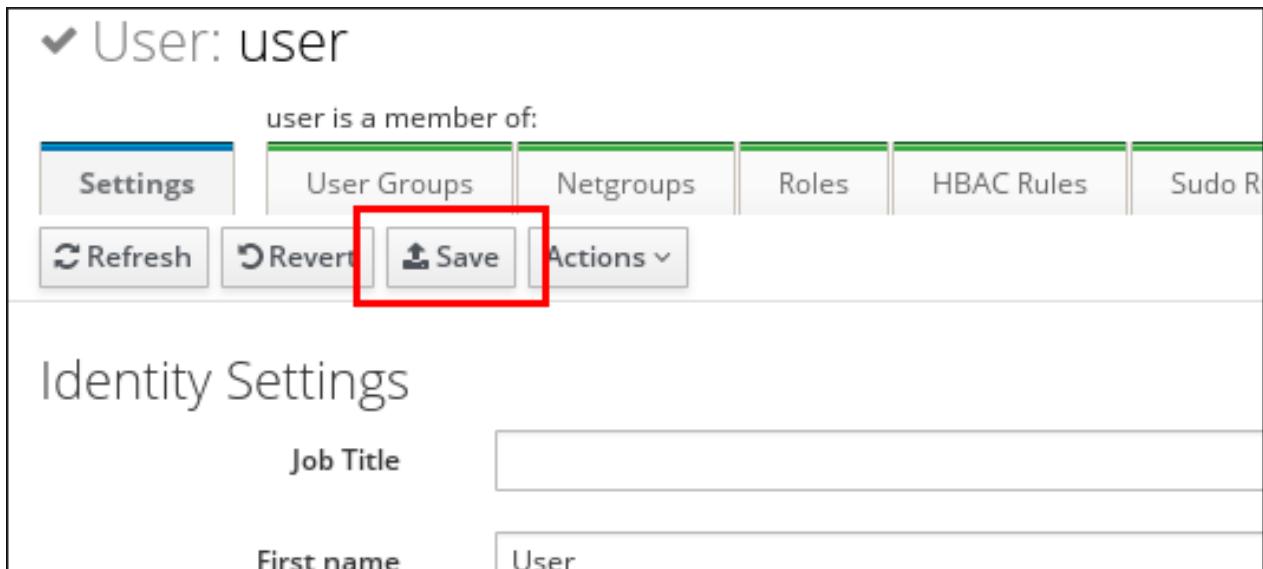


Figure 8.11. Save Modified User Attributes

Editing Users from the Command Line

To modify a user in the **active** or **preserved** states, use the **ipa user-mod** command. To modify a user in the **stage** state, use the **ipa stageuser-mod** command.

The **ipa user-mod** and **ipa stageuser-mod** commands accept the following options:

- » the user login, which identifies the user account to be modified
- » options specifying the new attribute values

For a complete list of user entry attributes that can be modified from the command line, see the list of options accepted by **ipa user-mod** and **ipa stageuser-mod**. To display the list of options, run the commands with the **--help** option added.

Simply adding an attribute option to **ipa user-mod** or **ipa stageuser-mod** overwrites the current attribute value. For example, the following changes a user's title or adds a new title if the user did not yet have a title specified:

```
$ ipa user-mod user_login --title=new_title
```

For LDAP attributes that are allowed to have multiple values, IdM also accepts multiple values. For example, a user can have two email addresses saved in their user account. To add an additional attribute value without overwriting the existing value, use the **--addattr** option together with the option to specify the new attribute value. For example, to add a new email address to a user account that already has an email address specified:

```
$ ipa user-mod user --addattr=mobile=new_mobile_number
-----
Modified user "user"
-----
User login: user
...
Mobile Telephone Number: mobile_number, new_mobile_number
...
```

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

```
$ ipa user-mod user --addattr=mobile=mobile_number_1 --  
addattr=mobile=mobile_number_2
```

The **ipa user-mod** command also accepts the **--setattr** option for setting attribute values and the **--delattr** option for deleting attribute values. These options are used in a way similar to using **--addattr**. For details, see the output of the **ipa user-mod --help** command.

Note

To overwrite the current email address for a user, use the **--email** option. However, to add an additional email address, use the **mail** option with the **--addattr** option:

```
$ ipa user-mod user --email=email@example.com  
$ ipa user-mod user --addattr=mail=another_email@example.com
```

8.4. Enabling and Disabling User Accounts

The administrator can disable and enable active user accounts. Disabling a user account deactivates the account. Disabled user accounts cannot be used to authenticate. A user whose account has been disabled cannot log into IdM and cannot use IdM services, such as Kerberos, or perform any tasks.

Disabled user accounts still exist within IdM and all of the associated information remains unchanged. Unlike preserved user accounts, disabled user accounts remain in the **active** state. Therefore, they are displayed in the output of the **ipa user-find** command. For example:

```
$ ipa user-find  
...  
User login: user  
First name: User  
Last name: User  
Home directory: /home/user  
Login shell: /bin/sh  
UID: 1453200009  
GID: 1453200009  
Account disabled: True  
Password: False  
Kerberos keys available: False  
...
```

Any disabled user account can be enabled again.



Note

After disabling a user account, existing connections remain valid until the user's Kerberos TGT and other tickets expire. After the ticket expires, the user will not be able to renew it.

Enabling and Disabling User Accounts in the Web UI

1. Select the **Identity → Users** tab.
2. From the **Active users** list, select the required user or users, and then click **Disable** or **Enable**.

Active users						
Search		Actions				
	User login	First name	Last name	Status	UID	Email address
<input type="checkbox"/>	admin		Administrator	<input checked="" type="checkbox"/> Enabled	1453200000	
<input checked="" type="checkbox"/>	user	User	User	<input checked="" type="checkbox"/> Enabled	1453200009	
<input type="checkbox"/>	user2	User2	User2	<input checked="" type="checkbox"/> Enabled	1453200007	

Figure 8.12. Disabling or Enabling a User Account

Disabling and Enabling User Accounts from the Command Line

To disable a user account, use the **ipa user-disable** command.

```
$ ipa user-disable user_login
-----
Disabled user account "user_login"
-----
```

To enable a user account, use the **ipa user-enable** command.

```
$ ipa user-enable user_login
-----
Enabled user account "user_login"
-----
```

8.5. Allowing Non-admin Users to Manage User Entries

By default, only the **admin** user is allowed to manage user lifecycle and disable or enable user accounts. To allow another, non-admin user to do this, create a new role, add the relevant permissions to this role, and assign the non-admin user to the role.

By default, IdM includes the following privileges related to managing user accounts:

Modify Users and Reset passwords

This privilege includes permissions to modify various user attributes.

User Administrators

This privilege includes permissions to add active users, activate non-active users, remove users, modify user attributes, and other permissions.

Stage User Provisioning

This privilege includes a permission to add stage users.

Stage User Administrator

This privilege includes permissions to perform a number of lifecycle operations, such as adding stage users or moving users between lifecycle states. However, it does not include permissions to move users to the active state.

For information on defining roles, permissions, and privileges, see [Section 28.4, “Defining Role-Based Access Controls”](#).

Allowing Different Users to Perform Different User Management Operations

The different privileges related to managing user accounts can be added to different users. For example, you can separate privileges for employee account entry and activation by:

- » Configuring one user as a *stage user administrator*, who is allowed to add future employees to IdM as stage users, but not to activate them.
- » Configuring another user as a *security administrator*, who is allowed to activate the stage users after their employee credentials are verified on the first day of employment.

To allow a user to perform certain user management operations, create a new role with the required privilege or privileges, and assign the user to that role.

Example 8.1. Allowing a Non-admin User to Add Stage Users

This example shows how to create a user who is only allowed to add new stage users, but not to perform any other stage user management operations.

1. Log in as the **admin** user or another user allowed to manage role-based access control.

```
$ kinit admin
```

2. Create a new custom role to manage adding stage users.
 - a. Create the **System Provisioning** role.

```
$ ipa role-add --desc "Responsible for provisioning stage
users" "System Provisioning"
-----
Added role "System Provisioning"
```

```
-----  
Role name: System Provisioning  
Description: Responsible for provisioning stage users
```

- b. Add the **Stage User Provisioning** privilege to the role. This privilege provides the ability to add stage users.

```
$ ipa role-add-privilege "System Provisioning" --  
privileges="Stage User Provisioning"  
Role name: System Provisioning  
Description: Responsible for provisioning stage users  
Privileges: Stage User Provisioning  
-----  
Number of privileges added 1  
-----
```

3. Grant a non-admin user the rights to add stage users.

- a. If the non-admin user does not yet exist, create a new user. In this example, the user is named **stage_user_admin**.

```
$ ipa user-add stage_user_admin --password  
First name: first_name  
Last name: last_name  
Password:  
Enter password again to verify:  
...
```

- b. Assign the **stage_user_admin** user to the **System Provisioning** role.

```
$ ipa role-add-member "System Provisioning" --  
users=stage_user_admin  
Role name: System Provisioning  
Description: Responsible for provisioning stage users  
Member users: stage_user_admin  
Privileges: Stage User Provisioning  
-----  
Number of members added 1  
-----
```

- c. To make sure the **System Provisioning** role is configured correctly, you can use the **ipa role-find** command to display the role settings.

```
$ ipa role-find "System Provisioning"  
-----  
1 role matched  
-----  
Role name: System provisioning  
Description: Responsible for provisioning stage users  
Member users: stage_user_admin  
Privileges: Stage User Provisioning  
-----  
Number of entries returned 1  
-----
```

4. Test adding a new stage user as the **stage_user_admin** user.

- Log in as **stage_user_admin**. Note that if you created **stage_user_admin** as a new user in one of the previous steps, IdM will ask you to change the initial password set by **admin**.

```
$ kinit stage_user_admin
Password for stage_user_admin@EXAMPLE.COM:
Password expired. You must change it now.
Enter new password:
Enter it again:
```

- To make sure your Kerberos ticket for **admin** has been replaced with a Kerberos ticket for **stage_user_admin**, you can use the **klist** utility.

```
$ klist
Ticket cache: KEYRING:persistent:0:krb_ccache_xIlCQDW
Default principal: stage_user_admin@EXAMPLE.COM

Valid starting     Expires            Service principal
02/25/2016 11:42:20 02/26/2016 11:42:20
krbtgt/EXAMPLE.COM
```

- Add a new stage user.

```
$ ipa stageuser-add stage_user
First name: first_name
Last name: last_name
ipa: ERROR: stage_user: stage user not found
```



Note

The error that IdM reports after adding a stage user is expected. The **stage_user_admin** is only allowed to add stage users, not to display information about them. Therefore, instead of displaying a summary of the newly added **stage_user** settings, IdM displays the error.

The **stage_user_admin** user is not allowed to display information about stage users. Therefore, an attempt to display information about the new **stage_user** user while logged in as **stage_user_admin** fails:

```
$ ipa stageuser-show stage_user
ipa: ERROR: stage_user: stage user not found
```

To display information about **stage_user**, you can log in as **admin**:

```
$ kinit admin
Password for admin@EXAMPLE.COM:
$ ipa stageuser-show stage_user
User login: stage_user
```

```
First name: Stage
Last name: User
...
```

8.6. Using an External Provisioning System for Users and Groups

Identity Management supports configuring your environment, so that an external solution for managing identities is used to provision user and group identities in IdM. This section describes an example of such configuration. The example includes:

- » [Section 8.6.1, “Configuring User Accounts to Be Used by the External Provisioning System”](#)
- » [Section 8.6.2, “Configuring IdM to Automatically Activate Stage User Accounts”](#)
- » [Section 8.6.3, “Configuring the LDAP Provider of the External Provisioning System to Manage the IdM Identities”](#)

8.6.1. Configuring User Accounts to Be Used by the External Provisioning System

Outcome: Two IdM user accounts are configured to be used by the external provisioning system. The accounts are added to a group with an appropriate password policy. Using these accounts, the external provisioning system is able to manage user provisioning in IdM.

1. Create a user, **provisionator**, with the privileges to add stage users. The user account will be used by the external provisioning system to add new stage users.

- a. Add the **provisionator** user account:

```
$ ipa user-add provisionator --first=provisioning --
last=account --password
```

- b. Grant the **provisionator** user the required privileges.

Create a custom role, **System Provisioning**, to manage adding stage users:

```
$ ipa role-add --desc "Responsible for provisioning stage
users" "System Provisioning"
```

Add the **Stage User Provisioning** privilege to the role. This privilege provides the ability to add stage users:

```
$ ipa role-add-privilege "System Provisioning" --
privileges="Stage User Provisioning"
```

Add the **provisionator** user to the role:

```
$ ipa role-add-member --users=provisionator "System
Provisioning"
```

2. Create a user, **activator**, with the privileges to manage user accounts. The user account will be used to automatically activate stage users added by the external provisioning system.

- a. Add the **activator** user account:

```
$ ipa user-add activator --first=activation --last=account --password
```

- b. Grant the **activator** user the required privileges.

Add the user to the default **User Administrator** role:

```
$ ipa role-add-member --users=activator "User Administrator"
```

3. Create a user group for service and application accounts:

```
$ ipa group-add service-accounts
```

4. Update the password policy for the group. The following policy prevents password expiration and lockout for the account but compensates the potential risks by requiring complex passwords:

```
$ ipa pwpolicy-add service-accounts --maxlife=10000 --minlife=0 --history=0 --minclasses=4 --minlength=20 --priority=1 --maxfail=0 --failinterval=1 --lockouttime=0
```

5. Add the provisioning and activation accounts to the group for service and application accounts:

```
$ ipa group-add-member service-accounts --users={provisionator,activator}
```

6. Change the passwords for the user accounts:

```
$ kpasswd provisionator  
$ kpasswd activator
```

Changing the passwords is necessary because passwords of new IdM users expire immediately.

Additional resources:

- For details on adding new users, see [Section 8.2.1, “Adding Stage or Active Users”](#).
- For details on granting users the privileges required to manage other user accounts, see [Section 8.5, “Allowing Non-admin Users to Manage User Entries”](#).
- For details on managing IdM password policies, see [Chapter 22, Defining Password Policies](#).

8.6.2. Configuring IdM to Automatically Activate Stage User Accounts

Outcome: A script is created for activating stage users. The system runs the script automatically at specified time intervals. This ensures that new user accounts are automatically activated and available for use shortly after they are created.



Important

This procedure assumes that the new user accounts do not require validation before the script adds them to IdM. For example, validation is not required when the users have already been validated by the owner of the external provisioning system.

It is sufficient to enable the activation process on only one of your IdM servers.

1. Generate a keytab file for the activation account:

```
# ipa-getkeytab -s example.com -p "activator" -k /etc/krb5.ipa-activation.keytab
```

If you want to enable the activation process on more than one IdM server, generate the keytab file on one server only. Then copy the keytab file to the other servers.

2. Create a script, **/usr/local/sbin/ipa-activate-all**, with the following contents to activate all users:

```
#!/bin/bash

kinit -k -i activator

ipa stageuser-find --all --raw | grep " uid:" | cut -d ":" -f 2 |
while read uid; do ipa stageuser-activate ${uid}; done
```

3. Edit the permissions and ownership for the **ipa-activate-all** script to make it executable:

```
# chmod 755 /usr/local/sbin/ipa-activate-all
# chown root:root /usr/local/sbin/ipa-activate-all
```

4. Create a **systemd** unit file, **/etc/systemd/system/ipa-activate-all.service**, with the following contents:

```
[Unit]
Description=Scan IdM every minute for any stage users that must be activated

[Service]
Environment=KRB5_CLIENT_KTNAME=/etc/krb5.ipa-activation.keytab
Environment=KRB5CCNAME=FILE:/tmp/krb5cc_ipa-activate-all
ExecStart=/usr/local/sbin/ipa-activate-all
```

5. Create a **systemd** timer, **/etc/systemd/system/ipa-activate-all.timer**, with the following contents:

```
[Unit]
```

```
Description=Scan IdM every minute for any stage users that must be activated
[Timer]
OnBootSec=15min
OnUnitActiveSec=1min

[Install]
WantedBy=multi-user.target
```

6. Enable `ipa-activate-all.timer`:

```
# systemctl enable ipa-activate-all.timer
```

Additional resources:

- » For more information on `systemd` unit files, see the [Managing Services with systemd Unit Files](#) chapter of the *System Administrator's Guide*.

8.6.3. Configuring the LDAP Provider of the External Provisioning System to Manage the IdM Identities

Outcome: Using templates for various user and group management operations, the LDAP provider of your provisioning system is configured to manage IdM user accounts. For example, you can configure the system to deactivate a user account after the employee has left the company.

Managing User Accounts Using LDAP

You can add new user entries, modify existing entries, move users between different lifecycle states, or delete users by editing the underlying Directory Server database. To edit the database, use the `ldapmodify` utility.

The following LDIF-formatted templates provide information on what attributes to modify using `ldapmodify`. For detailed example procedures, see [Example 8.2, “Adding a Stage User with ldapmodify”](#) and [Example 8.3, “Preserving a User with ldapmodify”](#).

Adding a new stage user

Adding a user with UID and GID automatically assigned:

```
dn: uid=user_login,cn=staged
users,cn=accounts,cn=provisioning,dc=example,dc=com
changetype: add
objectClass: top
objectClass: inetorgperson
uid: user_login
sn: surname
givenName: first_name
cn: full_name
```

Adding a user with UID and GID statically assigned:

```
dn: uid=user_login,cn=staged
users,cn=accounts,cn=provisioning,dc=example,dc=com
```

```

changetype: add
objectClass: top
objectClass: person
objectClass: inetorgperson
objectClass: organizationalperson
objectClass: posixaccount
uid: user_login
uidNumber: UID_number
gidNumber: GID_number
sn: surname
givenName: first_name
cn: full_name
homeDirectory: /home/user_login

```

You are not required to specify any IdM object classes when adding stage users. IdM adds these classes automatically after the users are activated.

Note that the distinguished name (DN) of the created entry must start with ***uid=user_login***.

Modifying existing users

Before modifying a user, obtain the user's distinguished name (DN) by searching by the user's login. In the following example, the *user_allowed_to_read* user in the following example is a user allowed to read user and group information, and *password* is this user's password:

```

# ldapsearch -LLL -x -D
"uid=user_allowed_to_read,cn=users,cn=accounts,dc=example,
dc=com" -w "password" -H ldap://server.example.com -b "cn=users,
cn=accounts, dc=example, dc=com" uid=user_login

```

To modify a user's attribute:

```

dn: distinguished_name
changetype: modify
replace: attribute_to_modify
attribute_to_modify: new_value

```

To disable a user:

```

dn: distinguished_name
changetype: modify
replace: nsAccountLock
nsAccountLock: TRUE

```

To enable a user:

```

dn: distinguished_name
changetype: modify
replace: nsAccountLock
nsAccountLock: FALSE

```

To preserve a user:

```
dn: distinguished_name
changetype: modrdn
newrdn: uid=user_login
deleteoldrdn: 0
newsuperior: cn=deleted
users,cn=accounts,cn=provisioning,dc=example
```

Updating the **nssAccountLock** attribute has no effect on stage and preserved users. Even though the update operation completes successfully, the attribute value remains **nssAccountLock: TRUE**.

Creating a new group

To create a new group:

```
dn:
cn=group_distinguished_name,cn=groups,cn=accounts,dc=example,dc=com
changetype: add
objectClass: top
objectClass: ipaobject
objectClass: ipausergroup
objectClass: groupofnames
objectClass: nestedgroup
objectClass: posixgroup
cn: group_name
gidNumber: GID_number
```

Modifying groups

Before modifying a group, obtain the group's distinguished name (DN) by searching by the group's name.

```
# ldapsearch -YGSSAPI -H ldap://server.example.com -b
"cn=groups,cn=accounts,dc=example,dc=com" "cn=group_name"
```

To delete an existing group:

```
dn: group_distinguished_name
changetype: delete
```

To add a member to a group:

```
dn: group_distinguished_name
changetype: modify
add: member
member: uid=user_login,cn=users,cn=accounts,dc=example,dc=com
```

To remove a member from a group:

```
dn: distinguished_name
changetype: modify
delete: member
member: uid=user_login,cn=users,cn=accounts,dc=example,dc=com
```

Do not add stage or preserved users to groups. Even though the update operation completes successfully, the users will not be updated as members of the group. Only active users can belong to groups.

Example 8.2. Adding a Stage User with `ldapmodify`

To add a new `stageuser` user using the standard `inetorgperson` object class:

1. Use `ldapmodify` to add the user.

```
# ldapmodify -Y GSSAPI
SASL/GSSAPI authentication started
SASL username: admin@EXAMPLE
SASL SSF: 56
SASL data security layer installed.
dn: uid=stageuser,cn=staged
users,cn=accounts,cn=provisioning,dc=example
changetype: add
objectClass: top
objectClass: inetorgperson
cn: Stage
sn: User

adding new entry "uid=stageuser,cn=staged
users,cn=accounts,cn=provisioning,dc=example"
```

2. Consider validating the contents of the stage entry to make sure your provisioning system added all required POSIX attributes and the stage entry is ready to be activated. To display the new stage user's LDAP attributes using the `ipa stageuser-show --all --raw` command. Note that the user is explicitly disabled by the `nsaccountlock` attribute:

```
$ ipa stageuser-show stageuser --all --raw
dn: uid=stageuser,cn=staged
users,cn=accounts,cn=provisioning,dc=example
uid: stageuser
sn: User
cn: Stage
has_password: FALSE
has_keytab: FALSE
nsaccountlock: TRUE
objectClass: top
objectClass: inetorgperson
objectClass: organizationalPerson
objectClass: person
```

Example 8.3. Preserving a User with `ldapmodify`

To preserve `user` by using the LDAP `modrdn` operation:

1. Use the `ldapmodify` utility to modify the user entry.

```
$ ldapmodify -Y GSSAPI
SASL/GSSAPI authentication started
```

```
SASL username: admin@EXAMPLE
SASL SSF: 56
SASL data security layer installed.
dn: uid=user1,cn=users,cn=accounts,dc=example
changetype: modrdn
newrdn: uid=user1
deleteoldrdn: 0
newsuperior: cn=deleted
users,cn=accounts,cn=provisioning,dc=example

modifying rdn of entry
"uid=user1,cn=users,cn=accounts,dc=example"
```

2. Optionally, verify the user has been preserved by listing all preserved users.

```
$ ipa user-find --preserved=true
-----
1 user matched
-----
User login: user1
First name: first_name
Last name: last_name
...
-----
Number of entries returned 1
-----
```

Chapter 9. User Authentication

This chapter describes managing user authentication mechanisms, including information on how to manage users' passwords, SSH keys, and certificates, or how to configure one-time password (OTP) and smart card authentication.

9.1. Managing Public SSH Keys for Users

Identity Management allows you to upload a public SSH key to a user entry. The user who has access to the corresponding private SSH key can use `ssh` to log into an IdM machine without using Kerberos credentials. If `pam_krb5` is configured properly or if SSSD is used as the IdM server's identity provider, the user also receives a Kerberos ticket-granting ticket (TGT) after login; see [Section 6.2, “Obtaining Kerberos Tickets Automatically”](#) for more details.

Note that users can still authenticate by providing their Kerberos credentials if they are logging in from a machine where their private SSH key file is not available.

Caching and Retrieving SSH Keys Automatically

During an IdM server or client installation, SSSD is automatically configured on the machine to cache and retrieve user and host SSH keys. This allows IdM to serve as a universal and centralized repository of SSH keys.

If the server or client was not configured during installation, you can configure SSSD on the machine manually. For information on how to do this, see the [System-Level Authentication Guide](#). Note that caching SSH keys by SSSD requires administrative privileges on the local machines.

SSH Key Format Requirements

IdM accepts the following two SSH key formats:

OpenSSH-style key

See [RFC 4716](#) for more details about this format.

Raw RFC 4253-style key

See [RFC 4253](#) for more details about this format.

Note that IdM automatically converts RFC 4253-style keys into OpenSSH-style keys before saving them into the IdM LDAP server.

A key file, such as `id_rsa.pub`, consists of three parts: the key type, the key itself, and an additional comment or identifier. In the following example, the key type is RSA and the comment associates the key with the `client.example.com` host name:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQADMM4xPu54Kf2dx7C4Ta2F7vnIzuL1i6P21TTKnISkj
FuA+r
qW06588e7v14Im4VejwnNk352gp49A62qSV0zp8IKA9xdtyRmHYCTUvmkcyspZvFRI713zfR
KQVFyJ0qHmW/m
dCmak7QBxYou2ELSPhH3pe8MYTQIulKDSu5Zbsrqedg1VGkJxf7mDnCSPNWWzAY9AFB9Lmd
2m2xZmNgVAQEQ
nZXNMaiIroLD/51rmMSkJGHGb1068kEq9Z client.example.com
```

When uploading a key to IdM, you can either upload all three key parts, or only the key itself. If you only upload the key itself, IdM automatically identifies the key type, such as RSA or DSA, from the uploaded key.

9.1.1. Generating an SSH Key

You can generate an SSH key using the OpenSSH **ssh-keygen** utility. The utility displays information about the location of the public key. For example:

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

To upload an SSH key for a user, use the public key string stored in the displayed file.

9.1.2. Uploading User SSH Keys

Uploading User SSH Keys in the Web UI

1. Select **Identity** → **Users**.
2. Click the name of the user to edit.
3. Under the **Settings** tab in the **Account Settings** area, click **SSH public keys**: **Add**.

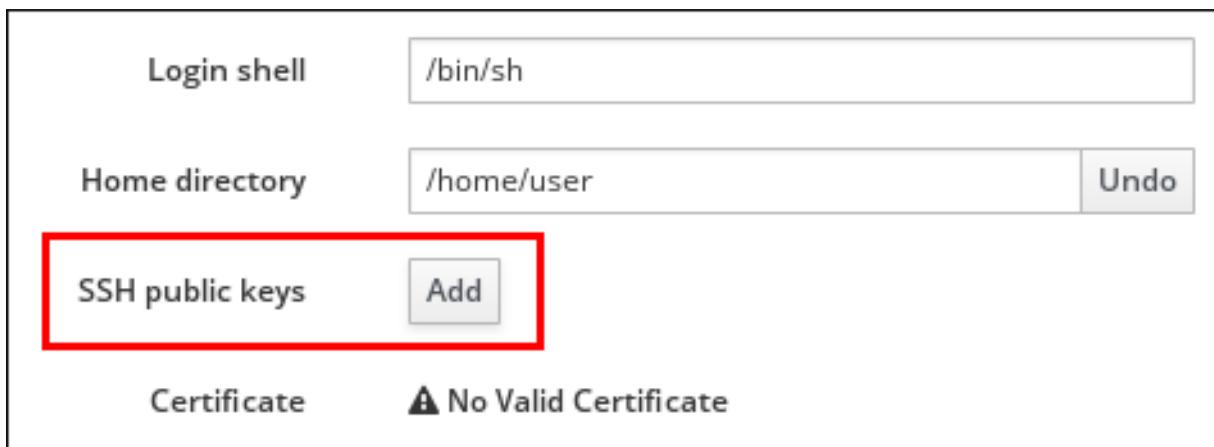


Figure 9.1. SSH public keys in the Account Settings

4. Paste in the base64-encoded public key string, and click **Set**.



Figure 9.2. Pasting in the Public Key

5. Click **Save** at the top of the page.

Uploading User SSH Keys from the Command Line

Use the **ipa user-mod** command and pass the base64-encoded public key string using the **--sshpubkey** option.

For example, to upload the key type, the key itself, and the host name identifier:

```
$ ipa user-mod user --sshpubkey="ssh-rsa AAAAB3Nza...SNc5dv==  
client.example.com"
```

To upload multiple keys, use **--sshpubkey** multiple times. For example, to upload two SSH keys:

```
--sshpubkey="AAAAB3Nza...SNc5dv==" --sshpubkey="RjlzYQo...ZEt0TAo="
```



Note

Instead of pasting the key string manually into the command line, you can use command redirection and point to the file containing the key. For example:

```
$ ipa user-mod user --sshpubkey="$(cat ~/.ssh/id_rsa.pub)" --  
sshpubkey="$(cat ~/.ssh/id_rsa2.pub)"
```

9.1.3. Deleting User Keys

Deleting User SSH Keys in the Web UI

1. Select **Identity** → **Users**.
2. Click the name of the user to edit.
3. Under the **Settings** tab in the **Account Settings** area, click **Delete** next to the key you want to remove.

Home directory	/home/user
SSH public keys	3B:A1:D7:94:33:B3:1E:FD:A2:4A:81:65:FD:1C:78:56 (ssh-rsa)
<input type="button" value="Show/Set key"/> <input style="border: 2px solid red;" type="button" value="Delete"/>	
<input type="button" value="Add"/>	

Figure 9.3. Deleting User SSH Public Key

4. Click **Save** at the top of the page.

Deleting User SSH Keys from the Command Line

To delete all SSH keys assigned to a user account, add the **--sshpubkey** option to the **ipa user-mod** command without specifying any key:

```
$ ipa user-mod user --sshpubkey=
```

If you only want to delete a specific SSH key or keys, use the **--sshpubkey** option to specify the key or keys you want to keep.

9.2. Defining User Authentication Methods

The administrator can control what authentication methods are available for users. IdM supports the following authentication methods:

- password authentication
- RADIUS proxy server authentication; for information on configuring a RADIUS server for OTP validation, see [Section 9.5.7, “Migrating from a Proprietary OTP Solution”](#)

- two-factor authentication (password + OTP); for more information, see [Section 9.5, “One-Time Passwords”](#)

If you set multiple methods at once, either one of them will be sufficient for successful authentication, with minor exceptions depending on whether users authenticate against IdM over Kerberos or LDAP:

Password + two-factor authentication

If you configure both password and two-factor authentication, only LDAP will allow authentication with either one of the authentication methods. Kerberos will still enforce authentication with both password and OTP.

To enforce two-factor authentication for a user, use Kerberos from the application that integrates with IdM.

RADIUS + another authentication method

If you configure RADIUS together with another authentication method, Kerberos will always use RADIUS, but LDAP will not. LDAP only recognizes the password and two-factor authentication methods.

If you use an external two-factor authentication provider, use Kerberos from your applications. If you want to let users authenticate with a password only, use LDAP. It is recommended that the applications leverage Apache modules and SSSD, which allows to configure either Kerberos or LDAP.

Global and User-specific Authentication Methods

Authentication methods can be set globally for all users or individually on a per-user basis. By default, user-specific authentication method settings take precedence over global settings. If no authentication method is set for a user, the user will automatically use the globally-defined authentication methods.

You can disable per-user authentication method settings for any user. This ensures IdM will ignore the per-user settings and always apply the global settings for the user.

Defining User Authentication Methods in the Web UI

To set authentication methods globally for all users:

- Select **IPA Server → Configuration**.
- In the **User Options** area, select the required **Default user authentication types**.

Default user authentication types ⓘ	<input type="checkbox"/> Disable per-user override <input type="checkbox"/> Password <input type="checkbox"/> Radius <input checked="" type="checkbox"/> Two factor authentication (password + OTP)
---	--

Figure 9.4. User Authentication Methods

To ensure the global settings are not overridden with per-user settings, select **Disable per-user override**. If **Disable per-user override** is unselected, authentication methods configured per user take precedence over the global settings.

To set authentication methods individually on a per-user basis:

1. Select **Identity** → **Users**, and click the name of the user to edit.
2. In the **Account Settings** area, select the required **User authentication types**.

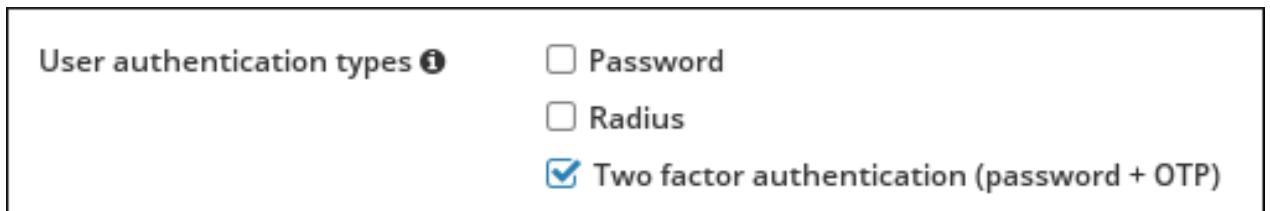


Figure 9.5. User Authentication Methods

Defining User Authentication Methods from the Command Line

To set authentication methods globally for all users, run the **ipa config-mod** command and add the **--user-auth-type** option. To set an authentication method, use the **password**, **radius**, and **otp** values. For example, to set the global authentication method to two-factor authentication:

```
$ ipa config-mod --user-auth-type=otp
```

To disable per-user overrides and ensure the global settings are not overridden with per-user settings, add **--user-auth-type=disabled** to **ipa config-mod**. If you do not use **--user-auth-type=disabled**, authentication methods configured per user take precedence over the global settings. For example, to configure the password authentication method globally and to disable per-user overrides:

```
$ ipa config-mod --user-auth-type=password --user-auth-type=disabled
```

To set authentication methods individually for a specified user, run the **ipa user-mod** command and add the **--user-auth-type** option. For example, to set that **user** will be required to authenticate with their personal password:

```
$ ipa user-mod user --user-auth-type=password
```

To set multiple authentication methods, add **--user-auth-type** multiple times. For example, to configure both password and two-factor authentication globally for all users:

```
$ ipa config-mod --user-auth-type=otp --user-auth-type=password
```

9.3. Changing and Resetting User Passwords

Regular users without the permission to change other users' passwords can only change their own personal password. Personal passwords set by regular users must meet the IdM password policies. For more information about configuring these policies, see [Chapter 22, Defining Password Policies](#).

The administrator and users with password change rights can set initial passwords for new users as well as reset passwords for existing users. These passwords are not required to meet the IdM password policies and will expire when the user first logs in using the password. When this happens, IdM asks the user to change the expired password immediately. This ensures that user passwords always remain confidential.

Note

The LDAP Directory Manager (DM) user can change user passwords using LDAP tools. The new password can override any IdM password policies. Passwords set by DM do not expire after the first login.

Changing Your Own Personal Password in the Web UI

1. In the top right corner, click **User name** → **Change password**.

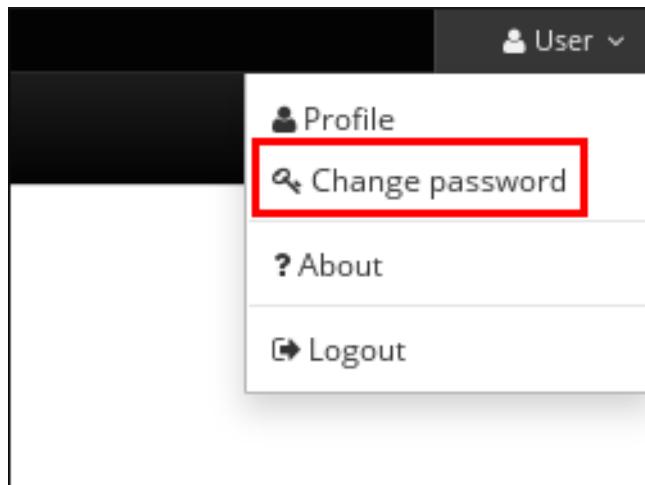


Figure 9.6. Resetting Password

2. Enter the new password.

Resetting Another User's Password in the Web UI

1. Select **Identity** → **Users**.
2. Click the name of the user to edit.
3. Click **Actions** → **Reset password**.

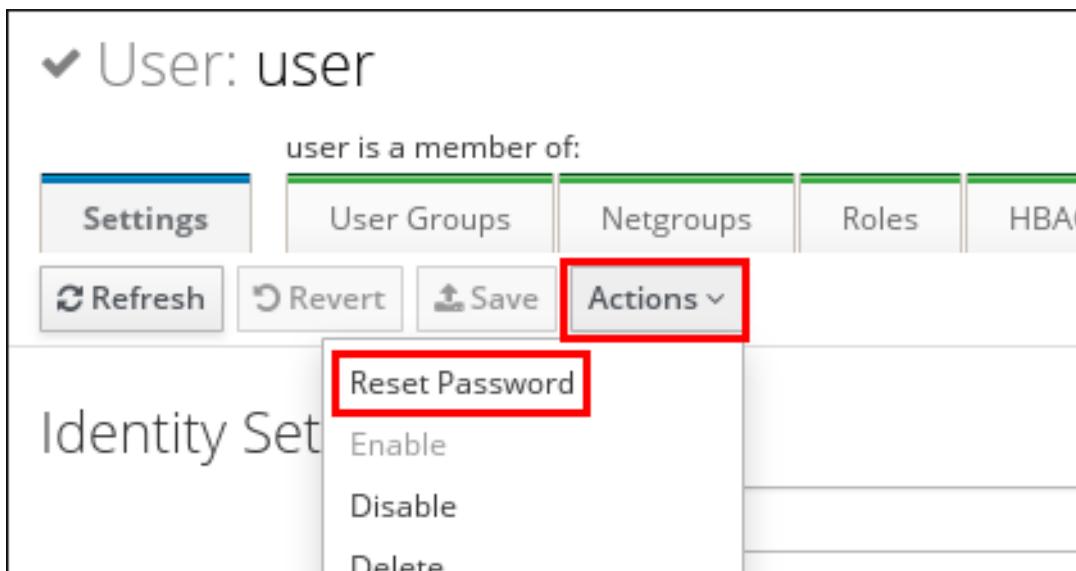


Figure 9.7. Resetting Password

4. Enter the new password, and click **Reset Password**.



Figure 9.8. Confirming New Password

Changing or Resetting Another User's Password from the Command Line

To change your own personal password or to change or reset another user's password, add the **--password** option to the **ipa user-mod** command. The command will prompt you for the new password.

```
$ ipa user-mod user --password
Password:
Enter Password again to verify:
-----
Modified user "user"
-----
...
```

9.4. Unlocking User Accounts After Password Failures

If a user attempts to log in using an incorrect password a certain number of times, IdM will lock the user account, which prevents the user from logging in. Note that IdM does not display any warning message that the user account has been locked.

Note

For information on setting the exact number of allowed failed attempts and the duration of the lockout, see [Section 22.6, “Setting Account Lockout Policies”](#).

IdM automatically unlocks the user account after a specified amount of time has passed. Alternatively, the administrator can unlock the user account manually.

Unlocking a User Account Manually

To unlock a user account, use the `ipa user-unlock` command.

```
$ ipa user-unlock user
-----
Unlocked account "user"
-----
```

After this, the user is able to log in again.

Checking the Status of a User Account

To display the number of failed login attempts for a user, use the `ipa user-status` command. If the displayed number exceeds the number of allowed failed login attempts, the user account is locked.

```
$ ipa user-status user
-----
Account disabled: False
-----
Server: example.com
Failed logins: 8
Last successful authentication: 20160229080309Z
Last failed authentication: 20160229080317Z
Time now: 2016-02-29T08:04:46Z
-----
Number of entries returned 1
-----
```

9.5. One-Time Passwords

One-time password (OTP) is a password valid for only one authentication session; it becomes invalid after use. Unlike a traditional static password, OTP generated by an authentication token keeps changing. OTPs are used as part of two-factor authentication: the first step requires the user to authenticate with a traditional password, and the second step prompts for an OTP issued by a recognized token.

Two-factor authentication is considered safer than authentication using a traditional password alone. Even if a potential intruder intercepts the OTP during login, the intercepted OTP will already be invalid by that point because it can only be used for successful authentication once.

Hardware and Software Tokens

Both hardware and software tokens are used for issuing OTPs. A hardware token is stored on a dedicated physical device. A software token is typically stored on the user's mobile device, such as a smartphone or a tablet.

Hardware tokens are often managed by the administrator. However, some hardware tokens, such as the YubiKey token, are typically user-managed. Administrators can purchase hardware tokens in bulk and then distribute them to the users.

Similarly, software tokens are often, but not always, managed by the user. For example, companies that issue mobile devices to their employees can use administrator-managed software tokens.

9.5.1. One-Time Passwords in Identity Management



Important

The IdM solution for OTP authentication is only supported for clients running Red Hat Enterprise Linux 7.1 or later.



Warning

The following security and other limitations currently relate to the IdM native OTP support:

- ✖ The most important security limitation is the potential vulnerability to replay attacks across the system. Replication is asynchronous, and an OTP code can therefore be reused during the replication period. A user might be able to log on to two servers at the same time. However, this vulnerability is usually difficult to exploit due to comprehensive encryption.
- ✖ It is not possible to obtain a ticket-granting ticket (TGT) via a client that does not support OTP authentication. This might affect certain use cases, such as authentication using the `mod_auth_kerb` module or the Generic Security Services API (GSSAPI).

Identity Management supports both user-managed and administrator-managed OTP tokens:

User-managed tokens

Users have full control over user-managed tokens in Identity Management: they are allowed to create, edit, or delete their tokens.

Administrator-managed tokens

The administrator adds administrator-managed tokens to the users' accounts. Users themselves have read-only access for such tokens: they do not have the permission to manage or modify the tokens and they are not required to configure them in any way.

Note that users cannot delete or deactivate a token if it is their only active token at the moment. Similarly, the administrator cannot delete or deactivate the last remaining active token assigned to a user.

Supported OTP Algorithms

Identity Management supports the following two standard OTP mechanisms:

- » The HMAC-Based One-Time Password (HOTP) algorithm is based on a counter. HMAC stands for Hashed Message Authentication Code.
- » The Time-Based One-Time Password (TOTP) algorithm is an extension of HOTP to support time-based moving factor.

Offline Authentication and GNOME Keyring Service

IdM supports offline OTP authentication and also integrates OTP authentication with the GNOME Keyring service. Note that both offline authentication and GNOME Keyring integration require the user to enter the first and second factors separately:

First factor: *static_password*
Second factor: *one-time_password*

For more information about offline OTP authentication in IdM, see [Section 9.5.6, “Offline Authentication with OTP”](#).

Resynchronizing an OTP Token

See [Section A.4.4, “OTP Token Out of Sync”](#).

9.5.2. Enabling OTP Authentication in IdM

To enable OTP authentication, configure two-factor authentication either globally or for individual users. For more information, see [Section 9.2, “Defining User Authentication Methods”](#).

9.5.3. Adding a User-Managed Software Token

1. Log in with your standard password.
2. Make sure the **FreeOTP Authenticator** application is installed on your mobile device. To download **FreeOTP Authenticator**, see [the FreeOTP source page](#).
3. Create the software token in the IdM web UI or from the command line.
 - » To create the token in the web UI, click **Add** under the **OTP tokens** tab. If you are logged-in as the administrator, the **OTP Tokens tab** is accessible through the **Authentication** tab.

The screenshot shows the IPA web interface for managing OTP tokens. At the top, there are tabs for 'Users' and 'OTP Tokens'. The 'OTP Tokens' tab is selected, displaying the title 'OTP Tokens'. Below the title is a search bar with a magnifying glass icon. To the right of the search bar are buttons for 'Refresh', 'Delete', and 'Add'. The 'Add' button is highlighted with a red box. The main area contains a table with columns: Unique ID, Owner, Status, and Delete. One entry is listed: Unique ID 230c1a3e-85b1-4310-9c91-a9461f91ca70, Owner employee, Status Enabled. Below the table, it says 'Showing 1 to 1 of 1 entries.'

Figure 9.9. Adding an OTP Token for a User

- To create the token from the command line, run the **ipa otptoken-add** command.

```
$ ipa otptoken-add
-----
Added OTP token ""

-----
Unique ID: 7060091b-4e40-47fd-8354-cb32fec548a
Type: TOTP
...
```

For more information about **ipa otptoken-add**, run the command with the **--help** option added.

- A QR code is displayed in the web UI or on the command line. Scan the QR code with **FreeOTP Authenticator** to provision the token to the mobile device.

9.5.4. Adding a User-Managed YubiKey Hardware Token

A programmable hardware token, such as a YubiKey token, can only be added from the command line. To add a YubiKey hardware token as the user owning the token:

- Log in with your standard password.
- Insert your YubiKey token.
- Run the **ipa otptoken-add-yubikey** command.
 - If the YubiKey has an empty slot available, the command will select the empty slot automatically.
 - If no empty slot is available, you must select a slot manually using the **--slot** option. For example:

```
$ ipa otptoken-add-yubikey --slot=2
```

Note that this overwrites the selected slot.

9.5.5. Adding a Token for a User as the Administrator

To add a software token as the administrator:

1. Make sure you are logged-in as the administrator.
2. Make sure the **FreeOTP Authenticator** application is installed on the mobile device. To download **FreeOTP Authenticator**, see [the FreeOTP source page](#).
3. Create the software token in the IdM web UI or from the command line.
 - To create the token in the web UI, select **Authentication** → **OTP Tokens** and click **Add** at the top of the list of OTP tokens. In the **Add OTP Token** form, select the owner of the token.

Unique ID	Token ID
Description	User's Token
Owner	user
Validity start	2016-02-03 00 : 00 UTC

Figure 9.10. Adding an Administrator-Managed Software Token

- To create the token from the command line, run the **ipa otptoken-add** command with the **--owner** option. For example:

```
$ ipa otpoken-add --owner=user
-----
Added OTP token ""

-----
Unique ID: 5303baa8-08f9-464e-a74d-3b38de1c041d
Type: TOTP
...
```

4. A QR code is displayed in the web UI or on the command line. Scan the QR code with **FreeOTP Authenticator** to provision the token to the mobile device.

To add a programmable hardware token, such as a YubiKey token, as the administrator:

1. Make sure you are logged-in as the administrator.
2. Insert the YubiKey token.
3. Run the **ipa otpoken-add-yubikey** command with the **--owner** option. For example:

```
$ ipa otpoken-add-yubikey --owner=user
```

9.5.6. Offline Authentication with OTP

IdM supports offline OTP authentication. However, to be able to log in offline, the user must first authenticate when the system is online by entering the static password and OTP separately:

First factor: *static_password*
 Second factor: *one-time_password*

If both passwords are entered separately like this when logging in online, the user will subsequently be able to authenticate even if the central authentication server is unavailable. Note that IdM only prompts for the first-factor traditional static password when the user authenticates offline.

IdM also supports entering both the static password and OTP together in one string in the **First factor** prompt. However, note that this is not compatible with offline OTP authentication. If the user enters both factors in a single prompt, IdM will always have to contact the central authentication server when authenticating, which requires the system to be online.



Important

If you use OTP authentication on devices that also operate offline, such as laptops, Red Hat recommends to enter the static password and OTP separately to make sure offline authentication will be available. Otherwise, IdM will not allow you to log in after the system goes offline.

If you want to benefit from OTP offline authentication, apart from entering the static and OTP passwords separately, also make sure to meet the following conditions:

- ▶ The **cache_credentials** option in the `/etc/sssd/sssd.conf` file is set to **True**, which enables caching the first factor password.
- ▶ The first-factor static password meets the password length requirement defined in the **cache_credentials_minimal_first_factor_length** option set in `/etc/sssd/sssd.conf`. The default minimal length is 8 characters. For more information about the option, see the `sssd.conf(5)` man page.

Note that even if the **krb5_store_password_if_offline** option is set to **true** in `/etc/sssd/sssd.conf`, SSSD does not attempt to refresh the Kerberos ticket-granting ticket (TGT) when the system goes online again because the OTP might already be invalid at that point. To obtain a TGT in this situation, the user must authenticate again using both factors.

9.5.7. Migrating from a Proprietary OTP Solution

To enable migrating a large deployment from a proprietary OTP solution to the IdM-native OTP solution, IdM offers a way to offload OTP validation to a third-party RADIUS server for a subset of users. The administrator creates a set of RADIUS proxies where each proxy can contain multiple individual RADIUS servers. The administrator then assigns one of these proxy sets to a user. As long as the user has a RADIUS proxy set assigned, IdM bypasses all other authentication mechanisms.



Note

IdM does not provide any token management or synchronization support for tokens in the third-party system.

To configure a RADIUS server for OTP validation and to add a user to the proxy server:

1. Make sure that the `radius` user authentication method is enabled. See [Section 9.2, “Defining User Authentication Methods”](#).
2. Run the `ipa radiusproxy-add proxy_name` command to add a RADIUS proxy. The command prompts you for the required information.
3. Run the `ipa user-mod radiususer --radius=proxy_name` command to assign a user to the added proxy.
4. If required, configure the user name to be sent to RADIUS by running the `ipa user-mod radiususer --radius-username=radius_user` command.

After this, the user OTP authentication will now be processed through the RADIUS proxy server.

When the user is ready to be migrated to the IdM native OTP system, you can simply remove the RADIUS proxy assignment for the user.

9.6. Smart Cards

Authentication based on smart cards is an alternative to password-based authentication. User credentials are stored on the smart card, and special software and hardware is then used to access them. In order to authenticate using a smart card, the user must place the smart card into a smart card reader and then supply the PIN code for the smart card.

9.6.1. Smart Card Authentication in Identity Management

Red Hat Identity Management (IdM) supports two smart card-based authentication options: local authentication and remote `ssh` authentication. Both require the System Security Services Daemon (SSSD) to be running on the IdM client.

With SSSD-based smart card authentication configured, the system prompts for the smart card PIN code after the user attempts to log in. The user is successfully authenticated if the supplied PIN is correct, the certificate on the smart card is valid and belongs to the user attempting to log in, and other configurable criteria are met.

Local authentication

IdM supports smart card authentication at a text or graphical console, such as the Gnome Display Manager (GDM), as well as authentication using local authentication services like `su` or `sudo`.

Remote authentication with ssh

Certificates on a smart card are stored together with the PIN-protected private key; this key is used for the `ssh` authentication. On the client side, the `ssh` client program in Red Hat Enterprise Linux accesses the smart card; on the server side, only the public key from the certificate is then used for `ssh` access.

IdM only supports the above-mentioned local authentication services and `ssh` for smart card authentication. Other services, such as FTP, are not supported.

Note that to be able to authenticate using smart cards, the `pam_cert_auth` option must be set to `True` in the `[pam]` section of the `/etc/sssd/sssd.conf` file.

Smart Card and Smart Card Reader Support in Identity Management

If your smart card is supported by the `coolkey` package, the PKCS #11 module required by the smart card reader is already present in the central `/etc/pki/nssdb/` NSS database. If your smart card reader is not supported, you must add the required PKCS #11 module manually using the `modutil` utility. For example:

```
modutil -dbdir /etc/pki/nssdb -add "My PKCS#11 module" -libfile libmypkcs11.so
```

...

Module "My PKCS#11 Module" added to database.

For detailed information on using `modutil`, see the `modutil(1)` man page.

9.6.1.1. Configuring Smart Card Authentication on an IdM Client

1. Place the smart card into the reader.
2. If you have the smart card certificate saved in a file, you can use the base64-encoded certificate string from the file for the next step.

Alternatively, to be able to extract the certificate from the smart card later, use the `certutil` utility to list the certificates on the smart card. In the output, locate the certificate you want to use for authentication and note its nickname.

\$ certutil -L -d /etc/pki/nssdb -h all	
Certificate Nickname	Trust
Attributes	
SSL,S/MIME,JAR/XPI	
my_certificate	CT,C,C

3. Assign the certificate to an IdM user using the `ipa user-mod` or `ipa user-add-cert` commands. For example, using `ipa user-mod`:

- » If you have the smart card certificate saved in a file, pass the whole certificate to the command using the `--certificate` option. For example:

```
$ ipa user-mod user --certificate=MIIEkjC ...
-----
User "user" modified
-----
User login: user
...
Certificate: MIIEkjC ...
```

- » If you determined the certificate nickname in the previous step, you can use command redirection to add the certificate to the user entry without copy-pasting the certificate contents into the command line.

The following command extracts the raw DER encoded form of the certificate and passes it to the **base64** utility to re-encode it into a Base64 string. This automatically strips the headers, footers, and newline characters from the PEM certificate:

```
$ ipa user-mod user --certificate="$(certutil -L -d /etc/pki/nssdb -n 'my_certificate' -r | base64 -w 0)"  
-----  
Modified user "user"  
-----  
User login: user  
...  
Certificate: MIIEkjC ...
```

The **base64** utility is part of the *coreutils* package.

For information on using **ipa user-add-cert**, see [Section 17.2, “Managing Certificates Issued by External CAs”](#).

After this, the smart card certificate is mapped to the user entry, which enables the user to use the smart card for local authentication on an IdM client or for logging in using **ssh**. Note that when logging in with **ssh**, the user must specify the following information:

- » the path to the smart card reader module you want to use
- » the user that you want to log in as
- » the name of the IdM client to which you want to log in

For example:

```
$ ssh -I /usr/lib/libmypkcs11.so -l user@example.com host.example.com  
Enter PIN for 'Smart Card':
```

9.7. User Certificates

For information on user certificates, see [Chapter 17, Managing Certificates for Users, Hosts, and Services](#).

Chapter 10. User Groups

10.1. 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 [2] assigned to NIS groups or other system groups.

10.1.1. 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
-----
```

10.1.2. 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 27, Defining Automatic Group Membership for Users and Hosts](#)).

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

10.1.3. 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 ~]# systemctl restart dirsrv.target
```

Managed Entries Plug-in instances can be re-enabled with the **enable** option.

10.2. 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.

10.2.1. 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. 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 27, 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.

10.2.2. 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 10.1. Default Identity Management Group Object Classes

Description	Object Classes
IdM object classes	ipaobject ipausergroup nestedgroup
Group object classes	groupofnames

10.2.2.1. Creating User Groups

10.2.2.1.1. With the Web UI

1. Open the **Identity** tab, and select the **User Groups** subtab.

2. Click **Add** at the top of the groups list.

User Groups			
Search		Actions	
	Group name	GID	Description
<input type="checkbox"/>	admins	1120000000	Account administrators group
<input type="checkbox"/>	editors	1120000002	Limited admins who can edit other users

Figure 10.1. List of User Groups

3. Enter information for the group.

Add User Group

Group name *

Description

Group Type

Normal External POSIX

GID

* Required field

Figure 10.2. Adding a User 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 10.2.2.2.1, “With the Web UI \(Group Page\)”\).](#)

10.2.2.1.2. 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

If you add a new group entry without specifying a custom GID number, IdM automatically assigns a number that is next available in the ID range. This means that GIDs are always unique. For more information about ID ranges, see [Chapter 11, Unique UID and GID Number Assignments](#).

When you specify a custom ID number, the server does not validate if the custom ID number is unique. Due to this, multiple entries might have the same ID number assigned. Even though this behavior is expected, it is not recommended to have multiple entries with the same ID number.



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.

10.2.2.2. Adding Group Members

10.2.2.2.1. 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.

User Groups			
Search		Refresh	Delete
	Group name	GID	Description
<input type="checkbox"/>	admins	1120000000	Account administrators group
<input type="checkbox"/>	editors	1120000002	Limited admins who can edit other users

Figure 10.3. Group List

3. Click **Add** at the top of the task area.

User Group: editors

editors members:

Users	User Groups	External	Settings
-------	-------------	----------	----------

editors is a member of:

User Groups	Netgroups	Roles	HBAC Rules	Sudo Rules
-------------	-----------	-------	------------	------------

+ Add Direct Membership Indirect Membership

<input type="checkbox"/> User login	UID	Email address	Telephone Number	Job Title
-------------------------------------	-----	---------------	------------------	-----------

Figure 10.4. Groups Menu

4. Select the check box by the names of the users to add, and click the right arrow button, >, to move the names to the selection box.

Add Users into User Group editors

Filter available Users

Available		Prospective	
<input type="checkbox"/>	Full name	<input type="button" value=">"/>	<input type="checkbox"/>
<input type="checkbox"/>	Administrator	<input type="button" value="<"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	John Smith	<input type="checkbox"/>	Full name
<input type="checkbox"/>	Manager	<input type="checkbox"/>	Barbara Jensen

Add **Cancel**

Figure 10.5. Adding Users into a User Group

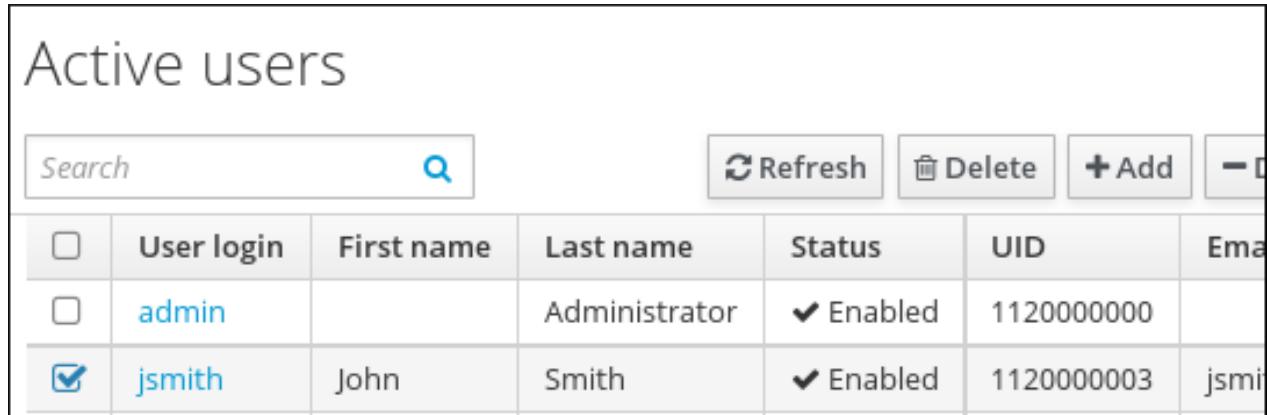
- Click the **Add** button.

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.

10.2.2.2. With the Web UI (User's Page)

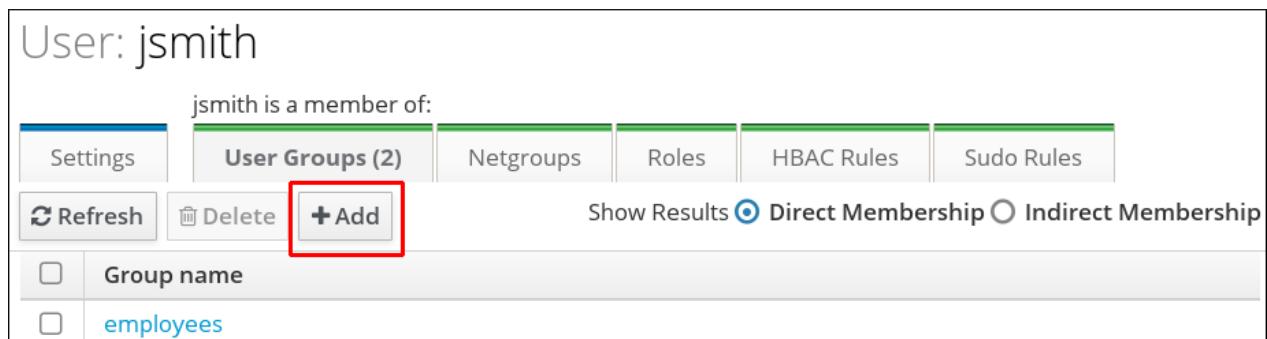
Users can also be added to a group through the user's page.

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



<input type="checkbox"/>	User login	First name	Last name	Status	UID	Email
<input type="checkbox"/>	admin		Administrator	✓ Enabled	1120000000	
<input checked="" type="checkbox"/>	jsmith	John	Smith	✓ Enabled	1120000003	jsmi

- Open the **User Groups** tab on the user entry page.
- Click the **Add** link at the top of the task area.



User: jsmith

jsmith is a member of:

Settings	User Groups (2)	Netgroups	Roles	HBAC Rules	Sudo Rules
----------	-----------------	-----------	-------	------------	------------

+ Add

Show Results Direct Membership Indirect Membership

<input type="checkbox"/> Group name
<input type="checkbox"/> employees

Figure 10.6. Adding User Groups

- Select the check box by the names of the groups for the user to join, and click the right arrow button, >, to move the groups to the selection box.

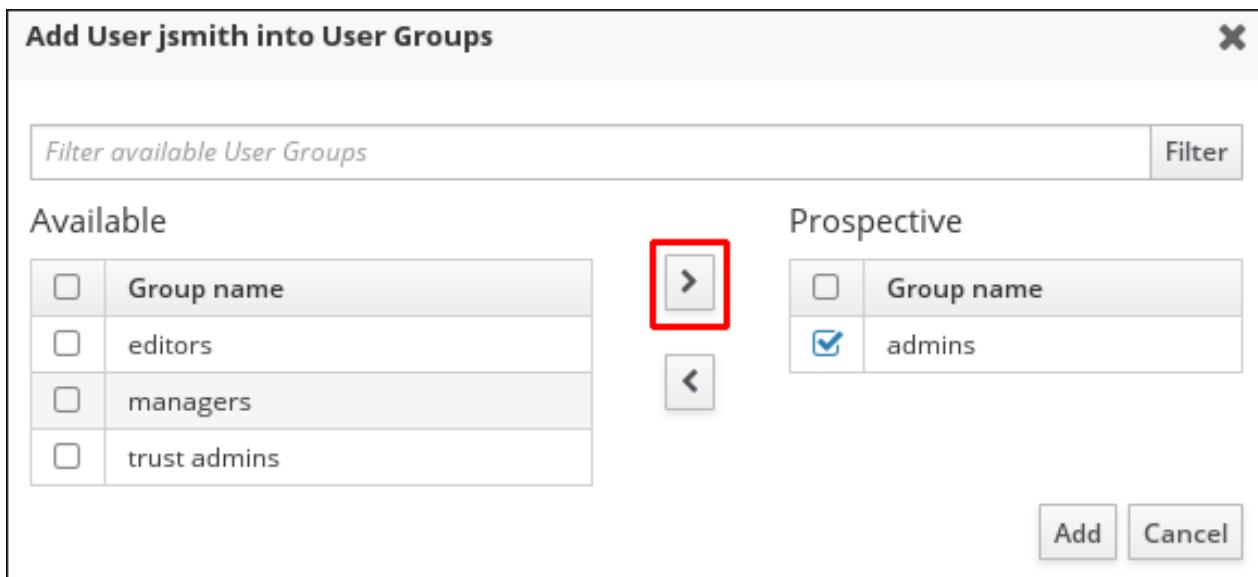


Figure 10.7. Selecting Groups a Member Should be Added to

- Click the **Add** button.

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=qel --groups=dev2
Group name: engineering
Description: for engineers
GID: 387115842
```

```
Member groups: dev,qel,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
-----
```

10.2.2.2.4. 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.

The screenshot shows a user interface for managing a user group named 'editors'. At the top, it says 'User Group: editors' and 'editors members:'. Below this are two sets of tabs: 'Users', 'User Groups', 'External', and 'Settings' in the first set; and 'User Groups', 'Netgroups', 'Roles', 'HBAC Rules', and 'Sudo Rules' in the second set. A red box highlights the 'Show Results' button and the 'Direct Membership' radio button. Below these are search and filter fields for 'User login', 'UID', 'Email address', 'Telephone Number', and 'Job Title'.

Figure 10.8. Indirect and Direct Members

Being able to track indirect members makes it easier to assign group membership properly, without duplicating membership.

10.2.2.3. 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.

10.2.2.3.1. With the Web UI

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Select the check box by the name of the group to delete.

The screenshot shows a table of user groups. The columns are 'Group name', 'GID', and 'Description'. There are three rows: one for 'admins' (GID 1120000000, description 'Account administrators group') and one for 'editors' (GID 1120000002, description 'Limited admins who can edit other users'). The checkbox for 'editors' is checked and highlighted with a red box.

Figure 10.9. Selecting Groups to Be Deleted

3. Click the **Delete** link at the top of the task area.

4. When prompted, confirm the delete action.

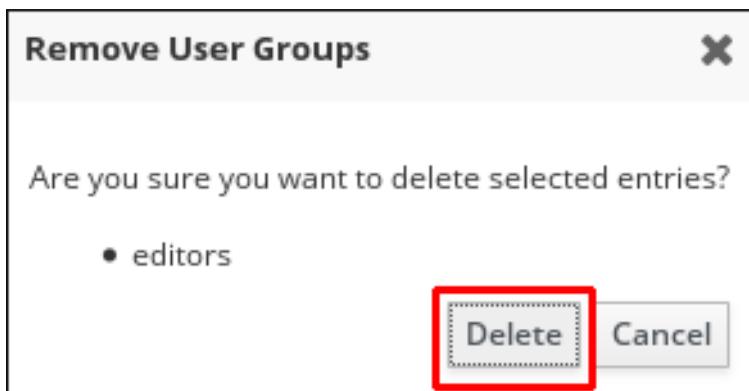


Figure 10.10. Confirming Group Removal

10.2.2.3.2. With the Command Line

The **group-del** command to deletes the specified group. For example:

```
[bjensen@server ~]$ ipa group-del examplegroup
```

10.2.3. 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 12.4, “Specifying Default User and Group Attributes”](#).

10.2.3.1. Setting Search Limits

10.2.3.1.1. 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.

10.2.3.1.2. 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.

10.2.3.1.2.1. With the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Search Options** area.

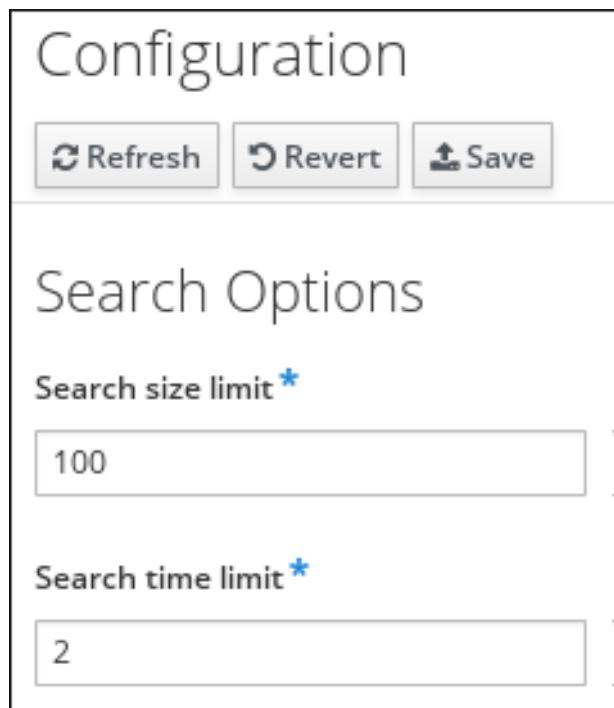


Figure 10.11. Setting the Search Size and Time Limit

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.

10.2.3.1.2.2. 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: 0=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.

10.2.3.1.3. 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
```

10.2.3.2. 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](#).

10.2.3.2.1. 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 10.2, “Default Search Attributes”](#). All search attributes are searched in a user/group search.

Table 10.2. 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 10.2.3.2, “Setting Search Attributes”](#) and [Section 10.2.3.2.3, “Changing Group Search Attributes”](#).

10.2.3.2.2. Changing User Search Attributes

10.2.3.2.2.1. From the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **User Options** area.

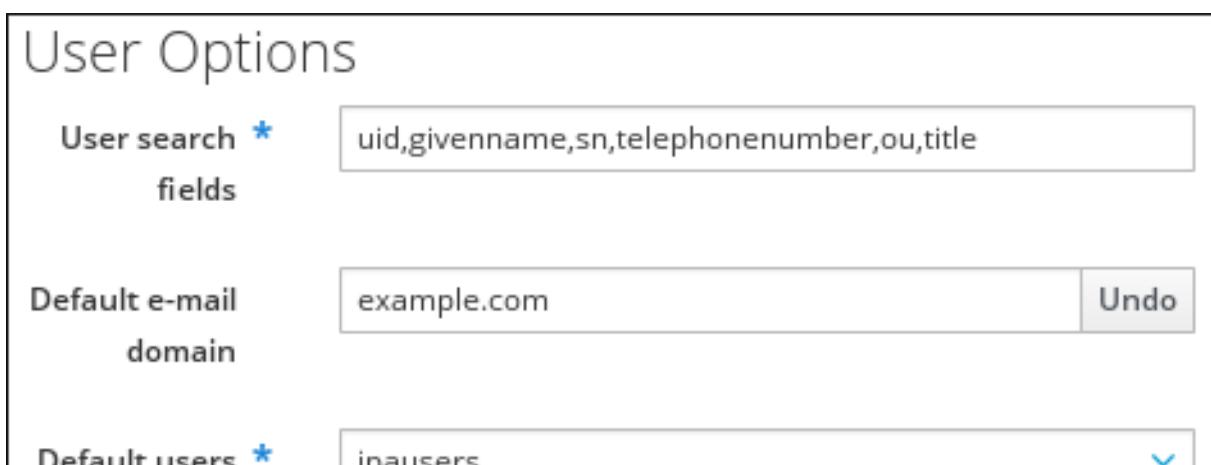


Figure 10.12. User Options Area of the Configuration Subtab

4. Add any additional search attributes, in a comma-separated list, in the **User search fields** field.
5. When the changes are complete, click **Save** at the top of the **Configuration** page.

10.2.3.2.2.2. 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,telephononenumber,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.

10.2.3.2.3. 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](#).

10.2.3.2.3.1. From the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Group Options** area.

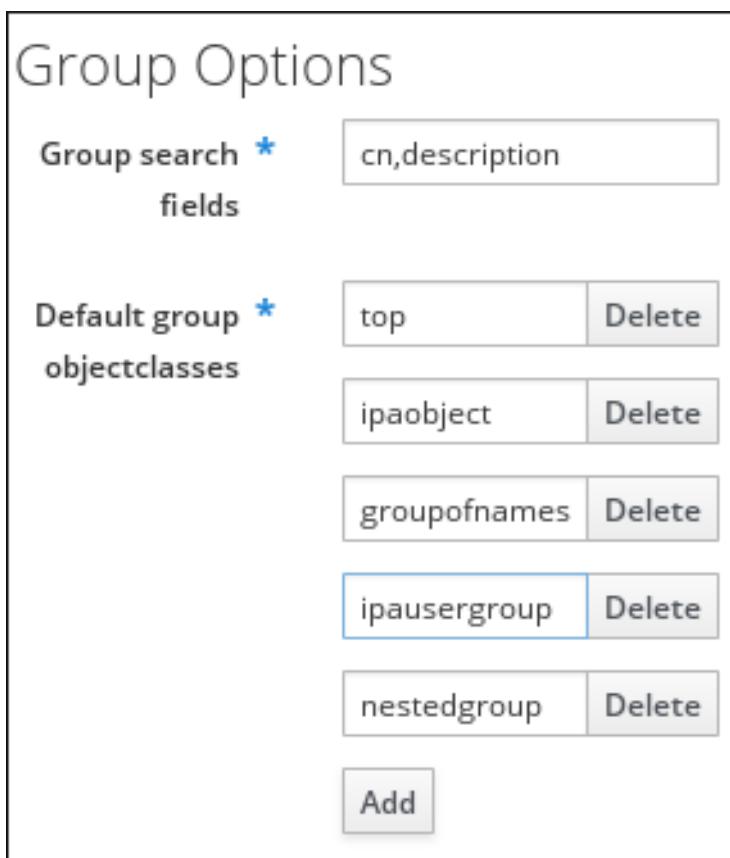


Figure 10.13. Group Options Area of the Configuration Subtab

4. Add any additional search attributes, in a comma-separated list, in the **Group search fields** field.
5. When the changes are complete, click **Save** at the top of the **Configuration** page.

10.2.3.2.3.2. 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.

10.2.3.2.4. 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.

10.2.3.3. 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 10.3, “Common Group Search Options”](#).

Table 10.3. 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-inhbac-rules	Displays only groups which belong to a given host-based access control rule (or which do not belong to the rule, for the --not-in option). There are similar options to display (or not) groups which belong to a specified sudo rule and role.

Option	Criteria Description
--in-groups, --not-in-groups	Displays only groups which belong to another, specified group (or which do not belong to the group, for the --not-in option). There are similar options to display (or not) groups which belong to a specified netgroup.

[2] See [Chapter 11, Unique UID and GID Number Assignments](#) for information on changing GID/UID assignment ranges.

Chapter 11. 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.

11.1. 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](#).

11.2. 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 11.5, “Manual ID Range Extension and Assigning a New ID Range”](#).

11.3. 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 dnanextra-range-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 dnanextra-range-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 dnanextra-range-show masterA.example.com
masterA.example.com: 1001-1500
```

For more information about these two commands, see the `ipa-replica-manage(1)` man page.

11.4. 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 dnanextra-range-show** commands, described in [Section 11.3, “Displaying Currently Assigned ID Ranges”](#).

11.5. 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 UIDs 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 dnarange-show** command described in [Section 11.3, “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 dnarange-set** allows you to define the current ID range for a specified server:

```
# ipa-replica-manage dnarange-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**.

11.6. 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 11.7, “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.

11.7. 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
```

Chapter 12. 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 12.1. Default Identity Management User 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 12.1, “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 12.4, “Specifying Default User and Group Attributes”](#).

Table 12.2. Default Identity Management User Attributes

UI Field	Command-Line Option	Required, Optional, or Default [a]
User login	username	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

UI Field	Command-Line Option	Required, Optional, or Default
User ID number	--uid	Default
Group ID number	--gidnumber	Default
Street address	--street	Optional
City	--city	Optional
State/Province	--state	Optional
Zip code	--postalcode	Optional
Telephone number	--phone	Optional
Mobile telephone number	--mobile	Optional
Pager number	--pager	Optional
Fax number	--fax	Optional
Organizational unit	--orgunit	Optional
Job title	--title	Optional
Manager	--manager	Optional
Car license	--carlicense	Optional
	--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.		

12.1. 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 ([Chapter 12, 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.

12.2. 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.

12.2.1. 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.

The screenshot shows the 'User Options' configuration page. It includes fields for 'User search * fields' (uid,givenname,sn,telephonenumber,ou,title), 'Default e-mail domain' (example.com), and 'Default users *' (inausers). There is also an 'Undo' button next to the domain field.

Figure 12.1. User Options in Server Configuration

5. At the bottom of the users area, click **Add** to include 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.

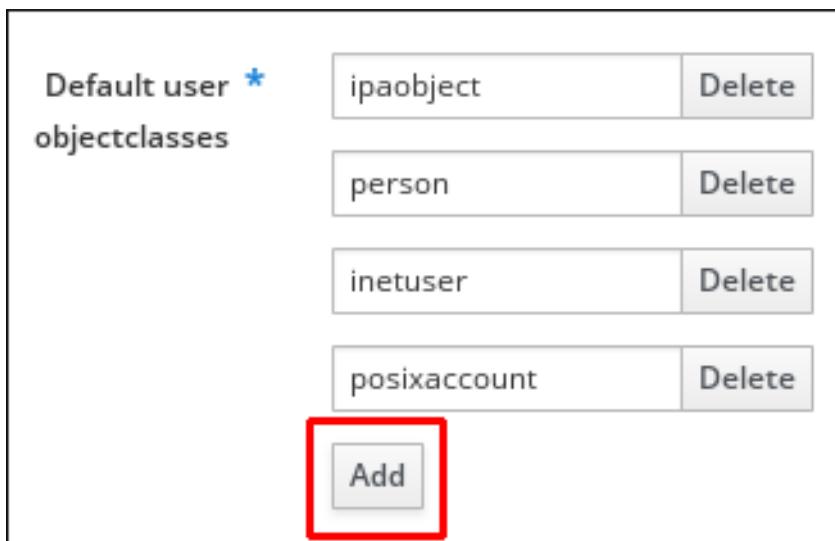


Figure 12.2. Changing Default User Object Classes

- When the changes are complete, click **Save** at the top of the **Configuration** page.

12.2.2. 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 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, inetuser, posixaccount, krbprincipalaux, krbticketpolicyaux, ipaobject, ipasshuser, employeeinfo}
```

12.3. 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.

12.3.1. 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.

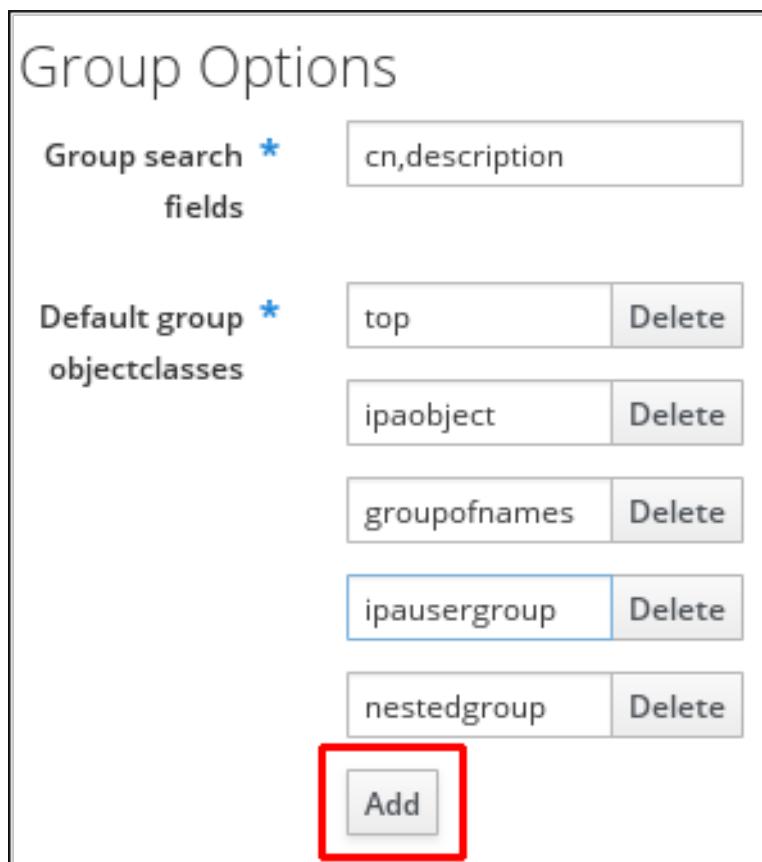


Figure 12.3. Group Options in Server Configuration

5. Click **Add** to include a new field for another object class.



Important

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

6. When the changes are complete, click **Save** at the top of the **Configuration** page.

12.3.2. From the Command Line

1. Add all of the custom schema elements to the 389 Directory Server instance used by Identity Management. Adding schema elements is described in [the schema chapter of the Directory Server Administrator's Guide](#).
2. Add the new object class to the list of object classes added to entries. The option for group object classes is **--groupobjectclasses**.



Important

Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by Identity Management are not included, then subsequent attempts to add an entry will fail with object class violations.

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,ipabject,ipasshuser,employeegroup}
```

12.4. 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 12.3. Default User Parameters

Field	Command-Line Option	Descriptions
Maximum username length	<code>--maxusername</code>	Sets the maximum number of characters for usernames. The default value is eight.

Field	Command-Line Option	Descriptions
Root for home directories	--homedirectory	Sets the default directory to use for user home directories. The default value is /home .
Default shell	--defaultshell	Sets the default shell to use for users. The default value is /bin/sh .
Default user group	--defaultgroup	Sets the default group to which all newly created accounts are added. The default value is ipausers , which is automatically created during the IdM server installation process.
Default e-mail domain	--emaildomain	Sets the email domain to use to create email addresses based on the new accounts. The default is the IdM server domain.
Search time limit	--searchtimelimit	Sets the maximum amount of time, in seconds, to spend on a search before the server returns results.
Search size limit	--searchrecordslimit	Sets the maximum number of records to return in a search.
User search fields	--usersearch	Sets the fields in a user entry that can be used as a search string. Any attribute listed has an index kept for that attribute, so setting too many attributes could affect server performance.
Group search fields	--groupsearch	Sets the fields in a group entry that can be used as a search string.
Certificate subject base		Sets the base DN to use when creating subject DNs for client certificates. This is configured when the server is set up.
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.

Field	Command-Line Option	Descriptions
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.

12.4.1. 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.

The screenshot shows the 'Configuration' subtab in the IPA Server interface. At the top, there are three buttons: 'Refresh', 'Revert', and 'Save'. Below these, the section 'Search Options' is displayed. It contains two input fields: 'Search size limit*' with the value '100' and 'Search time limit*' with the value '2'. Both fields have a blue asterisk indicating they are required.

Figure 12.4. Setting Search Limits

User Options

User search fields *

`uid,givenname,sn,telephonenumber,ou,t`

Figure 12.5. User Attributes

Group Options

Group search fields *

`cn,description`

Figure 12.6. Group Attributes

12.4.2. 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: 0=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
```

Chapter 13. ID Views

The *ID Views* feature enables you to specify POSIX attributes for users or groups. Every ID view is a collection of *user overrides* and *group overrides* that apply to specified hosts. An override provides a new user or group attribute that overrides the previous one. This enables you to, for example, replace a previously generated attribute with a new one.

An example use case for ID views is setting different user SSH public keys for different production environments, such as development, testing, or production.

Note

ID views also have several use cases in environments involving Active Directory, as described in the [Windows Integration Guide](#).

ID views can be added, modified, or deleted. Apart from specifying which ID attributes an ID view should override, you can also define which client hosts it should apply to.

13.1. User Overrides and Group Overrides

Every override is related to a user or user group. The following user attributes can be overridden in an ID view:

- » **uid**: user login name
- » **uidNumber**: user UID number
- » **gidNumber**: user GID number
- » **loginShell**: user login shell
- » **gecos**: user GECOS entry
- » **homeDirectory**: user home directory
- » **ipaSshPubkey**: user SSH public key or keys

The following group attributes can be overridden in an ID view:

- » **cn**: group name
- » **gidNumber**: group GID number

13.2. ID Views and SSSD

If the administrator applies another ID view on a client, the client and all the other clients applying this ID view must restart the SSSD service. Moreover, if the new ID view changes a UID or GID, the client and all the other clients applying the ID view must clear the SSSD cache.

Note that applying an ID view can have a negative impact on SSSD performance because certain optimizations and ID views cannot run at the same time. For example, ID views prevent SSSD from optimizing the process of looking up groups on the server. With ID views, SSSD must check every member on the returned list of group member names if

the group name is overridden. Without ID views, SSSD can only collect the user names from the member attribute of the group object. This negative effect will most likely become apparent when the SSSD cache is empty or after clearing the cache which makes all entries invalid.

13.3. Managing ID Views from the Web UI

To manage ID views from the IdM Web UI, open the **IPA Server** main tab and then select the **ID Views** subtab.

To add a new ID view:

1. Click **Add** above the list of all ID views.

The screenshot shows the IPA Server main menu with tabs for Identity, Policy, Authentication, Network Services, IPA Server, Role Based Access Control, ID Ranges, ID Views (which is highlighted in blue), Realm Domains, and Configuration. Below the menu is a search bar and a toolbar with Refresh, Delete, + Add, and Un-apply buttons. The main content area is titled 'ID Views' and shows a table with one entry: 'testview'. A message at the bottom says 'Showing 1 to 1 of 1 entries.'

Figure 13.1. Adding a New ID View

2. Fill out the information about the new ID view in the form that shows up.

The screenshot shows a modal dialog titled 'Add ID View'. It has fields for 'ID View Name' (containing 'New ID View') and 'Description'. A note at the bottom left says '* Required field'. At the bottom are buttons for 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

Figure 13.2. Form for Adding a New ID View

3. Click the **Add** button under the form.

To define the properties of an ID view:

1. Click on the name of the ID view in the list of ID views, and then choose the appropriate tab.

User to override	User login	UID	Home directory	Description
<input type="checkbox"/> admin		123456		
<input type="checkbox"/> employee		6666		

Figure 13.3. ID View Tabs

2. **Users** shows the list of users whose user attributes the ID view overrides.

User to override	User login	UID	Home directory	Description
<input type="checkbox"/> admin		123456		
<input type="checkbox"/> employee		6666		

Figure 13.4. Adding a User Override

Click **Add** to create a new user override; you will be asked to fill out the new values for the user attributes.

Add User ID override X

User to override * employee i

User login

GECOS

UID 6666

GID

Login shell

Home directory

Description

* Required field

Add Add and Add Another Add and Edit Cancel

Figure 13.5. Adding a User Override

- Click **Delete** to remove selected user overrides.
3. **User Groups** shows the list of user groups whose group attributes the ID view overrides.

Group ID overrides: New ID View

New ID View overrides:		New ID View applies to:	
Users	User Groups	Hosts	Settings
<input type="text" value="Search"/> <input type="button" value="🔍"/> <input type="button" value="⟳ Refresh"/> <input type="button" value="Delete"/> <input type="button" value="+ Add"/> 			
<input type="checkbox"/> Group to override	Group name	GID	Description
<input type="checkbox"/> admins	administrators		
<input type="checkbox"/> editors		4321	

Showing 1 to 2 of 2 entries.

Figure 13.6. User Groups Tab

Click **Add** to create a new user group override; you will be asked to fill out the new values for the group attributes.

Add Group ID override

Group to * override ⓘ	<input type="text" value="employees"/>
Group name	<input type="text"/>
GID	<input type="text" value="5555"/>
Description	<input type="text"/>

* Required field

Figure 13.7. Adding a Group Override

Click **Delete** to remove selected user group overrides.

4. **Hosts** shows the list of hosts or host groups to which the ID view applies.

ID View: New ID View

New ID View overrides:

- Users
- User Groups
- Hosts (1)**
- Settings

New ID View applies to:

- Refresh
- Un-apply
- Un-apply from host groups
- + Apply to hosts
- + Apply to host groups

<input type="checkbox"/> Host
<input type="checkbox"/> ipa.demo1.freeipa.org

Showing 1 to 1 of 1 entries.

Figure 13.8. Hosts Tab

Click **Apply to hosts** or **Apply to host groups** to add a new host or to add hosts belonging to a host group. In the form that shows up, move the required hosts or hosts group from the **Available** to **Prospective** column and click **Apply**.

Apply ID view New ID View on Hosts

Filter available Hosts

Available		Prospective	
<input type="checkbox"/>	Hosts	<input type="checkbox"/>	Hosts
<input type="checkbox"/>	testus.demo1.freeipa.org	<input type="checkbox"/>	ipa.demo1.freeipa.org

<

>

Figure 13.9. Applying an ID View to a Host

Un-apply removes the ID view from specified hosts. **Un-apply from host groups** enables you to remove the ID view from specified host groups.

5. **Settings** enables you to modify the ID view description.

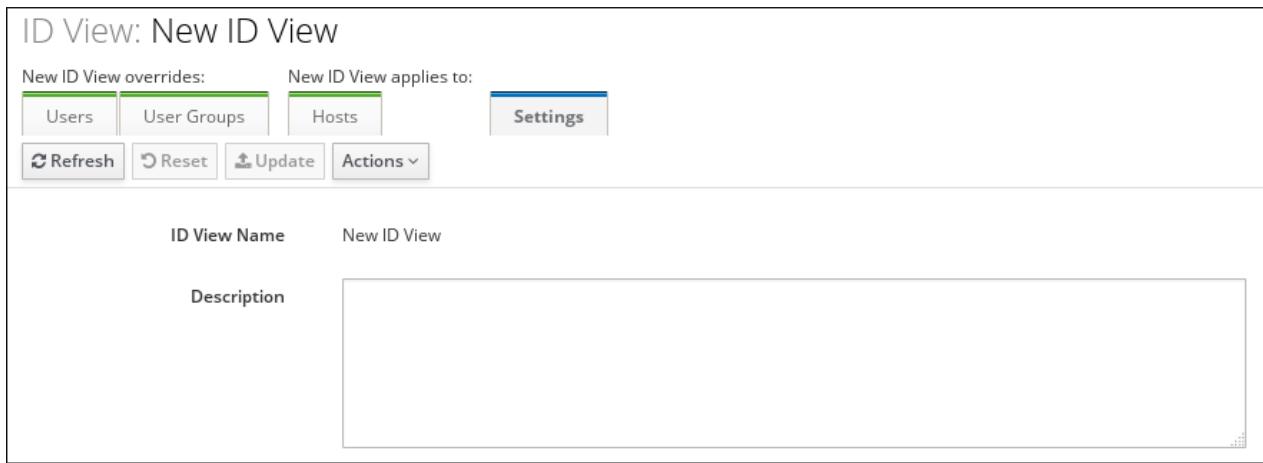


Figure 13.10. Settings Tab

13.4. Managing ID Views from the command line

To manage ID views on the command line, use the following commands:

- » **ipa idview-add** adds a new ID view
- » **ipa idview-apply** applies an ID view to specified hosts or host groups; any previously applied ID view is overridden
- » **ipa idview-del** deletes an ID view
- » **ipa idview-find** searches for a specified ID view
- » **ipa idview-mod** modifies an ID view
- » **ipa idview-show** displays information about an ID view
- » **ipa idview-unapply** removes an ID view from specified hosts or host groups

To manage group and user ID overrides, use the following commands:

- » **ipa idoverridegroup-add** adds a new group ID override
`ipa idoverrideuser-add` adds a new user ID override
- » **ipa idoverridegroup-del** deletes a group ID override
`ipa idoverrideuser-del` deletes a user ID override
- » **ipa idoverridegroup-find** searches for a specified group ID override
`ipa idoverrideuser-find` searches for a specified user ID override
- » **ipa idoverridegroup-mod** modifies a group ID override
`ipa idoverrideuser-mod` modifies a user ID override
- » **ipa idoverridegroup-show** displays information about a group ID override
`ipa idoverrideuser-show` displays information about a user ID override

For detailed information on what options can be passed to these commands, see the corresponding man pages or run one of them with the **--help** option added.



Note

The **--hostgroups** option applies the ID view to hosts belonging in a specified host group and can be used in the same way as the **--hosts** option. The **--hostgroups** option does not associate the ID view with the host group itself; it expands the members of the specified host group and applies **--hosts** individually to every one of them.

Chapter 14. 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 host name.

This chapter describes how to manage identity services that relate directly to the client machine:

- » DNS entries and settings
- » Machine authentication
- » Host name changes (which affect domain services)

14.1. 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 [Chapter 1, Introduction to Red Hat Identity Management](#)) 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 14.4, "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 23, 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*.

14.2. 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 14.1. Host Configuration Properties

UI Field	Command-Line Option	Description
Description	--desc= <i>description</i>	A description of the host.
Locality	--locality= <i>locality</i>	The geographic location of the host.
Location	--location= <i>location</i>	The physical location of the host, such as its data center rack.

UI Field	Command-Line Option	Description
Platform	--platform= <i>string</i>	The host hardware or architecture.
Operating system	--os= <i>string</i>	The operating system and version for the host.
MAC address	--macaddress= <i>address</i>	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	--sshpubkey= <i>string</i>	The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.
Principal name (not editable)	--principalname= <i>principal</i>	The Kerberos principal name for the host. This defaults to the host name during the client installation, unless a different principal is explicitly set in the -p . This can be changed using the command-line tools, but cannot be changed in the UI.
Set One-Time Password	--password= <i>string</i>	Sets a password for the host which can be used in bulk enrollment.
-	--random	Generates a random password to be used in bulk enrollment.
-	--certificate= <i>string</i>	A certificate blob for the host.
-	--updatedns	This sets whether the host can dynamically update its DNS entries if its IP address changes.

14.3. 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.

14.3.1. 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.

14.3.2. 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.

14.4. 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, host name, 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.

14.4.1. About the SSH Key Format

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

The IdM server can identify the type of key, such as an RSA or DSA key, from the uploaded key blob. However, in a key file such as `~/.ssh/known_hosts`, a key entry is identified by the host name 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.

14.4.2. About ipa-client-install and OpenSSH

The **ipa-client-install** script, by default, configures an OpenSSH server and client on the IdM client machine. It also configures SSSD to perform host and user key caching. Essentially, simply configuring the client does all of the configuration necessary for the host to use SSSD, OpenSSH, and Identity Management for key caching and retrieval.

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.

14.4.3. Uploading Host SSH Keys Through the Web UI

1. The key for a host can probably be retrieved from a `~/.ssh/known_hosts`. For example:

```
server.example.com,1.2.3.4 ssh-rsa
AAAAB3NzaC1yc2EAAAQEApvjBvSFSkTU0WQW4e0weeo0DZZ08F9Ud21x1Ly
6F0hzwpXFGIyxvXZ52+siHBHbbqGL5+14N7UvElruyslIHx9LYUR/pPKSMXC GyboLy
5aTNl50Q5EHwrhVnFDIKXkvp45945R7SKYCUTRumm0Iw6wq0XD4o+ILeVbV3wm cB1b
Xs36ZvC/M6riefn9PcJmh6vNCvIsbMY6S+FhkWUTT i0Xj jUDYRLLwM273FfWhzHK+S
SQXeBp/zIn1gFvJhSZMRi9HZpDoqxLbBB9QIdIw6U4MIjNmKsSI/ASpkFm2GuQ7ZK9
KuMItY2AoCuIRmRA dF8iYNHBTXNfFurGogXwRDjQ==
```

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 . *|
|          0 . . . *|
|          S + . 0+|
|          E . . . .|
|          . = . 0 |
|          o . . . 0|
|          .... |
+-----+
```

2. Copy the public key from the key file. The full key entry has the form `host name,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== [host name,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.

Hosts			
Search		Refresh	Delete
	Host name	Description	Enrolled
<input type="checkbox"/>	server.example.com		True
Showing 1 to 1 of 1 entries.			

Figure 14.1. List of Hosts

5. In the **Host Settings** area of the **Settings** tab, click **Add** next to **SSH public keys**.

Host Settings	
Host name	server.example.com
Principal name	host/server.example.com@EXAMPLE.COM
Description	<input type="text"/>
SSH public keys	existing_ssh_key <input type="button" value="Show/Set key"/> <input type="button" value="Delete"/> <input style="border: 2px solid red; border-radius: 5px; padding: 2px 10px; margin-left: 10px;" type="button" value="Add"/>

Figure 14.2. Adding an SSH Key

6. Paste in the public key for the host, and click **Set**.

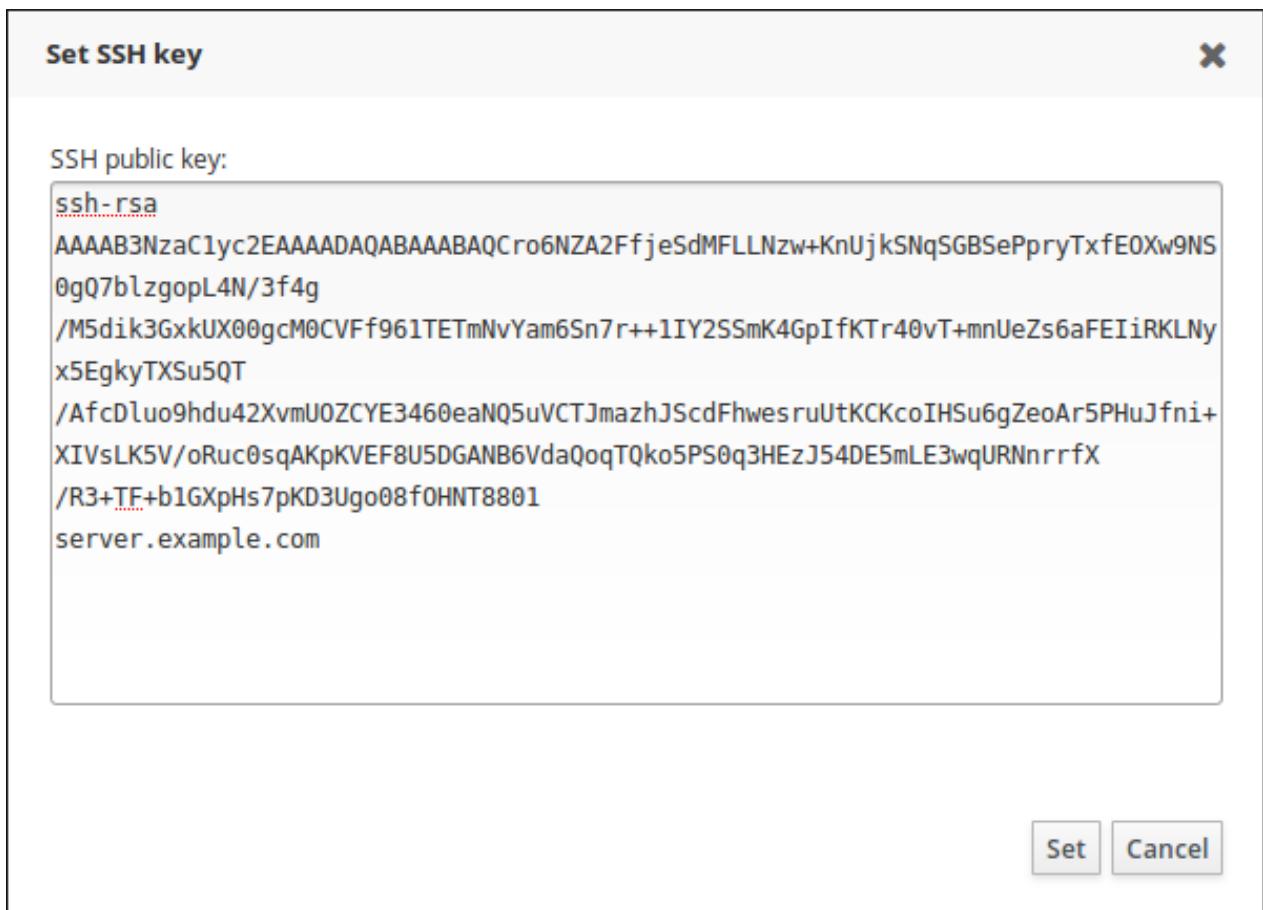


Figure 14.3. Setting an SSH Key

The **SSH public keys** area now shows the new key. Clicking **Show/Set key** opens the submitted key.

7. To upload multiple keys, click the **Add** link below the list of public keys, and upload the other keys.
8. When all the keys have been submitted, click **Save** 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 [3].

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](#).

14.4.4. Adding Host Keys from the Command Line

Host SSH keys are added to host entries in IdM, either when the host is created using **host-add** or by modifying the entry later.



Note

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](#).

14.4.5. Removing Host Keys

Host keys can be removed once they expire or are no longer valid.

To remove an individual host key, it is easiest to remove the key through the web UI:

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

Hosts			
Search		Refresh	Delete
	Host name	Description	Enrolled
<input type="checkbox"/>	server.example.com		True
Showing 1 to 1 of 1 entries.			

Figure 14.4. List of Hosts

3. In the **SSH public keys** area, click **Delete** by the fingerprint of the key to remove it.

Settings	Host Groups	Netgroups	Roles	HBAC Rules	Sudo Rules
Refresh	Revert	Save	Actions ▾		
<h3>Host Settings</h3> <p>Host name server.example.com</p> <p>Principal name host/server.example.com@EXAMPLE.COM</p> <p>Description</p> <p>SSH public keys existing_ssh_key Show/Set key Delete</p> <p>Add</p>					

Figure 14.5. Public Key Deletion

4. Click **Save** 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
```

14.5. Setting ethers information for a host

NIS can host an ethers table which can be used to 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:

1. 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
```

14.6. Managing Host Groups

Host groups are a way of centralizing control over important management tasks, particularly access control.

All groups in Identity Management are essentially *static* groups, meaning that the members of the group are manually and explicitly added to the group. 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.

14.6.1. Creating Host Groups

14.6.1.1. Creating Host Groups from the Web UI

1. Open the **Identity** tab, and select the **Host Groups** subtab.
2. Click **Add** 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 14.6.2.2, “Adding Host Group Members from the Web UI”](#).

14.6.1.2. Creating Host Groups from the Command Line

New groups are created using the **hostgroup-add** command. (This adds only the group; members are added separately.)

Two attributes are always required: the group name and the group description. If those attributes are not given as arguments, then the script prompts for them.

```
$ ipa hostgroup-add groupName --desc="description"
```

14.6.2. Adding Host Group Members

14.6.2.1. Showing and Changing Group Members

Members can be added to a group through the group configuration. There are tabs for all the member types which can belong to the group, and an administrator picks all of the 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.

On the host group page in the web UI, **host_group members** shows entries that can join the displayed host group, and **host_group is a member of** shows groups that the displayed host group can join.

The screenshot shows the IPA web interface for managing host groups. The top navigation bar includes 'Users', 'User Groups', 'Hosts', 'Host Groups' (which is underlined, indicating it's the active tab), 'Netgroups', 'Services', and 'Automember'. Below the navigation is a breadcrumb trail: 'Host Groups > host_group'. The main content area is titled 'Host Group: host_group'. It features two sections highlighted with red boxes: 'host_group members:' and 'host_group is a member of:'. Under 'host_group members:', there are tabs for 'Hosts' and 'Host Groups', along with 'Settings'. Under 'host_group is a member of:', there are tabs for 'Host Groups', 'Netgroups', 'HBAC Rules', and 'Sudo Rules'. At the bottom of the page are buttons for 'Refresh', 'Delete', and '+ Add', along with checkboxes for 'Show Results' (with options for 'Direct Membership' and 'Indirect Membership') and 'Host name'. A message at the bottom states 'No entries.'

Figure 14.6. Host Group Page

14.6.2.2. Adding Host Group Members from the Web UI

1. Open the **Identity** tab, and select the **Host Groups** subtab.

- Click the name of the group to which to add members.

The screenshot shows a 'Host Groups' list interface. At the top, there is a search bar, a refresh button, a delete button, and an 'Add' button. Below the header, there is a table with two columns: 'Host-group' and 'Description'. The first row contains the value 'host_group' in the 'Host-group' column, which is highlighted with a red box. In the 'Description' column, there is an empty field. At the bottom of the table, it says 'Showing 1 to 1 of 1 entries.'

Figure 14.7. List of Host Groups

- Click the **Add** link at the top of the task area.

The screenshot shows a 'Host Group: host_group' details interface. At the top, it says 'Host Group: host_group'. Below that, there are two tabs: 'host_group members:' and 'host_group is a member of:'. Under 'host_group members:', there are three buttons: 'Hosts', 'Host Groups', and 'Settings', with 'Host Groups' being the active tab. Under 'Host Groups', there are buttons for 'Refresh', 'Delete', and '+ Add', with '+ Add' being highlighted with a red box. There is also a checkbox for 'Host name' and a message 'No entries.'. Under 'host_group is a member of:', there are tabs for 'Host Groups', 'Netgroups', 'HBAC Rules', and 'Sudo Rules'. At the bottom, there is a 'Show Results' dropdown with 'Direct Membership' selected.

Figure 14.8. Adding a Member to a Host Group

- Move the names of the hosts to add to the **Prospective** column, and then click **Add** to confirm.

14.6.2.3. 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
-----
```

[3] The key type is determined automatically from the key itself, if it is not included in the uploaded key.

Chapter 15. 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 [Chapter 1, Introduction to Red Hat Identity Management](#)) provides three main services specifically for machines:

- » DNS
- » Kerberos
- » Certificate management

15.1. 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.

15.1.1. Adding Services and Keytabs from the Web UI

1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the **Add** button 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 host name of the IdM host on which the service is running. The host name is used to construct the full service principal name.
5. Click the **Add** button to save the new service principal.

6. Use the **ipa-getkeytab** command to generate and assign the new keytab for the service principal.

```
[root@ipaserver ~]# # ipa-getkeytab -s ipaserver.example.com -p
HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e 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 host name 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.

15.1.2. Adding Services and Keytabs from the Command Line

1. Create the service principal. The service is recognized through a name like **service/FQDN**:

```
# ipa service-add serviceName/hostname
```

For example:

```
$ ipa service-add HTTP/server.example.com
-----
Added service "HTTP/server.example.com@EXAMPLE.COM"
-----
Principal: HTTP/server.example.com@EXAMPLE.COM
Managed by: ipaserver.example.com
```

2. Create the service keytab file using the **ipa-getkeytab** command. This command is run on the client in the IdM domain. (Actually, it can be run on any IdM server or client, and then the keys copied to the appropriate machine. However, it is simplest to run the command on the machine with the service being created.)

The command requires the Kerberos service principal (**-p**), the IdM server name (**-s**), the file to write (**-k**), and the encryption method (**-e**). Be sure to copy the keytab to the appropriate directory for the service.

For example:

```
# ipa-getkeytab -s server.example.com -p HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e 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 host name 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.

15.2. 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 host names 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.

15.3. 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.

15.4. Retrieve Existing Keytabs for Multiple Servers

In some scenarios, like in a cluster environment, the same keytab file is required for a service represented on one common host name by different machines. IdM commands can be used to retrieve the same keytab on each of the hosts.

To prepare the common host name and the service principal, run the following commands on an IdM server:

1. Authenticate as **admin** user:

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

2. Add a common forward DNS record for all IP addresses that share this host name:

```
[root@ipaserver ~]# ipa dnsrecord-add idm.example.com cluster --a-rec={192.0.2.40,192.0.2.41}
Record name: cluster
A record: 192.0.2.40, 192.0.2.41
```

3. Create a new host entry object for the common DNS name:

```
[root@ipaserver ~]# ipa host-add cluster.idm.example.com
-----
Added host "cluster.idm.example.com"
-----
Host name: cluster.idm.example.com
```

```
Principal name: host/cluster.idm.example.com@IDM.EXAMPLE.COM
Password: False
Keytab: False
Managed by: cluster.idm.example.com
```

4. Add the service principal for the host:

```
[root@ipaserver ~]# ipa service-add HTTP/cluster.idm.example.com
-----
Added service "HTTP/cluster.idm.example.com@IDM.EXAMPLE.COM"
-----
Principal: HTTP/cluster.idm.example.com@IDM.EXAMPLE.COM
Managed by: cluster.idm.example.com
```

5. Add the hosts to the service, that should be able to retrieve the keytab from IdM:

```
[root@ipaserver ~]# ipa service-allow-retrieve-keytab
HTTP/cluster.idm.example.com --hosts=
{node01.idm.example.com,node02.idm.example.com}
Principal: HTTP/cluster.idm.example.com@IDM.EXAMPLE.COM
Managed by: cluster.idm.example.com
Hosts allowed to retrieve keytab: node01.idm.example.com,
node02.idm.example.com
-----
Number of members added 2
-----
```

6. Grant permission to create a new keytab to one host:

```
[root@ipaserver ~]# ipa service-allow-create-keytab
HTTP/cluster.idm.example.com --hosts=node01.idm.example.com
Principal: HTTP/cluster.idm.example.com@IDM.EXAMPLE.COM
Managed by: cluster.idm.example.com
Hosts allowed to retrieve keytab: node01.idm.example.com,
node02.idm.example.com
Hosts allowed to create keytab: node01.idm.example.com
-----
Number of members added 1
-----
```

On the clients, follow these steps:

1. Authenticate with the hosts Kerberos keytab:

```
# kinit -kt /etc/krb5.keytab
```

2. a. On the client you granted the respective permission to, generate a new keytab and store it in a file:

```
[root@node01 ~]# ipa-getkeytab -s ipaserver.idm.example.com -
p HTTP/cluster.idm.example.com -k /tmp/client.keytab
```

b. On all other clients, retrieve the existing keytab from the IdM server by adding the **-r** option to the command:

```
[root@node02 ~]# ipa-getkeytab -r -s
ipaserver.idm.example.com -p HTTP/cluster.idm.example.com -k
/tmp/client.keytab
```



Warning

Be aware that if you omit the **-r** option, a new keytab will be generated. This invalidates all previously retrieved keytabs for this service principal.

15.5. 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.

15.5.1. 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
[jsmith@ipaserver ~]$ 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.

15.5.2. Re-enabling 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
```

Chapter 16. Delegating User Access to Hosts and Services

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.

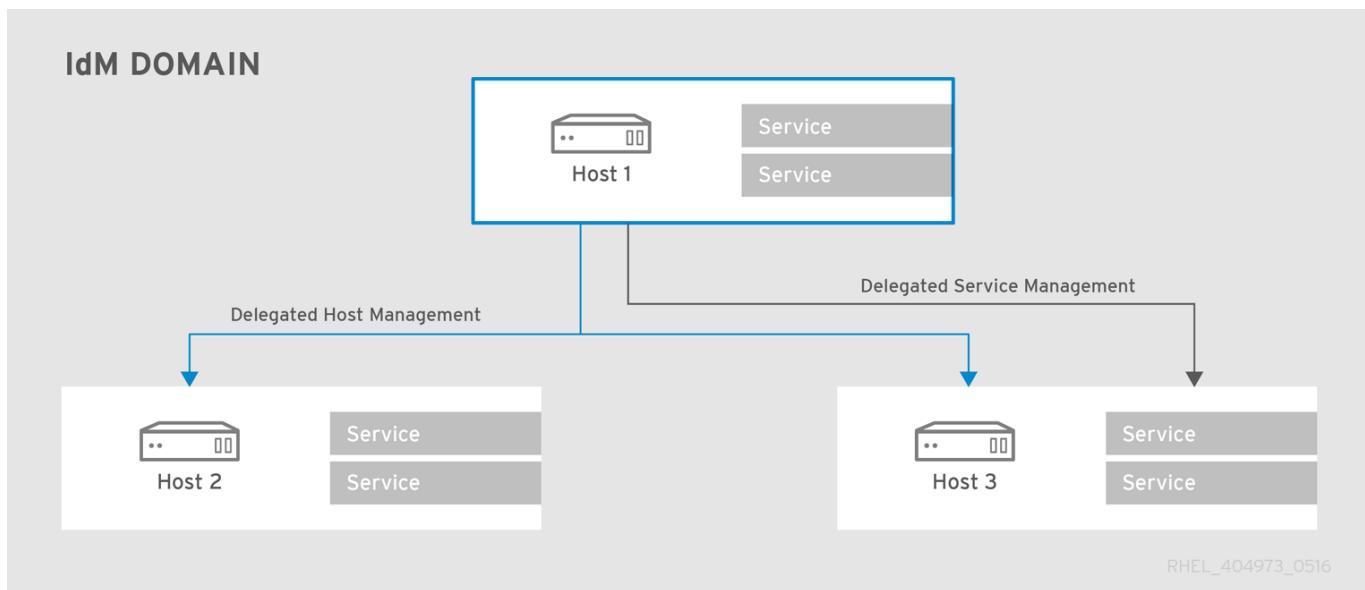


Figure 16.1. Host and Service Delegation

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.

16.1. 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
```

For more information on creating certificate requests and using **ipa cert-request**, see [Section 17.1.1, “Requesting New Certificates for a User, Host, or Service”](#).

16.2. 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
```

16.3. 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 that *can manage* the selected host/service.

Figure 16.2. Host Subtab

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 arrow button, **>**, to move the hosts to the selection box.

Figure 16.3. Host/Service Delegation Management

6. Click the **Add** button to close the selection box and to save the delegation settings.

16.4. 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 17. Managing Certificates for Users, Hosts, and Services

Identity Management (IdM) supports two types of certificate authorities (CAs):

Integrated Certificate System CA

The integrated CA can create, revoke, and issue certificates for users, hosts, and services. For more details, see [Section 17.1, “Managing Certificates with the Integrated IdM CA”](#).

External CA

An external CA is a CA other than the integrated IdM CA.

Using IdM tools, you add certificates issued by these CAs to users, services, or hosts as well as remove them. For more details, see [Section 17.2, “Managing Certificates Issued by External CAs”](#).

Each user, host, or service can have multiple certificates assigned.

Note

For more details on the supported CA configurations of the IdM server, see [Section 2.3.2, “Determining What CA Configuration to Use”](#).

17.1. Managing Certificates with the Integrated IdM CA

17.1.1. Requesting New Certificates for a User, Host, or Service

To request a certificate using:

- » the IdM web UI, see [Section 17.1.1, “Web UI: Requesting New Certificates”](#).
- » the command line, see [Section 17.1.1, “Command Line: Requesting New Certificates”](#).

Note that you must generate the certificate request itself with a third-party tool. The following procedures use the `certutil` utility.



Important

Services typically run on dedicated service nodes on which the private keys are stored. Copying a service's private key to the IdM server is considered insecure. Therefore, when requesting a certificate for a service, create the CSR on the service node.

Web UI: Requesting New Certificates

1. Under the **Identity** tab, select the **Users, Hosts, or Services** subtab.
2. Click the name of the user, host, or service to open its configuration page.

Hosts			
	Host name	Description	Enrolled
<input type="checkbox"/>	server.example.com		True
Showing 1 to 1 of 1 entries.			

Figure 17.1. List of Hosts

3. Click **Actions** → **New Certificate**.
4. Follow the instructions on the screen for using **certutil**.
5. Click **Issue**.

Command Line: Requesting New Certificates

1. Generate a certificate request.
 - a. Create a certificate database or use an existing one. To create a new database:

```
# certutil -N -d path_to_database
```

- b. Create the certificate request and use output redirection to save it to a file.

```
# certutil -R -d path_to_database -a -g key_size -s
"CN=server.example.com,0=EXAMPLE.COM" >
certificate_request.csr
```

2. Submit the certificate request file to the server. Be sure to specify the Kerberos principal to associate with the newly-issued certificate.

```
# ipa cert-request certificate_request.csr --
principal=host/server.example.com
```

IdM uses the **caIPAserviceCert** certificate profile for the certificate by default. To select a custom profile, use the **--profile-id** option with the **ipa cert-request**. For more information about certificate profiles, see [Section 17.4, “Certificate Profiles”](#).

17.1.2. Revoking Certificates with the Integrated IdM CA

If you need to invalidate the certificate before its expiration date, you can revoke it. To revoke a certificate using:

- » the IdM web UI, see [Section 17.1.2, “Web UI: Revoking Certificates”](#)
- » the command line, see [Section 17.1.2, “Command Line: Revoking Certificates”](#)

A revoked certificate is invalid and cannot be used for authentication. All revocations are permanent, except for reason 6: Certificate Hold.

Table 17.1. Revocation Reasons

ID	Reason	Explanation
0	Unspecified	
1	Key Compromised	The key that issued the certificate is no longer trusted. Possible causes: lost token, improperly accessed file.
2	CA Compromised	The CA that issued the certificate is no longer trusted.
3	Affiliation Changed	Possible causes: <ul style="list-style-type: none">» A person has left the company or moved to another department.» A host or service is being retired.
4	Superseded	A newer certificate has replaced the current certificate.
5	Cessation of Operation	The host or service is being decommissioned.
6	Certificate Hold	The certificate is temporarily revoked. You can restore the certificate later.
8	Remove from CRL	The certificate is not included in the certificate revocation list (CRL).
9	Privilege Withdrawn	The user, host, or service is no longer permitted to use the certificate.
10	Attribute Authority (AA) Compromise	The AA certificate is no longer trusted.

Web UI: Revoking Certificates

To revoke a certificate:

1. Open the **Authentication** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to open the certificate information page.

Certificates		
Subject	Serial Number	Subject
<input type="checkbox"/>	1	CN=Certificate Authority,O=EXAMPLE.COM
<input type="checkbox"/>	2	CN=OCSP Subsystem,O=EXAMPLE.COM
<input type="checkbox"/>	3	CN=server.example.com,O=EXAMPLE.COM
<input type="checkbox"/>	4	CN=CA Subsystem O=EXAMPLE COM

Figure 17.2. List of Certificates

3. Click **Actions** → **Revoke Certificate**.
4. Select the reason for revoking, and click **Revoke**. See [Table 17.1, “Revocation Reasons”](#) for details.

Command Line: Revoking Certificates

Use the **ipa cert-revoke** command, and specify:

- » the certificate serial number
- » a number that identifies the reason for the revocation; see [Table 17.1, “Revocation Reasons”](#) for details

For example, to revoke the certificate with serial number **1032** because of reason 1: Key Compromised:

```
$ ipa cert-revoke 1032 --revocation-reason=1
```

17.1.3. Restoring Certificates with the Integrated IdM CA

If you have revoked a certificate because of reason 6: Certificate Hold, you can restore it again. To restore a certificate using:

- » the IdM web UI, see [Section 17.1.3, “Web UI: Restoring Certificates”](#)
- » the command line, see [Section 17.1.3, “Command Line: Restoring Certificates”](#)

Web UI: Restoring Certificates

1. Open the **Authentication** tab, and select the **Certificates** subtab.
2. Click the serial number of the certificate to open the certificate information page.

Subject		Serial Number	Subject
<input type="checkbox"/>	1		CN=Certificate Authority,O=EXAMPLE.COM
<input type="checkbox"/>	2		CN=OCSP Subsystem,O=EXAMPLE.COM
<input type="checkbox"/>	3		CN=server.example.com,O=EXAMPLE.COM
<input type="checkbox"/>	4		CN=CA Subsystem,O=EXAMPLE.COM

Figure 17.3. List of Certificates

3. Click **Actions** → **Restore Certificate**.

Command Line: Restoring Certificates

Use the **ipa cert-remove-hold** command and specify the certificate serial number. For example:

```
$ ipa cert-remove-hold 1032
```

17.2. Managing Certificates Issued by External CAs

17.2.1. Command Line: Adding and Removing Certificates Issued by External CAs

To add a certificate to a user, host, or service:

- » **ipa user-add-cert**
- » **ipa host-add-cert**
- » **ipa service-add-cert**

To remove a certificate from a user, host, or service:

- » **ipa user-remove-cert**
- » **ipa host-remove-cert**
- » **ipa service-remove-cert**

A certificate issued by an external CA is not revoked after you remove it from IdM. This is because the certificate does not exist in the IdM CA database. You can only revoke these certificates manually from the external CA side.

The commands require you to specify the following information:

- » the name of the user, host, or service
- » the Base64-encoded DER certificate

To run the commands interactively, execute them without adding any options.

To provide the required information directly with the command, use command-line arguments and options:

```
$ ipa user-add-cert user --certificate=MIQTPr...Awg...
```

Note

Instead of copy-pasting the certificate contents into the command line, you can convert the certificate to the DER format and then re-encode it to base64. For example, to add the **user_cert.pem** certificate to **user**:

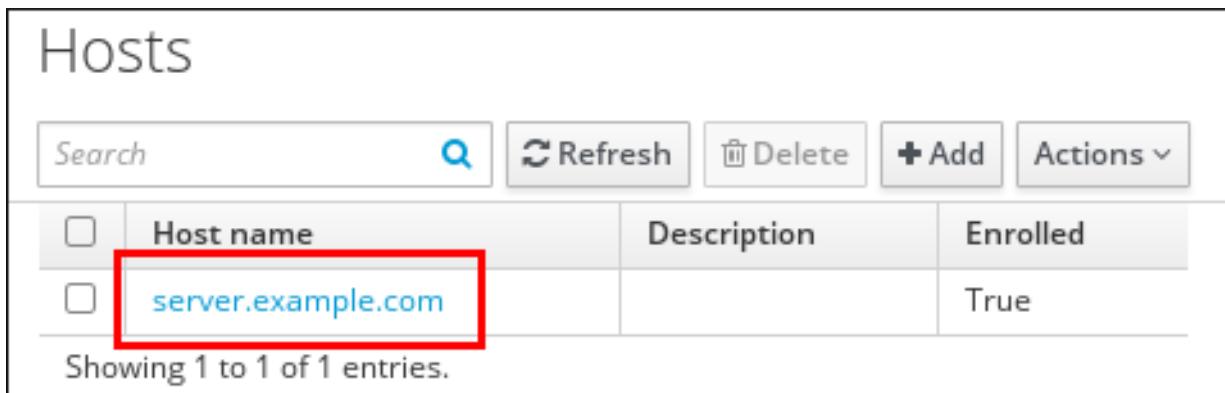
```
$ ipa user-add-cert user --certificate="$(openssl x509 -outform der -in user_cert.pem | base64 -w 0)"
```

17.3. Listing and Displaying Certificates

Listing and Displaying Certificates in the Web UI

To list certificates assigned to a user, host, or service entry:

1. Open the **Identity** tab, and select the **Users, Hosts, or Services** subtab.
2. Click on the name of the user, host, or service to open its configuration page.



	Host name	Description	Enrolled
<input type="checkbox"/>	server.example.com		True

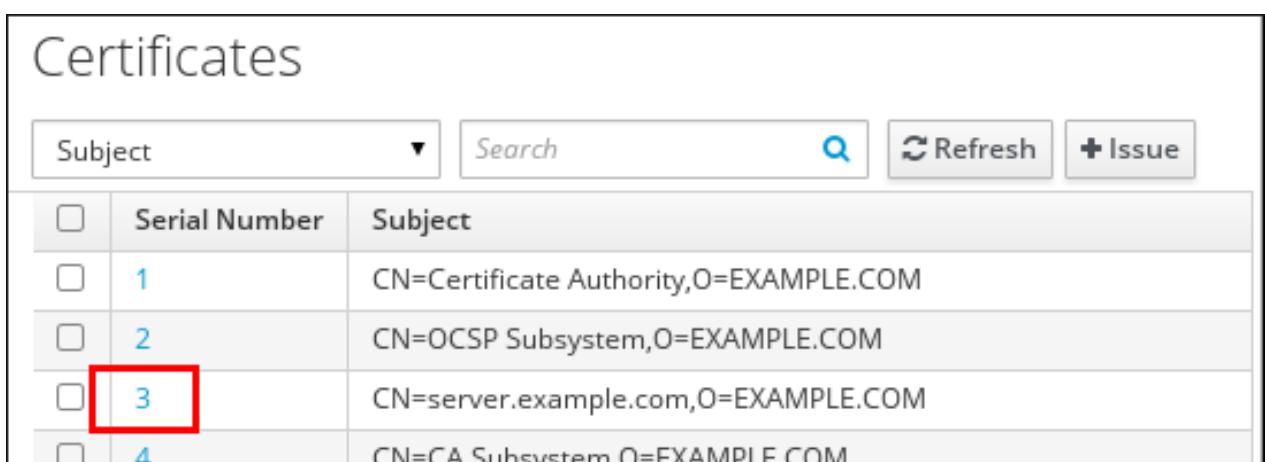
Showing 1 to 1 of 1 entries.

Figure 17.4. List of Hosts

3. The configuration page lists all certificates assigned to the entry. Additionally, clicking **Show** displays a particular certificate.

To list all certificates registered on the IdM server:

1. Open the **Authentication** tab, and select the **Certificates** subtab.
2. A list of all certificates is displayed in the **Certificates** section. To display a particular certificate, click on its serial number.



	Serial Number	Subject
<input type="checkbox"/>	1	CN=Certificate Authority,O=EXAMPLE.COM
<input type="checkbox"/>	2	CN=OCSP Subsystem,O=EXAMPLE.COM
<input type="checkbox"/>	3	CN=server.example.com,O=EXAMPLE.COM
<input type="checkbox"/>	4	CN=CA Subsystem O=EXAMPLE.COM

Figure 17.5. List of Certificates

Listing Certificates from the Command Line

To list all certificates in the IdM database, run the **ipa cert-find** command.

```
$ ipa cert-find
-----
10 certificates matched
-----
Serial number (hex): 0x1
Serial number: 1
Status: VALID
Subject: CN=Certificate Authority,0=EXAMPLE.COM
...
-----
Number of entries returned 10
-----
```

You can filter the search results by specifying certain certificate properties, such as issue date or validity date. For example, to search by an issue date interval, use the **--issuedon-from** or **--issuedon-to** options to specify the start and end points or a period of time.

```
ipa cert-find --issuedon-from=2016-01-07 --issuedon-to=2016-02-07
```

For a complete list of options used to filter the search for a certificate, run **ipa cert-find** with the **--help** option added.

Displaying Certificates from the Command Line

To display a certificate, use the **ipa cert-show** command and specify the serial number.

```
$ ipa cert-show 132
Serial number: 132
Certificate:
MIIDtzCCAp+gAwIBAgIBATANBgkqhkiG9w0BAQsFADBBMR8wHQYDVQQKExZMQUIu
...
LxIQjrEFtJmoBGb/TWRlwGEWylayr4iTf1ayZ+RGNylLalEAtk9RLjEjg==
Subject: CN=Certificate Authority,0=EXAMPLE.COM
Issuer: CN=Certificate Authority,0=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
```

To display the certificates assigned to a user, host, or service entry, use **ipa cert-show** and specify the entry. For example, to display the certificate assigned to a user:

```
$ ipa user-show user
User login: user
...
Certificate: MIICfzCCAwcCAQA...
...
```

You can also save a certificate to a file by adding the **--out** option to **ipa cert-show**.

```
$ ipa cert-show certificate_serial_number --out=path_to_file
```

Note that if the user, host, or service has more than one certificate, the **--out** option exports all of them. The certificate or certificates are exported as PEM objects.

17.4. 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 17.5, “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 17.6, “Using Certificate Profiles and ACLs to Issue User Certificates with the IdM CA”](#).

17.4.1. 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 28.4, “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-mod 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"
```

17.4.2. 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.

The screenshot shows the 'Certificates' section of the web interface. On the left, a sidebar lists 'Certificates', 'Certificates', 'Certificate Profiles >', and 'CA ACLs'. The 'Certificate Profiles' item is selected and highlighted in blue. The main content area is titled 'Certificate Profiles' and contains a search bar with a magnifying glass icon and a refresh/delete button. Below the search bar is a table with three rows. The first row has columns for 'Profile ID' (checkbox), 'Profile description' (text input), and 'Store issued certificates' (checkbox). The second row has columns for 'IECUserRoles' (checkbox), 'User profile that includes IECUserRoles extension from request' (text input), and 'TRUE' (checkbox). The third row has columns for 'calPAserviceCert' (checkbox), 'Standard profile for network services' (text input), and 'TRUE' (checkbox). A note at the bottom says 'Showing 1 to 2 of 2 entries.'

Profile ID	Profile description	Store issued certificates
<input type="checkbox"/> IECUserRoles	User profile that includes IECUserRoles extension from request	TRUE
<input type="checkbox"/> calPAserviceCert	Standard profile for network services	TRUE

Figure 17.6. 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.

The screenshot shows the 'smime' certificate profile configuration page. At the top, there are three buttons: 'Refresh', 'Revert', and 'Save', with 'Save' being highlighted and enclosed in a red box. Below the buttons, the 'Profile ID' is listed as 'smime'. The 'Profile description *' field contains 'S/Mime certificates'. There is also an 'Undo' button next to it. At the bottom, the 'Store issued certificates' option is checked with a blue checkmark.

Figure 17.7. 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 17.4.1, “Certificate Profile Management from the Command Line”](#).

17.4.3. 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.

17.5. 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 17.4, “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 17.6, “Using Certificate Profiles and ACLs to Issue User Certificates with the IdM CA”](#).

17.5.1. 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 28.4, “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 you 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 17.5.1, “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 17.5.1, “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 17.5.1, “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 17.5.1, “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 17.5.1, “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 17.5.1, “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
```

17.5.2. 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
hosts_services_caIPAserviceCert	Enabled	

Figure 17.8. 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.

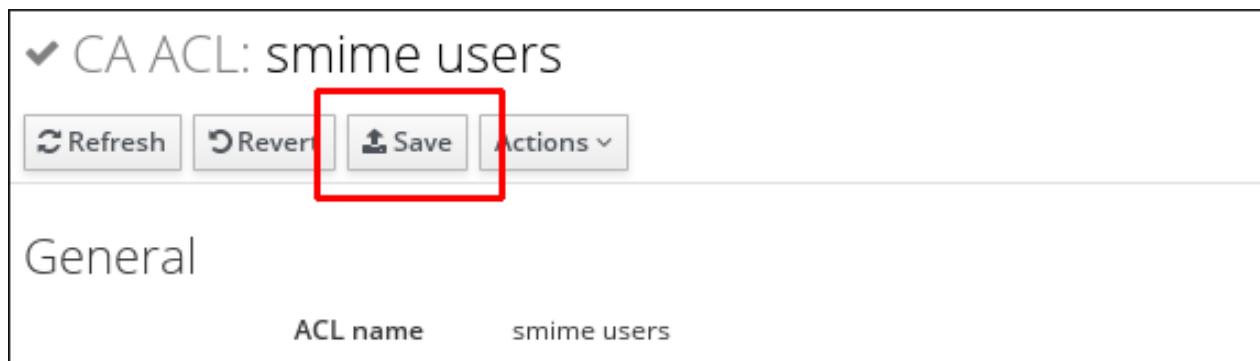


Figure 17.9. Modifying a CA ACL Rule in the Web UI

17.6. Using Certificate Profiles and ACLs to Issue User Certificates with the IdM CA

Users can request certificates for themselves when permitted by the Certificate Authority access control lists (CA ACLs). The following procedures use certificate profiles and CA ACLs, which are described separately in [Section 17.4, “Certificate Profiles”](#) and [Section 17.5, “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: MIICfzCCAwcCAQ...
...
```

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 17.4, “Certificate Profiles”](#).

2. In the web UI, under the **Authentication** tab, open the **CA ACLs** section.

ACL name	Status	Description
hosts_services_calPAserviceCert	Enabled	

Showing 1 to 1 of 1 entries.

Figure 17.10. 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.

- In the **Add CA ACL** window that opens, fill in the required information about the new CA ACL.

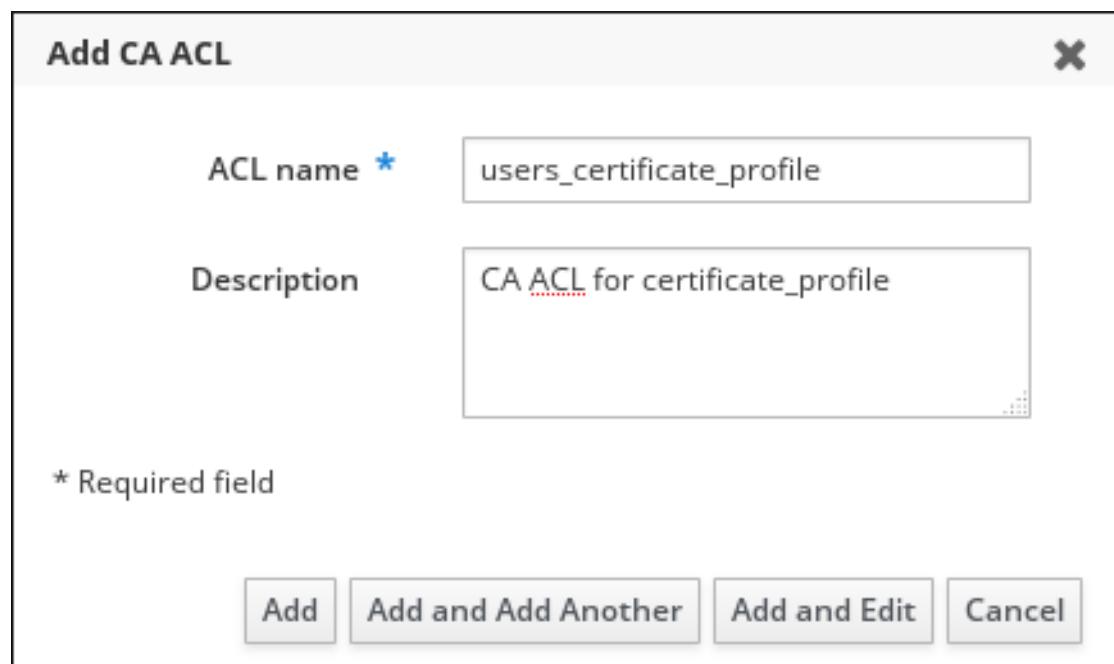


Figure 17.11. Adding a New CA ACL

Then, click **Add and Edit** to go directly to the CA ACL configuration page.

- In the CA ACL configuration page, scroll to the **Profiles** section and click **Add** at the top of the profiles list.

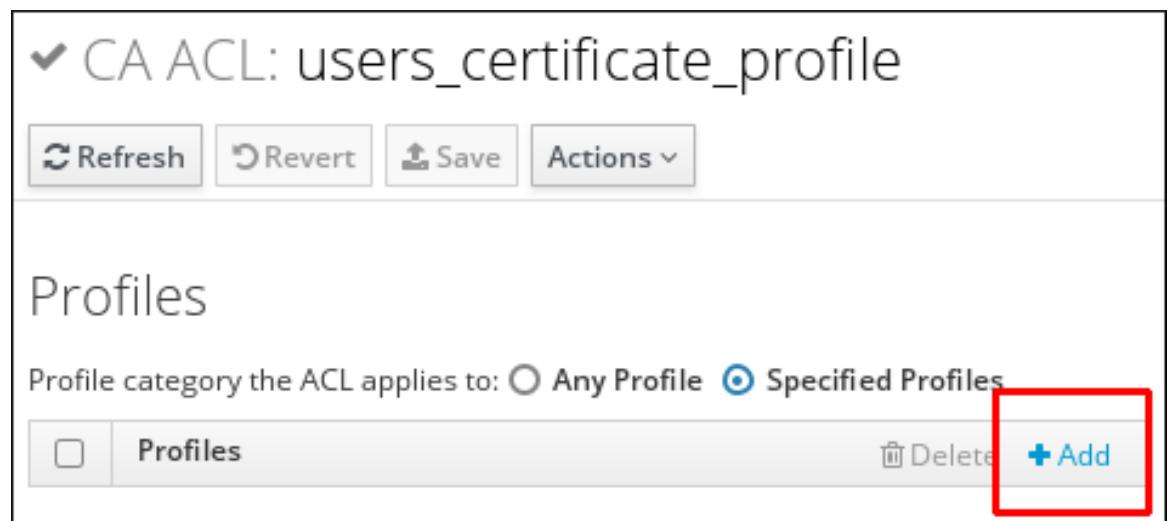
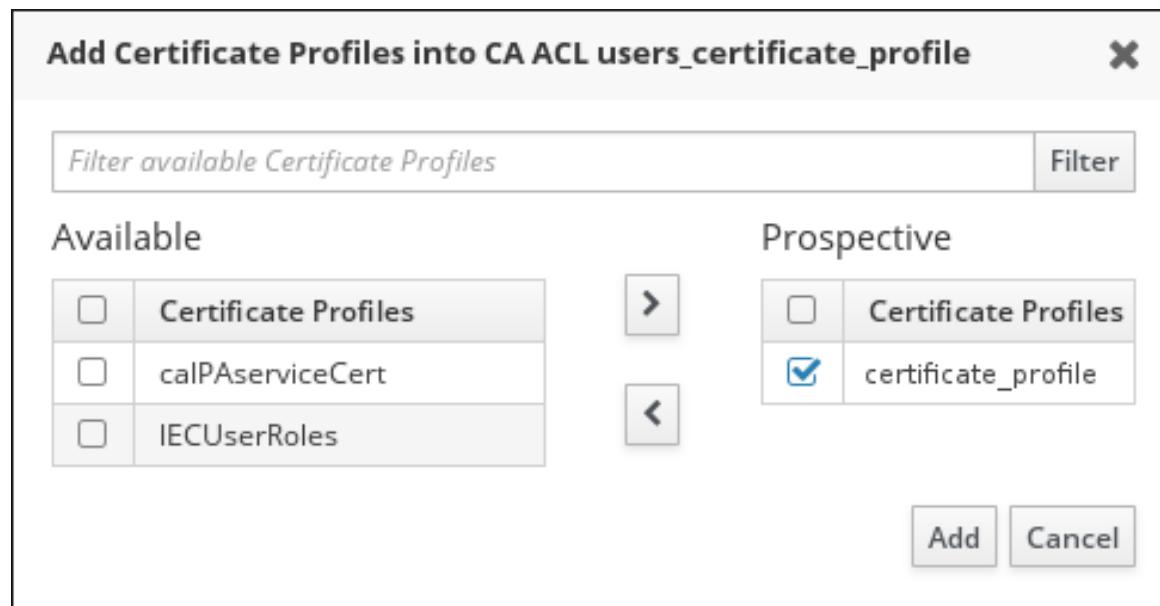


Figure 17.12. Adding a Certificate Profile to the CA ACL

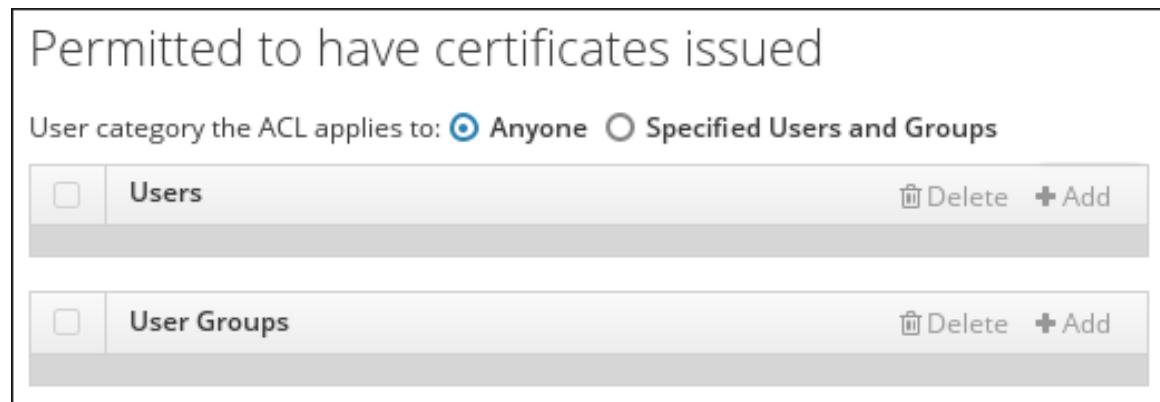
- Add the custom certificate profile to the CA ACL by selecting the profile and moving it to the **Prospective** column.

**Figure 17.13. Selecting a Certificate Profile**

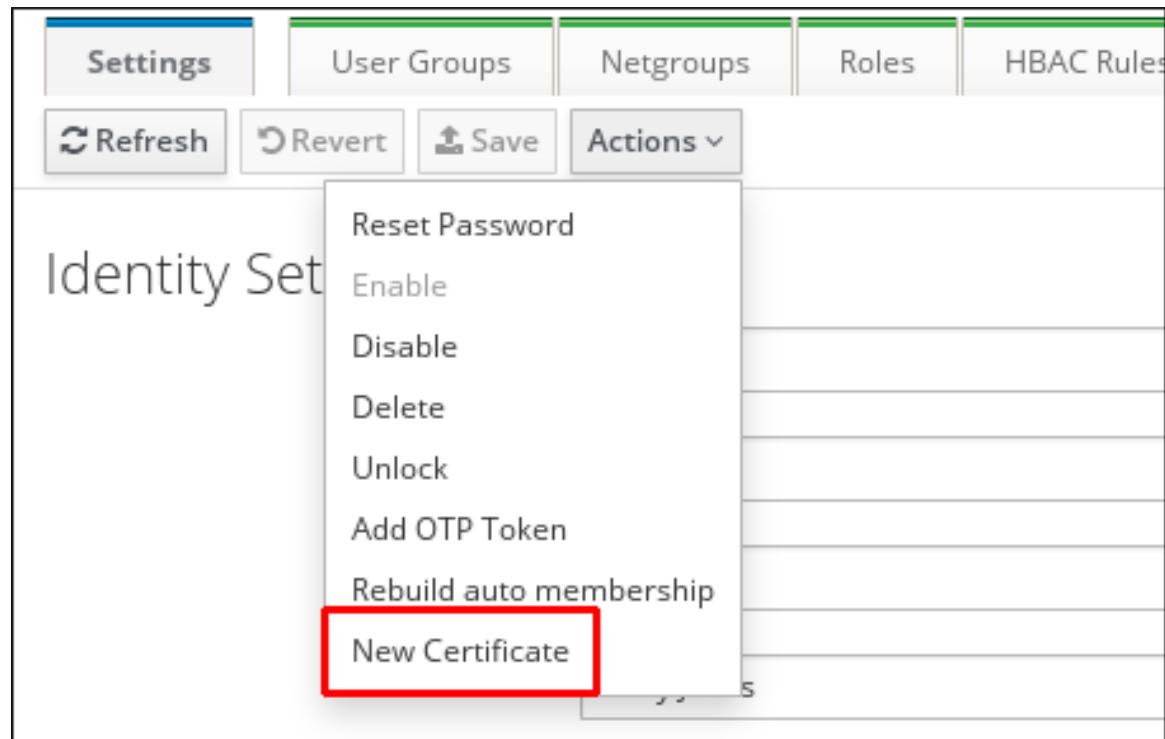
Then, click **Add**.

- 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 17.14. 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 17.15. Requesting a Certificate for a User**

- c. Fill in the required information.

The screenshot shows a dialog box titled "Issue New Certificate for User user". It has a dropdown for "Profile ID" set to "certificate_profile". The main area contains three numbered steps: 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-----
MIIBfDCB5gIBADAMQ4wDAYDVQQDDAVhbGljZTCBnzANBgkqhkiG9w0BAQEFAAOB
jQAwgYkCgYEAmJhtELuvf/gY8vsha9dQKtZmM+irSotMMRz3CiSRdQSsvxJeOID
QtPraU8z0gzbImwZY0ZGUkxKvJPGG4ErjHtcIR1b5V+XEuMr1R+TmclCXqGId7FP
l+IGIIvrtw2vuHplEHoPQps0VUzjEp8ql5Kr9lMmUTKI9QF/4EUw2VECAwEAAaAt
-----END CERTIFICATE REQUEST-----
```

At the bottom are "Issue" and "Cancel" buttons.

Figure 17.16. Issuing a Certificate for a User

Then, click **Issue**.

After this, the newly issued certificate is visible in the user configuration page.

Chapter 18. Storing Authentication Secrets with Vaults

A vault is a secure location for storing, retrieving, sharing, and recovering secrets. A secret is security-sensitive data that should only be accessible by a limited group of people or entities. For example, secrets include:

- » passwords
- » PINs
- » private SSH keys

Users and services can access the secrets stored in a vault from any machine enrolled in the Identity Management (IdM) domain.

Note

Vault is only available from the command line, not from the IdM web UI.

Use cases for vaults include:

Storing personal secrets of a user

See [Section 18.4, “Storing a User's Personal Secret”](#) for details.

Storing a secret for a service

See [Section 18.5, “Storing a Service Secret in a Vault”](#) for details.

Storing a common secret used by multiple users

See [Section 18.6, “Storing a Common Secret for Multiple Users”](#) for details.

Note that to use vaults, you must meet the conditions described in [Section 18.2, “Prerequisites for Using Vaults”](#).

18.1. How Vaults Work

18.1.1. Vault Owners, Members, and Administrators

IdM distinguishes the following vault user types:

Vault owner

A vault owner is a user or service with basic management privileges on the vault. For example, a vault owner can modify the properties of the vault or add new vault members.

Each vault must have at least one owner. A vault can also have multiple owners.

Vault member

A vault member is a user or service who can access a vault created by another user or service.

Vault administrator

Vault administrators have unrestricted access to all vaults and are allowed to perform all vault operations.

Note

Symmetric and asymmetric vaults are protected with a password or key and apply special access control rules (see [Section 18.1.2, “Standard, Symmetric, and Asymmetric Vaults”](#)). The administrator must meet these rules to:

- » access secrets in symmetric and asymmetric vaults
- » change or reset the vault password or key

A vault administrator is any user with the **Vault Administrators** privilege. See [Section 28.4, “Defining Role-Based Access Controls”](#) for information on defining user privileges.

Certain owner and member privileges depend on the type of the vault. See [Section 18.1.2, “Standard, Symmetric, and Asymmetric Vaults”](#) for details.

Vault User

The output of some commands, such as the **ipa vault-show** command, also displays **Vault user** for user vaults:

```
$ ipa vault-show my_vault
Vault name: my_vault
Type: standard
Owner users: user
Vault user: user
```

The vault user represents the user in whose container the vault is located. For details on vault containers and user vaults, see [Section 18.1.4, “Vault Containers”](#) and [Section 18.1.3, “User, Service, and Shared Vaults”](#).

18.1.2. Standard, Symmetric, and Asymmetric Vaults

The following vault types are based on the level of security and access control:

Standard vault

Vault owners and vault members can archive and retrieve the secrets without having to use a password or key.

Symmetric vault

Secrets in the vault are protected with a symmetric key. Vault members and vault owners can archive and retrieve the secrets, but they must provide the vault password.

Asymmetric vault

Secrets in the vault are protected with an asymmetric key. Users archive the secret using a public key and retrieve it using a private key. Vault members can only archive secrets, while vault owners can both archive and retrieve secrets.

18.1.3. User, Service, and Shared Vaults

The following vault types are based on ownership:

User vault: a private vault for a user

Owner: a single user.

Any user can own one or more user vaults.

Service vault: a private vault for a service

Owner: a single service.

Any service can own one or more service vaults.

Shared vault

Owner: the vault administrator who created the vault. Other vault administrators also have full access to the vault.

Shared vaults can be used by multiple users or services.

18.1.4. Vault Containers

A vault container is a collection of vaults.

IdM provides the following default vault containers:

User container: a private container for a user

This container stores: user vaults for a particular user.

Service container: a private container for a service

This container stores: service vaults for a particular service.

Shared container

This container stores: vaults that can be shared by multiple users or services.

IdM creates user and service containers for each user or service automatically when the first private vault for the user or service is created. After the user or service is deleted, IdM removes the container and its contents.

18.2. Prerequisites for Using Vaults

To enable vaults, install the Key Recovery Authority (KRA) Certificate System component on one of the servers in your IdM domain:

```
# ipa-kra-install
```

18.3. Getting Help for Vault Commands

To display all commands used to manage vaults and vault containers:

```
$ ipa help vault
```

To display detailed help for a particular command, add the **--help** option to the command:

```
$ ipa vault-add --help
```

Vault Commands Fail with vault not found Error

Some commands require you to specify the owner or the type of the vault using the following options:

- » **--user** or **--service** specify the owner of the vault you want to view

```
$ ipa vault-show user_vault --user user
```

- » **--shared** specify that the vault you want to view is a shared vault

For example, if you attempt to view another user's vault without adding **--user**, IdM informs you it did not find the vault:

```
[admin@server ~]$ ipa vault-show user_vault
ipa: ERROR: user_vault: vault not found
```

18.4. Storing a User's Personal Secret

Outcome: A user creates one or more private vaults to securely store personal secrets. The user then retrieves the secrets when required, on any machine in the domain. For example, the user can archive a personal certificate in a vault, thus storing the certificate securely in a centralized location.

This section includes these procedures:

- » [Section 18.4.1, “Archiving a User's Personal Secret”](#)
- » [Section 18.4.2, “Retrieving a User's Personal Secret”](#)

In the procedures:

- » **user** is the user who wants to create the vault
- » **my_vault** is the vault used to store the user's certificate
- » the vault type is **standard**, so that accessing the archived certificate does not require the user to provide a vault password
- » **secret.txt** is the file containing the certificate that the user wants to store in the vault
- » **secret_exported.txt** is the file to which the user exports the archived certificate

18.4.1. Archiving a User's Personal Secret

Create a private user vault and store your certificate in it. The vault type is standard, which ensures you will not be required to authenticate when accessing the certificate.

1. Log in as **user**:

```
$ kinit user
```

2. Use the **ipa vault-add** command to create a standard vault:

```
$ ipa vault-add my_vault --type standard
-----
Added vault "my_vault"
-----
Vault name: my_vault
Type: standard
Owner users: user
Vault user: user
```

3. Use the **ipa vault-archive --in** command to archive the **secret.txt** file into the vault:

```
$ ipa vault-archive my_vault --in secret.txt
-----
Archived data into vault "my_vault"
-----
```



Note

One vault can only store one secret.

18.4.2. Retrieving a User's Personal Secret

Export the certificate from your private standard vault.

1. Log in as **user**:

```
$ kinit user
```

2. Use the **ipa vault-retrieve --out** command to retrieve the contents of the vault and save them into the **secret_exported.txt** file.

```
$ ipa vault-retrieve my_vault --out secret_exported.txt
-----
Retrieved data from vault "my_vault"
-----
```

18.5. Storing a Service Secret in a Vault

Outcome: An administrator uses vaults to securely store a service secret in a centralized location. The service secret is encrypted with the service public key. The service then retrieves the secret using its private key on any machine in the domain. Only the service and the administrator are allowed to access the secret.

This section includes these procedures:

- » [Section 18.5.1, “Creating a User Vault to Store a Service Password”](#)
- » [Section 18.5.2, “Provisioning a Service Password from a User Vault to Service Instances”](#)
- » [Section 18.5.3, “Retrieving a Service Password for a Service Instance”](#)
- » [Section 18.5.4, “Changing Service Vault Password”](#)

In the procedures:

- » **admin** is the administrator who manages the service password
- » **http_password** is the name of the private user vault created by the administrator
- » **password.txt** is the file containing the service password
- » **password_vault** is the vault created for the service
- » **HTTP/server.example.com** is the service whose password is being archived
- » **service-public.pem** is the service public key used to encrypt the password stored in **password_vault**

18.5.1. Creating a User Vault to Store a Service Password

Create an administrator-owned user vault, and use it to store the service password. The vault type is standard, which ensures the administrator is not required to authenticate when accessing the contents of the vault.

1. Log in as the administrator:

```
$ kinit admin
```

2. Create a standard user vault:

```
$ ipa vault-add http_password --type standard
-----
Added vault "http_password"
-----
Vault name: http_password
Type: standard
Owner users: admin
Vault user: admin
```

3. Archive the service password into the vault:

```
$ ipa vault-archive http_password --in password.txt
-----
Archived data into vault "http_password"
-----
```

18.5.2. Provisioning a Service Password from a User Vault to Service Instances

Using an asymmetric vault created for the service, provision the service password to a service instance.

1. Log in as the administrator:

```
$ kinit admin
```

2. Obtain the public key of the service instance. For example, using the **openssl** utility:

- a. Generate the **service-private.pem** private key.

```
$ openssl genrsa -out service-private.pem 2048
Generating RSA private key, 2048 bit long modulus
.+++
.....+++e is 65537 (0x10001)
```

- b. Generate the **service-public.pem** public key based on the private key.

```
$ openssl rsa -in service-private.pem -out service-public.pem
-pubout
writing RSA key
```

3. Create an asymmetric vault as the service instance vault, and provide the public key:

```
$ ipa vault-add password_vault --service HTTP/server.example.com -
-type asymmetric --public-key-file service-public.pem
-----
Added vault "password_vault"
-----
Vault name: password_vault
Type: asymmetric
Public key: LS0tLS1C...S0tLS0tCg==
Owner users: admin
Vault service: HTTP/server.example.com@EXAMPLE.COM
```

The password archived into the vault will be protected with the key.

4. Retrieve the service password from the administrator's private vault, and then archive it into the new service vault:

```
$ ipa vault-retrieve http_password --out password.txt
-----
Retrieved data from vault "http_password"
-----
```

```
$ ipa vault-archive password_vault --service
```

```
HTTP/server.example.com --in password.txt
-----
```

```
Archived data into vault "password_vault"
-----
```

This encrypts the password with the service instance public key.

Repeat these steps for every service instance that requires the password. Create a new asymmetric vault for each service instance.

18.5.3. Retrieving a Service Password for a Service Instance

A service instance can retrieve the service vault password using the locally-stored service private key.

1. Log in as the administrator:

```
$ kinit admin
```

2. Obtain a Keberos ticket for the service:

```
# kinit HTTP/server.example.com -k -t /etc/httpd/conf/ipa.keytab
```

3. Retrieve the service vault password:

```
$ ipa vault-retrieve password_vault --service
HTTP/server.example.com --private-key-file service-private.pem --
out password.txt
-----
Retrieved data from vault "password_vault"
```

18.5.4. Changing Service Vault Password

If a service instance is compromised, isolate it by changing the service vault password and then re-provisioning the new password to non-compromised service instances only.

1. Archive the new password in the administrator's user vault:

```
$ ipa vault-archive http_password --in new_password.txt
-----
Archived data into vault "http_password"
```

This overwrites the current password stored in the vault.

2. Re-provision the new password to each service instance excluding the compromised instance.

- a. Retrieve the new password from the administrator's vault:

```
$ ipa vault-retrieve http_password --out password.txt
-----
Retrieved data from vault "http_password"
-----
```

- b. Archive the new password into the service instance vault:

```
$ ipa vault-archive password_vault --service
HTTP/server.example.com --in password.txt
-----
Archived data into vault "password_vault"
-----
```

18.6. Storing a Common Secret for Multiple Users

Outcome: An administrator creates a shared vault and allows other users to access the secret in the vault. The administrator archives a common password into the vault, and the other users are able to retrieve the password on any machine in the domain.

This section includes these procedures:

- » [Section 18.6.2, “Retrieving a Secret from a Shared Vault as a Member User”](#)
- » [Section 18.6.1, “Creating the Shared Vault with the Common Secret”](#)

In the procedures:

- » **shared_vault** is the vault used to store the common password
- » **admin** is the administrator who creates the shared vault
- » the vault type is **standard**, so that accessing the archived password does not require the user to provide a vault password
- » **secret.txt** is the file containing the common secret
- » **user1** and **user2** are the users allowed to access the vault

18.6.1. Creating the Shared Vault with the Common Secret

Create a shared vault and use it to store the common secret. Add the users who will be accessing the secret as vault members. The vault type is standard, which ensures any user accessing the secret will not be required to authenticate.

1. Log in as the administrator:

```
$ kinit admin
```

2. Create the shared vault:

```
$ ipa vault-add shared_vault --shared --type standard
-----
Added vault "shared_vault"
-----
```

```
Vault name: shared_vault
Type: standard
Owner users: admin
Shared vault: True
```

- Archive the secret into the vault. Add the **--shared** option to specify that the vault is in the shared container:

```
$ ipa vault-archive shared_vault --shared --in secret.txt
-----
Archived data into vault "shared_vault"
-----
```



Note

One vault can only store one secret.

- Add **user1** and **user2** as vault members:

```
ipa vault-add-member shared_vault --shared --users={user1,user2}
Vault name: shared_vault
Type: standard
Owner users: admin
Shared vault: True
Member users: user1, user2
-----
Number of members added 2
-----
```

18.6.2. Retrieving a Secret from a Shared Vault as a Member User

Log in as a member user of the vault, and export the file with the secret from the vault.

- Log in as the **user1** member user:

```
$ kinit user1
```

- Retrieve the secret from the shared vault:

```
$ ipa vault-retrieve shared_vault --shared --out
secret_exported.txt
-----
Retrieved data from vault "shared_vault"
-----
```

Chapter 19. 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.

19.1. 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.

19.2. 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
```

19.3. 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. 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.

19.3.1. Adding Netgroups

19.3.1.1. With the Web UI

1. Open the **Identity** tab, and select the **Netgroups** subtab.
2. Click **Add** at the top of the netgroups list.

<input type="checkbox"/>	Netgroup name	Description
<input type="checkbox"/>	server.example.com	An example

Showing 1 to 1 of 1 entries.

Figure 19.1. Netgroups List

3. Enter a unique name and, optionally, a description.

The screenshot shows a dialog box titled "Add Netgroup". It contains two input fields: "Netgroup name *" with the value "test-nis" and "Description" with the value "for existing nis groups". Below the fields is a note "* Required field". At the bottom are four buttons: "Add", "Add and Add Another", "Add and Edit", and "Cancel".

Figure 19.2. Add Netgroup Dialogue

The group name is the identifier used for the netgroup in the IdM domain, and it cannot be changed after it is created. The name cannot contain spaces, but other separators like an underscore (_) are allowed.

4. Click the **Add and Edit** button to go immediately to the netgroup's edit pages.
5. Optionally, set the NIS domain for the netgroup. This defaults to the IdM domain, but it can be changed.
 - a. Click the name of the group you wish to edit.
 - b. In the *General* part of the settings, enter the name of the alternate NIS domain in the **NIS domain name** field.

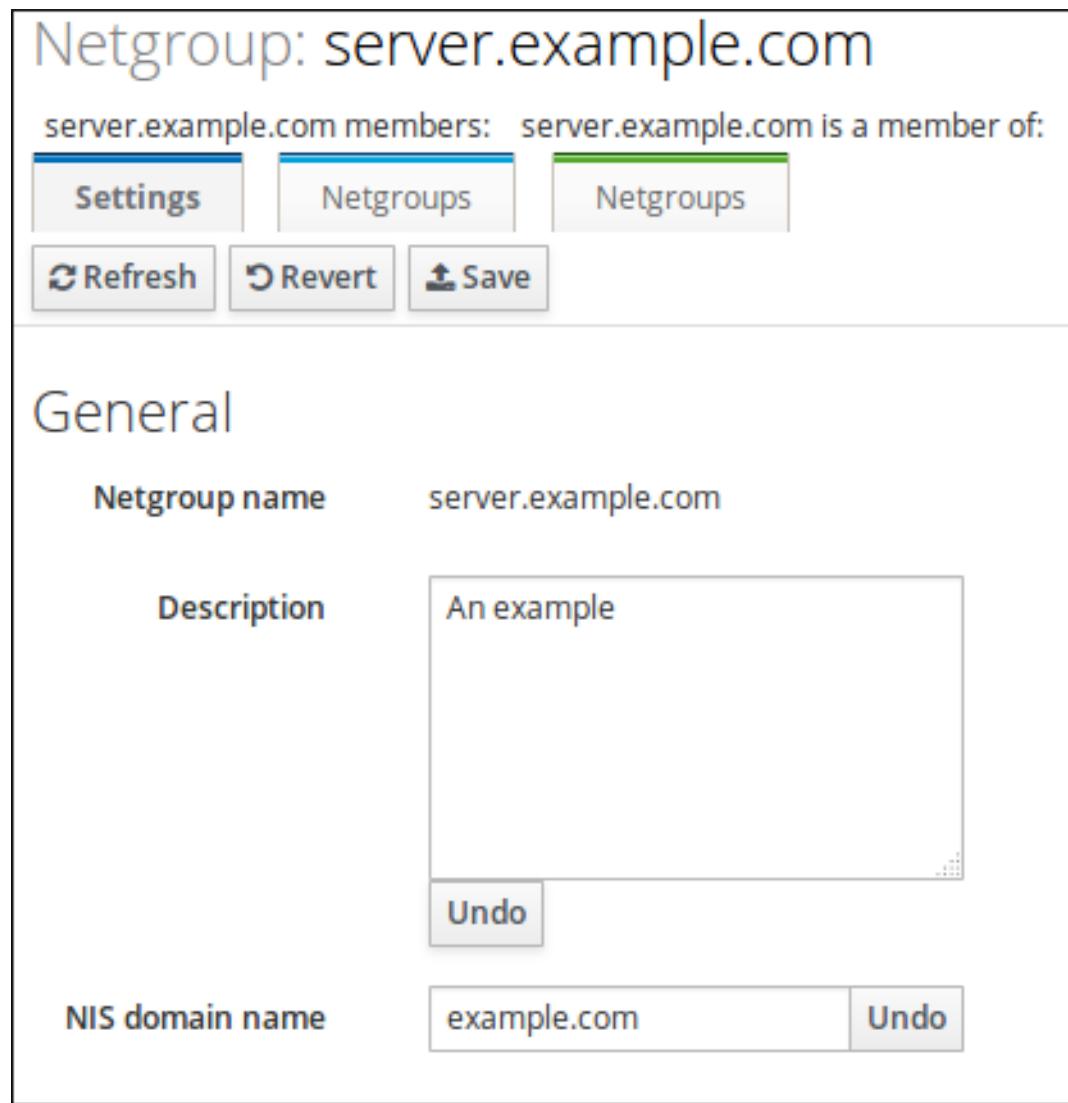


Figure 19.3. Netgroup Tab

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 19.3.2.1, “With the Web UI”](#).

19.3.1.2. 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.

19.3.2. 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.

19.3.2.1. 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.

<input type="checkbox"/>	Netgroup name	Description
<input type="checkbox"/>	server.example.com	An example

Showing 1 to 1 of 1 entries.

Figure 19.4. Netgroups List

3. Choose the type of netgroup member to add. Click **Add** by the list of the netgroup members.

Netgroup: server.example.com

server.example.com members: server.example.com is a member of:

User

User category the rule applies to: Anyone Specified Users and Groups

<input type="checkbox"/> Users	<input type="checkbox"/> Delete	<input type="button" value="+ Add"/>
<input type="checkbox"/> User Groups	<input type="checkbox"/> Delete	<input type="button" value="+ Add"/>

Figure 19.5. User Menu in the Netgroup Tab

4. Click the checkbox by the names of the users to add, and click the right arrow button, >, to move the names to the selection box.

Add Users into Netgroup server.example.com

Filter available Users

Available		Prospective	
<input type="checkbox"/> Users	>	<input type="checkbox"/> Users	
<input type="checkbox"/> admin			
<input type="checkbox"/> employee			
<input type="checkbox"/> helpdesk			
<input type="checkbox"/> manager			
<input type="checkbox"/> test			
<input type="checkbox"/> test_1			

Figure 19.6. Add User Menu in the Netgroup Tab

5. Click **Add**.

19.3.2.2. 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
```

19.4. Exposing Automount Maps to NIS Clients

When the NIS service is enabled on a system, the IdM server is automatically configured to set the NIS domain to the IdM domain's name, and to include IdM users, groups, and netgroups as passwd, group, and netgroup maps in the NIS domain.

If any automount maps are already defined, these maps need to be manually added to the NIS configuration in Identity Management for them to be exposed to NIS clients. The NIS server is managed by a special plug-in entry in the IdM LDAP directory; this is a container entry, and each NIS domain and map used by the NIS server is configured as a child entry beneath that container. The NIS domain entry must contain:

- » the name of the NIS domain
- » the name of the NIS map
- » information on how to find the directory entries to use as the NIS map's contents
- » information on 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.

19.5. 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.

19.5.1. 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 19.3, “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.

19.5.2. 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
```

19.5.3. 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.

19.5.3.1. 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.

19.5.3.2. 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
```

19.5.3.3. 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.

19.5.3.4. 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 19.1, “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
```

19.5.3.5. 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; a script 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=" "
    if [[ $line =~ ^([^\s]+)\s+(\S+)=(\S+)$ ]]; then
        key=${!1}
        value=${!2}
        map=${!3}
        ipa automountkey-add $1 $key $value
        ipa automountmapkey-add $1 $key $map
    fi
done
```

```

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

```

19.5.4. 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 20. Managing DNS

An Identity Management server can be installed without integrated DNS services so that it uses an external DNS service or with DNS configured. See [Section 2.3, “Installing an IdM Server”](#) and [Section 2.3.1, “Determining Whether to Use Integrated DNS”](#) for details.

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.

20.1. 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 with the `ipa-server-install` utility, as described in [Section 2.3.3, “Installing a Server with Integrated DNS”](#).

For more information about `ipa-dns-install`, see the `ipa-dns-install(1)` man page.

20.1.1. Setting up Additional Name Servers

IdM adds the newly-configured IdM DNS server to the list of DNS servers in the `/etc/resolv.conf` file. It is recommended to manually add other DNS servers as backup servers in case the IdM server becomes unavailable. For example:

```
search example.com

; the IdM server
nameserver 192.0.2.1

; backup DNS servers
nameserver 198.51.100.1
nameserver 198.51.100.2
```

For more details about configuring `/etc/resolv.conf`, see the `resolv.conf(5)` man page.

20.2. 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 [4]. 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.

20.3. 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 20.1. 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.

20.4. 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.

20.5. Managing Master DNS Zones

20.5.1. 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.

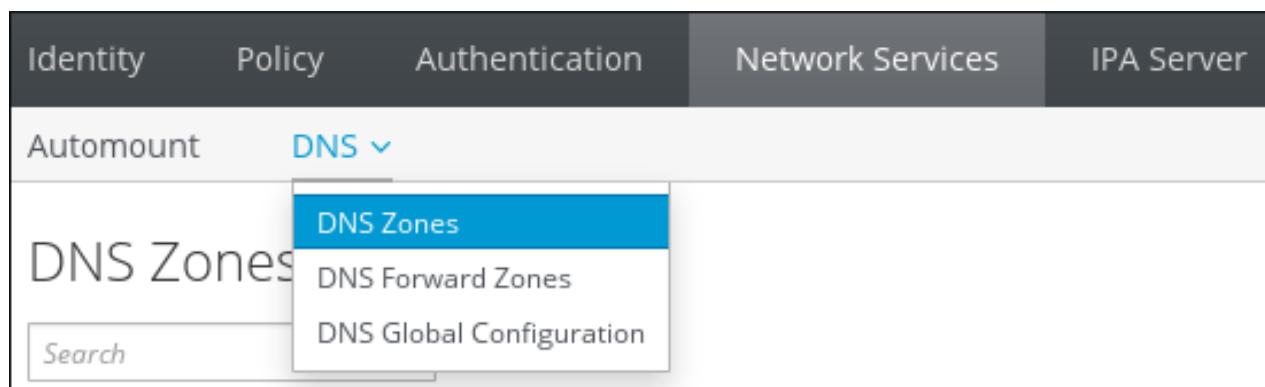


Figure 20.1. Managing DNS Master Zones

2. To add a new master zone, click **Add** at the top of the list of all zones.

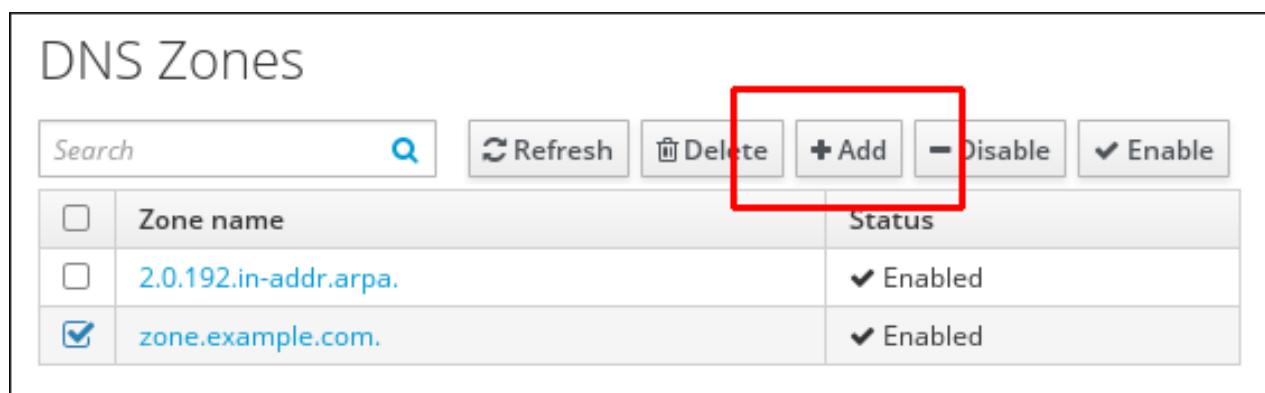


Figure 20.2. Adding a Master DNS Zone

3. Provide the zone name, and click **Add**.

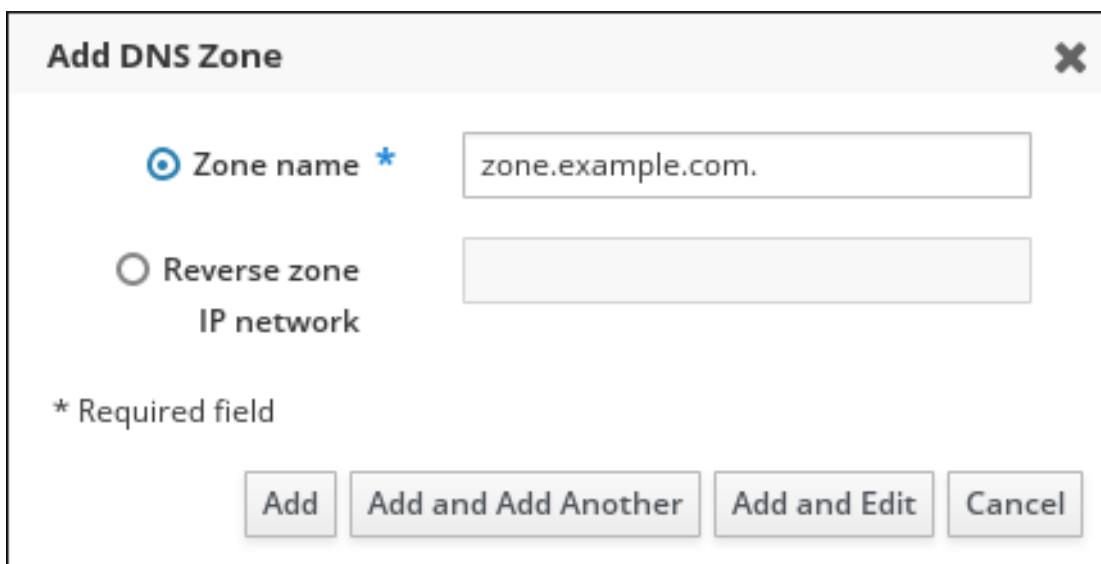


Figure 20.3. Entering a New Master Zone

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**.

DNS Zones		Search	Refresh	Delete	Add	Disable	Enable
<input type="checkbox"/>	Zone name						Status
<input type="checkbox"/>	2.0.192.in-addr.arpa.					<input checked="" type="checkbox"/> Enabled	
<input checked="" type="checkbox"/>	zone.example.com.					<input checked="" type="checkbox"/> Enabled	

Figure 20.4. 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
```

20.5.2. 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 20.1, “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 20.1. Zone Attributes

Attribute	Command-Line Option	Description
Authoritative name server	--name-server	Sets the domain name of the master DNS name server, also known as SOA MNAME. By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using --name-server is ignored.
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 RFC 2308 .
SOA time to live	--ttl	Sets TTL in seconds for records at zone apex. In zone example.com , for instance, all records (A, NS, or SOA) under name example.com are configured, but no other domain names, like test.example.com , are affected.

Attribute	Command-Line Option	Description
BIND update policy	--update-policy	Sets the permissions allowed to clients in the DNS zone. See Dynamic Update Policies in the BIND 9 Administrator Reference Manual for more information on update policy syntax.
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 Section 20.6.1, “Enabling Dynamic DNS Updates” for more information.
Allow transfer	--allow-transfer= <i>string</i>	Gives a semi-colon-separated list of IP addresses or network names which are allowed to transfer the given zone. Zone transfers are disabled by default. The default --allow-transfer value is none .
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=none only first	Specifies the forward policy. For information about the supported policies, see Section 20.7, “Forward Policies”

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.

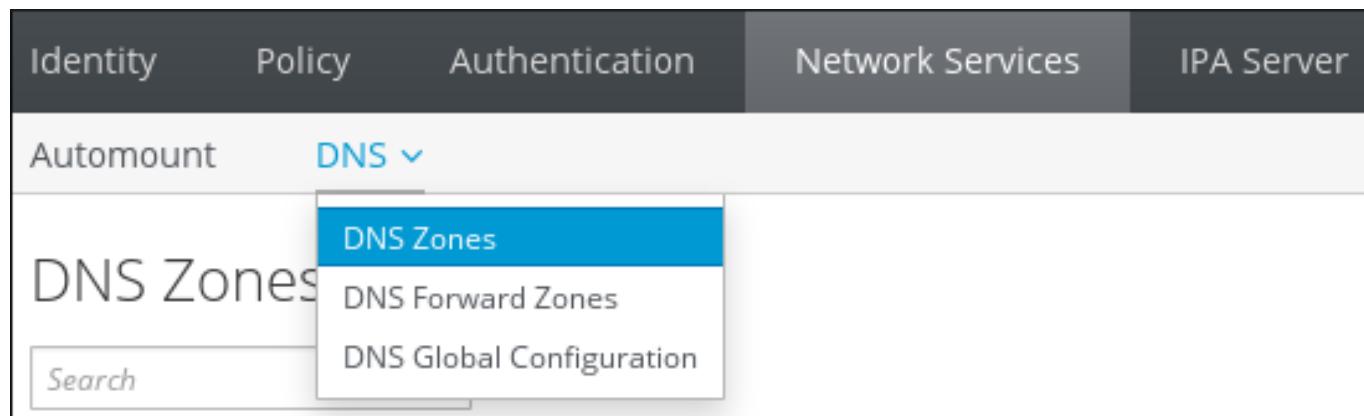


Figure 20.5. DNS Master Zones Management

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.

<input type="checkbox"/>	Zone name	Status
<input type="checkbox"/>	2.0.192.in-addr.arpa.	✓ Enabled
<input type="checkbox"/>	zone.example.com.	✓ Enabled

Figure 20.6. Editing a Master Zone

2. Click **Settings**, and then change the zone configuration as required.

The screenshot shows a top-level title '✓ DNS Zone: zone.example.com.' followed by a navigation bar with tabs: DNS Resource Records, Settings (which is highlighted with a red box), Refresh, Revert, Save, and Actions. Below this is a section titled 'DNS Zone Settings' with a 'Zone name' field containing 'zone.example.com.'

Figure 20.7. The Settings Tab in the Master Zone Edit Page

For information about the available settings, see [Table 20.1, “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 20.1, “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.

20.5.3. 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 20.5.2, “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	<input type="text" value="192.0.2.1"/> Undo
	<input type="text" value="198.51.100.1"/> Undo
	<input type="text" value="203.0.113.1"/> Undo
	<input type="button" value="Add"/> <input type="button" value="Undo All"/>

Figure 20.8. 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
```

20.5.4. 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 host name value must be a canonical host name of the host for which you want to create the record. For more information, see [Example 20.6, “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 20.5.2, “Editing the Zone Configuration in the Web UI”](#).
2. In the **DNS Resource Records** section, click **Add** to add a new record.

DNS Resource Records: zone.example.com.

DNS Resource Records		Settings	
<input style="width: 100%; height: 30px; border: 1px solid #ccc; margin-bottom: 5px;" type="text"/> Search		 	
	Record name	Record Type	Data
<input type="checkbox"/>	@	NS	example.com.
<input type="checkbox"/>	dns	CNAME	dns.example.com.

Showing 1 to 2 of 2 entries.

Figure 20.9. Adding a New DNS Resource Record

3. Select the type of record to create and fill out the other fields as required.

The dialog box has a title bar 'Add DNS Resource Record' with a close button. It contains three input fields: 'Record name *' with value 'dns', 'Record Type' with dropdown value 'CNAME', and 'Hostname *' with value 'dns.example.com.'. Below these fields is a note '* Required field'. At the bottom are four buttons: 'Add' (highlighted in grey), 'Add and Add Another', 'Add and Edit', and 'Cancel'.

Figure 20.10. 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 zone_name record_name --record_type_option=data
```

The *zone_name* is the name of the DNS zone to which the record is being added. The *record_name* is an identifier for the new DNS resource record.

[Table 20.2, “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 20.2. Common ipa dnsrecord-add Options

General Record Options	
Option	Description
--ttl=number	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=ARECORD	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=AAAARECORD	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=PTRRECORD	Passes a list of PTR records.
--ptr-hostname= <i>string</i>	Gives the hostname for the record.
"SRV" Record Options	
Option	Description
--srv-rec=SRVRECORD	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-port= <i>number</i>	Gives the port for the service on the target host.
--srv-target= <i>string</i>	Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.

20.5.5. Examples of Adding or Modifying DNS Resource Records from the Command Line

Example 20.2. 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 20.3. 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-rec` 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 20.4. 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 20.5. Adding an SRV Record

In the following example, `_ldap._tcp` defines the service type and the connection protocol for the SRV record. The `--srv-rec` option defines the priority, weight, port, and target values.

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 (**51** and **49** in this example) add up to 100 and represent the probability (in percentages) that a particular record is used.

Example 20.6. 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 the 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.
```

The next example adds a reverse DNS entry to the **0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa**. IPv6 reverse zone for the host **server2.example.com** with the IP address **2001:DB8::1111**:

```
$ ipa dnsrecord-add 0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.  
1.1.1.0.0.0.0.0.0.0.0.0.0 --ptr-rec server2.example.com.
```

20.5.6. 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 20.5.2, “Editing the Zone Configuration in the Web UI”](#).
2. In the **DNS Resource Records** section, click the name of the resource record.

DNS Resource Records: zone.example.com.

DNS Resource Records		Settings	
Search <input type="text"/> <input type="button" value="🔍"/>		<input type="button" value="⟳ Refresh"/> <input type="button" value="Delete"/> <input type="button" value="+ Add"/>	
<input type="checkbox"/>	Record name	Record Type	Data
<input type="checkbox"/>	@	NS	example.com.
<input checked="" type="checkbox"/>	dns	CNAME	dns.example.com.

Showing 1 to 2 of 2 entries.

Figure 20.11. Selecting a DNS Resource Record

3. Select the check box by the name of the record type to delete.

Standard Record Types

A	<input type="checkbox"/> IP Address	<input type="button" value="Delete"/> <input type="button" value="+ Add"/>
AAAA	<input type="checkbox"/> IP Address	<input type="button" value="Delete"/> <input type="button" value="+ Add"/>
CNAME	<input type="checkbox"/> Hostname	<input type="button" value="Delete"/> <input type="button" value="+ Add"/>
	<input checked="" type="checkbox"/> dns.example.com.	<input type="button" value="Edit"/>

Figure 20.12. 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 20.5.2, “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.

The screenshot shows a table of DNS resource records. The columns are 'Record name', 'Record Type', and 'Data'. There are three rows: one for the '@' record (NS type, example.com) and one for the 'dns' record (CNAME type, dns.example.com). A red box highlights the 'Delete' button in the top right corner of the table header area.

Record name	Record Type	Data
@	NS	example.com.
<input checked="" type="checkbox"/> dns	CNAME	dns.example.com.

Figure 20.13. 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.

20.5.7. Disabling and Enabling Zones

IdM allows the administrator to disable and enable DNS zones. While deleting a DNS zone, described in [Section 20.5.1, “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.

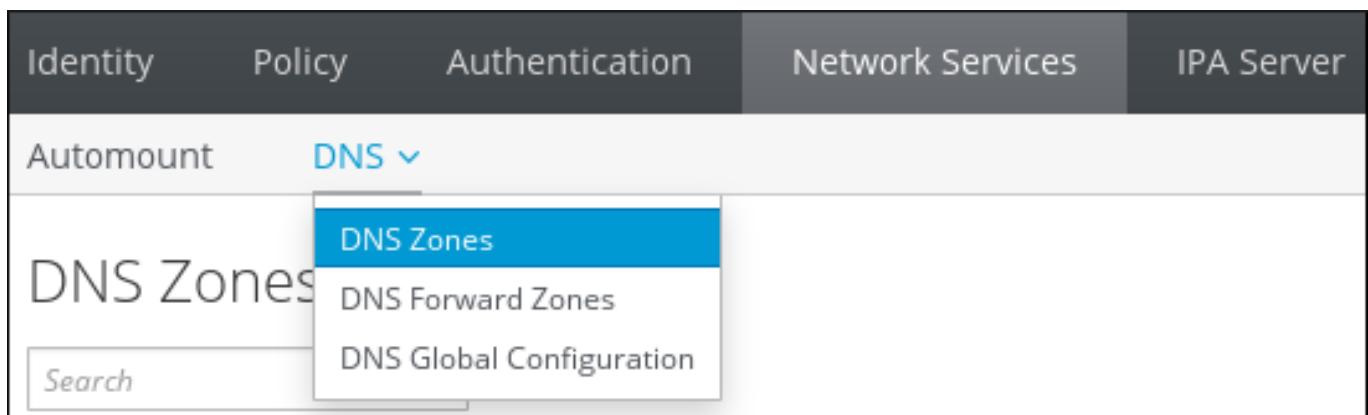


Figure 20.14. Managing DNS Zones

To disable a zone, select the check box next to the zone name and click **Disable**.

DNS Zones			
<input type="text" value="Search"/>  Refresh Delete + Add		— Disable ✓ Enable	
<input type="checkbox"/>	Zone name		Status
<input type="checkbox"/>	2.0.192.in-addr.arpa.		✓ Enabled
<input checked="" type="checkbox"/>	zone.example.com.		✓ Enabled

Figure 20.15. 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.

20.6. Managing Dynamic DNS Updates

20.6.1. 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

20.6.1.1. 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.

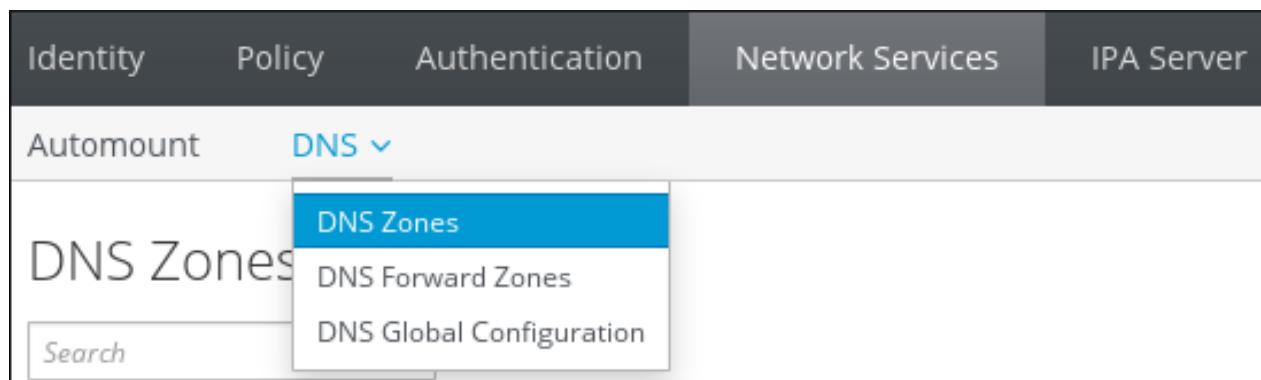


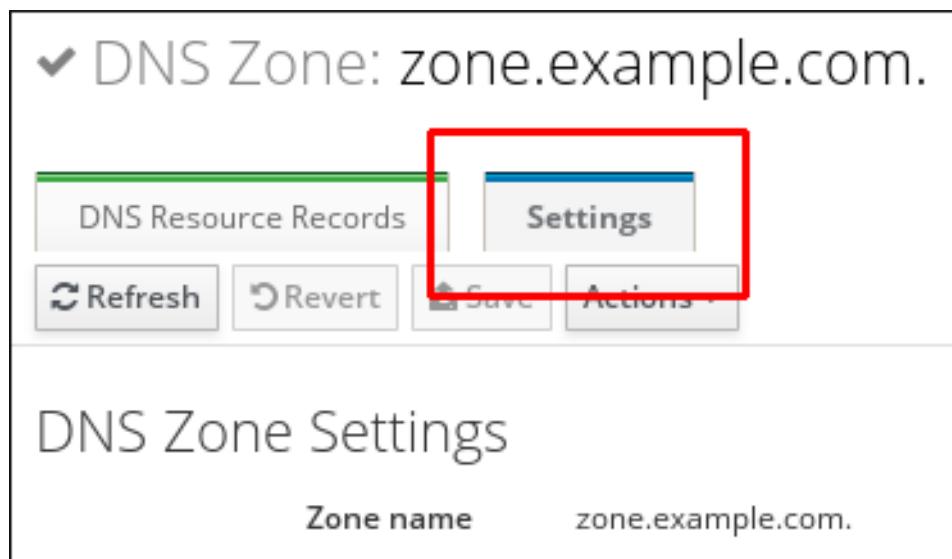
Figure 20.16. DNS Zone Management

2. Click on the zone name in the list of all zones to open the DNS zone page.

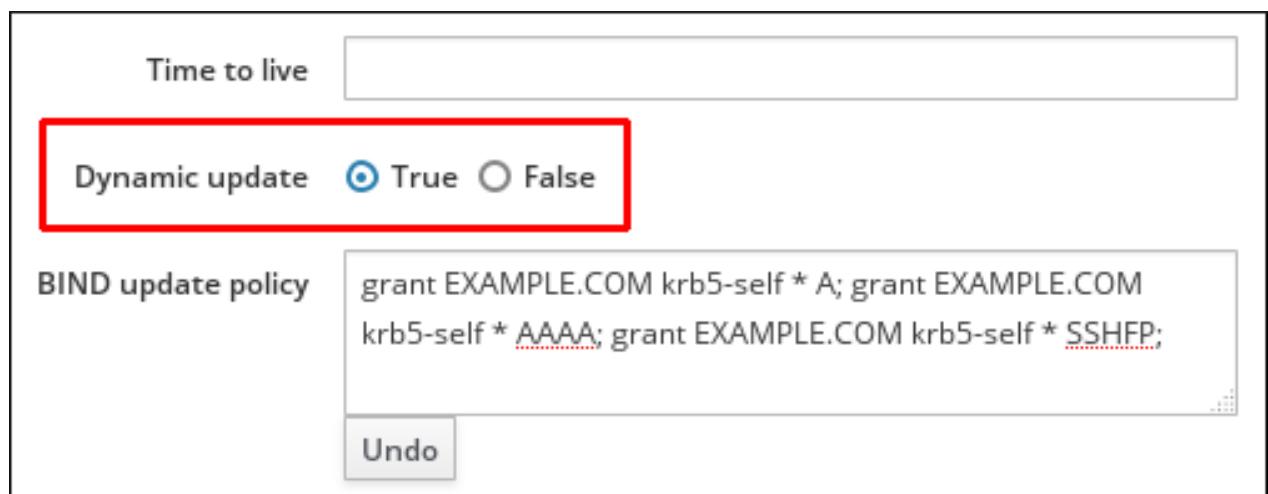
DNS Zones		
<input type="button" value="Search"/> <input type="button" value="Refresh"/> <input type="button" value="Delete"/> <input type="button" value="Add"/> <input type="button" value="Disable"/> <input type="button" value="Enable"/>		Status
<input type="checkbox"/>	Zone name	
<input type="checkbox"/>	2.0.192.in-addr.arpa.	<input checked="" type="checkbox"/> Enabled
<input type="checkbox"/>	zone.example.com.	<input checked="" type="checkbox"/> Enabled

Figure 20.17. Editing a Master Zone

- Click **Settings** to switch to the DNS zone settings tab.

**Figure 20.18. The Settings Tab in the Master Zone Edit Page**

- Scroll down to the **Dynamic update** field, and set the value to **True**.

**Figure 20.19. Enabling Dynamic DNS Updates**

- 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
```

20.6.1.2. 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/sssd/sssd.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
```

20.6.2. 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 20.6.1, “Enabling Dynamic DNS Updates”](#).

- » PTR synchronization must be enabled for the master forward zone, not for the reverse zone.
- » 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.

The screenshot shows the Network Services tab selected in the top navigation bar. Under the DNS subtab, the DNS Zones section is active. A dropdown menu is open over the 'DNS Zones' link, showing three options: 'DNS Zones' (which is highlighted in blue), 'DNS Forward Zones', and 'DNS Global Configuration'. Below the dropdown, there is a search input field labeled 'Search'.

Figure 20.20. DNS Zone Management

2. Click on the zone name in the list of all zones to open the DNS zone page.

The screenshot shows the 'DNS Zones' page. At the top, there is a header with buttons for 'Search' (with a magnifying glass icon), 'Refresh' (with a circular arrow icon), 'Delete' (with a trash bin icon), 'Add' (with a plus sign icon), 'Disable' (with a minus sign icon), and 'Enable' (with a checkmark icon). Below the header is a table with two columns: 'Zone name' and 'Status'. There are three rows in the table. The first row has a checkbox and the zone name '2.0.192.in-addr.arpa.' with a status of 'Enabled'. The second row has a checkbox and the zone name 'zone.example.com.' with a status of 'Enabled'. The third row has a checkbox and the zone name '2.0.192.in-addr.arpa.' with a status of 'Enabled'. The row for 'zone.example.com.' has a red border around it.

Zone name	Status
2.0.192.in-addr.arpa.	Enabled
zone.example.com.	Enabled
2.0.192.in-addr.arpa.	Enabled

Figure 20.21. Editing a DNS Zone

- Click **Settings** to switch to the DNS zone settings tab.

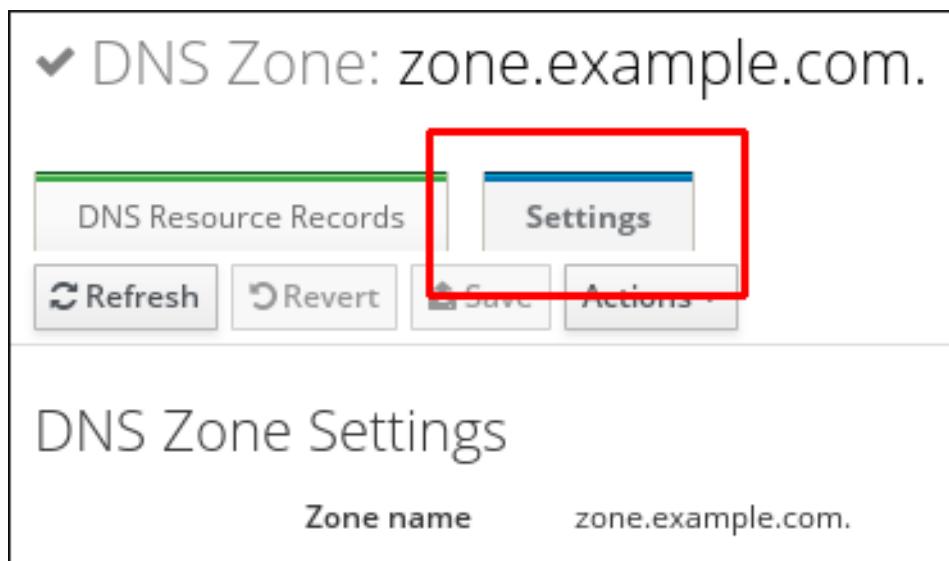


Figure 20.22. The Settings Tab in the Master Zone Edit Page

- Select the **Allow PTR sync** check box.



Figure 20.23. Enabling PTR Synchronization

- 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 zone.example.com
```

The default **--allow-sync-ptr** value is **0**, which disables synchronization.

20.6.3. Updating DNS Dynamic Update Policies

DNS domains maintained by IdM servers can accept a DNS dynamic update according to RFC 3007 [5].

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 20.6.1, “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.

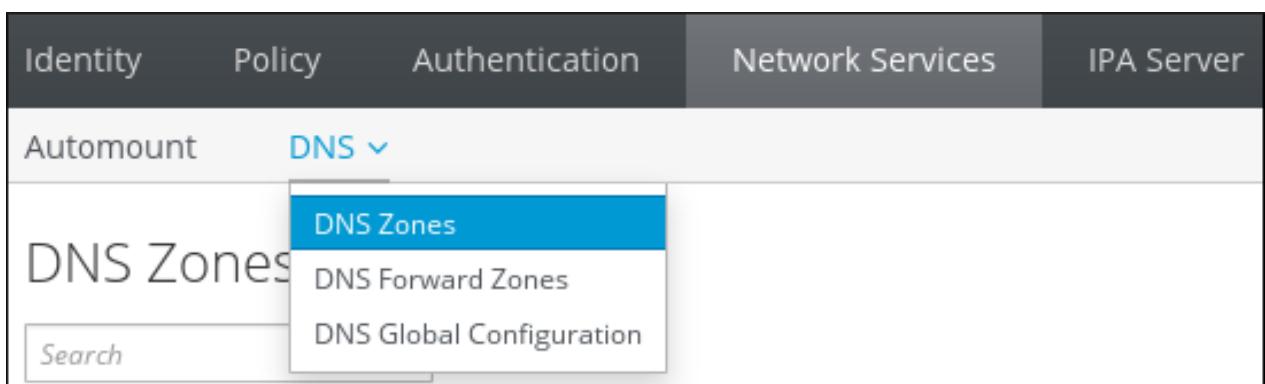


Figure 20.24. DNS Zone Management

2. Click on the zone name in the list of all zones to open the DNS zone page.

A screenshot of the 'DNS Zones' management page. The title is 'DNS Zones'. There are buttons for Search, Refresh, Delete, Add, Disable, and Enable. A table lists two zones:

	Zone name	Status
<input type="checkbox"/>	2.0.192.in-addr.arpa.	✓ Enabled
<input type="checkbox"/>	zone.example.com.	✓ Enabled

Figure 20.25. Editing a DNS Zone

3. Click **Settings** to switch to the DNS zone settings tab.

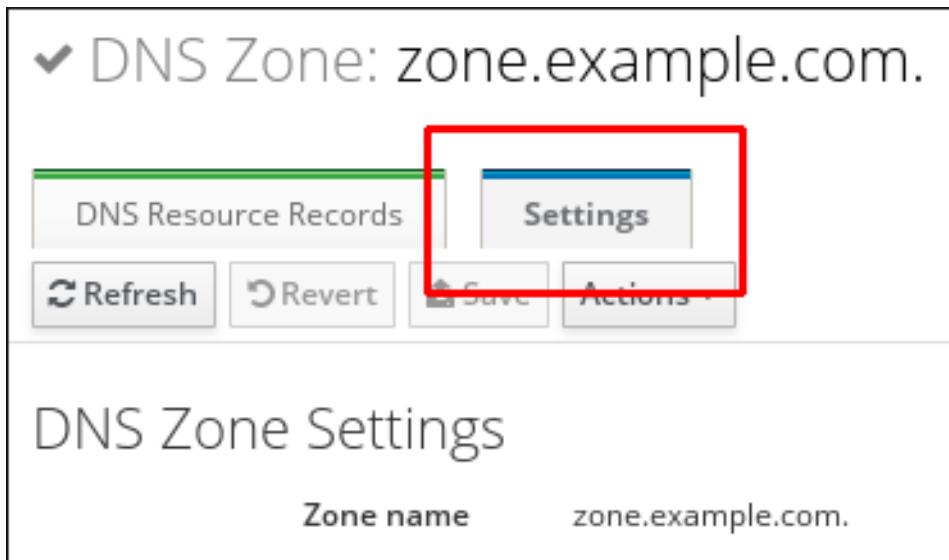


Figure 20.26. The Settings Tab in the Master Zone Edit Page

- Set the required update policies in a semi-colon separated list in the **BIND update policy** text box.

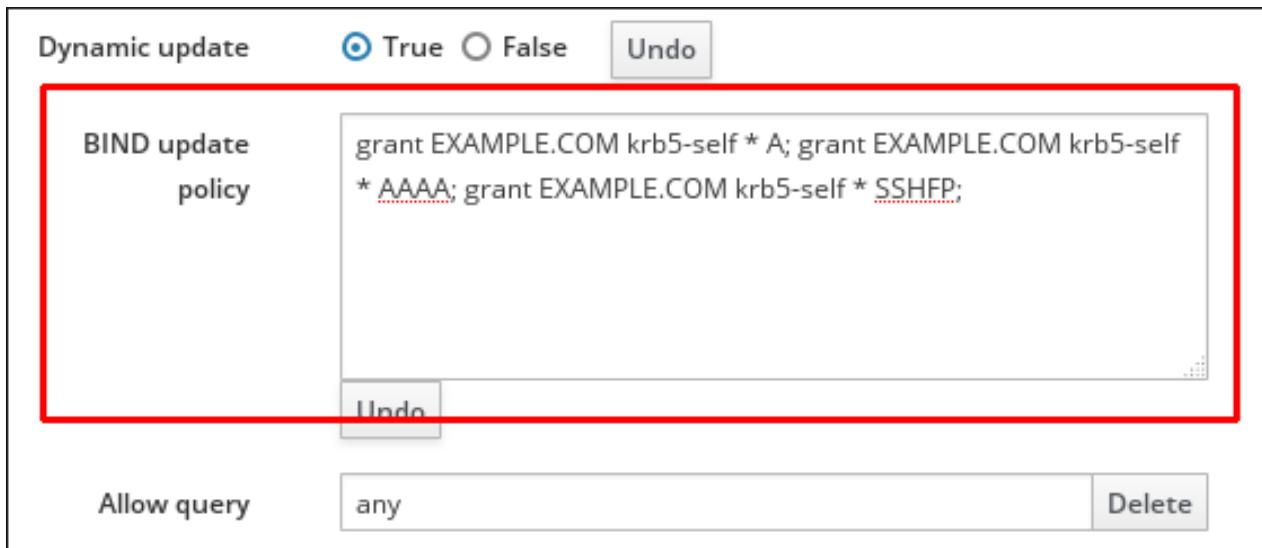


Figure 20.27. DNS Update Policy Settings

- 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 zone.example.com --update-policy "grant EXAMPLE.COM
krb5-self * A; grant EXAMPLE.COM krb5-self * AAAA; grant EXAMPLE.COM
krb5-self * SSHFP;"
```

20.7. 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 20.7. 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.

20.7.1. 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 20.7, “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 20.4, “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 shows the IPA Web UI with the 'DNS' tab selected. Under the 'Options' section, there is a checkbox for 'Allow PTR sync'. Below it, the 'Global forwarders' section contains two entries: '192.0.2.253' and '192.0.2.254', each with an 'Undo' button. There are also 'Add' and 'Undo All' buttons. At the bottom, the 'Forward policy' section has three radio buttons: 'Forward first' (selected), 'Forward only', and 'Forwarding disabled'.

Figure 20.28. Editing Global DNS Configuration in the Web UI

- 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
```

20.7.2. 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 20.7.1, “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 20.7, “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 [6].

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.

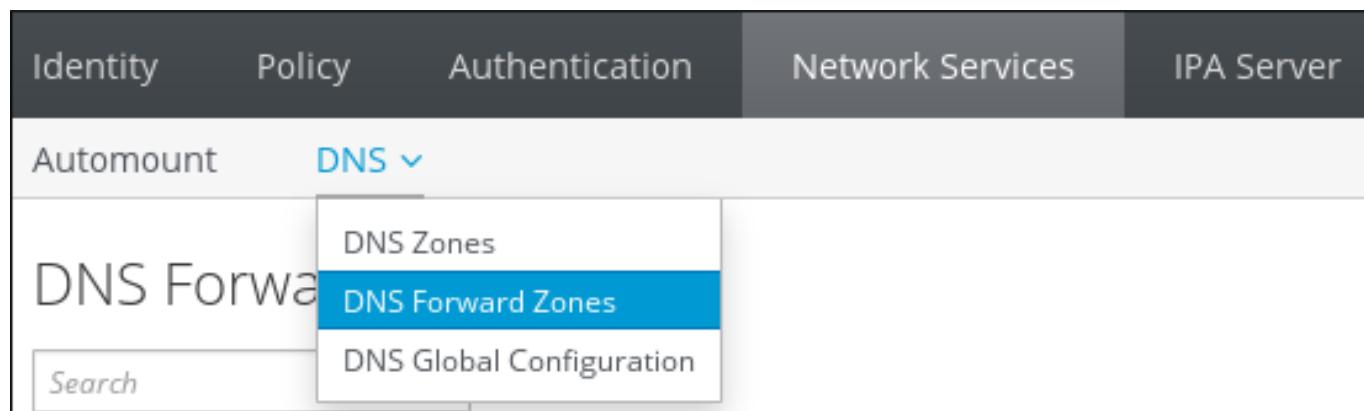


Figure 20.29. Managing DNS Forward Zones

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 20.1, “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.

20.8. 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.

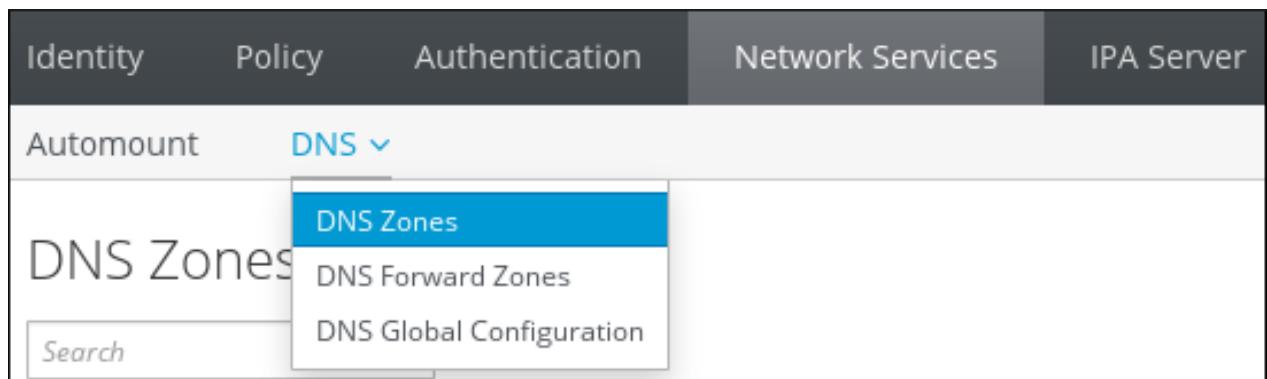


Figure 20.30. DNS Zone Management

2. Click **Add** at the top of the list of all zones.

DNS Zones	
<input type="text" value="Search"/>  Refresh  Delete  Add  Disable  Enable	
<input type="checkbox"/>	Zone name
<input type="checkbox"/>	2.0.192.in-addr.arpa.
<input checked="" type="checkbox"/>	zone.example.com.
Status	
<input checked="" type="checkbox"/>	Enabled
<input checked="" type="checkbox"/>	Enabled

Figure 20.31. 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:

Add DNS Zone

Zone name *

Reverse zone
IP network

* Required field

Figure 20.32. Creating a Reverse Zone by Name

b. Alternatively, to add a reverse DNS zone by the reverse zone IP network:

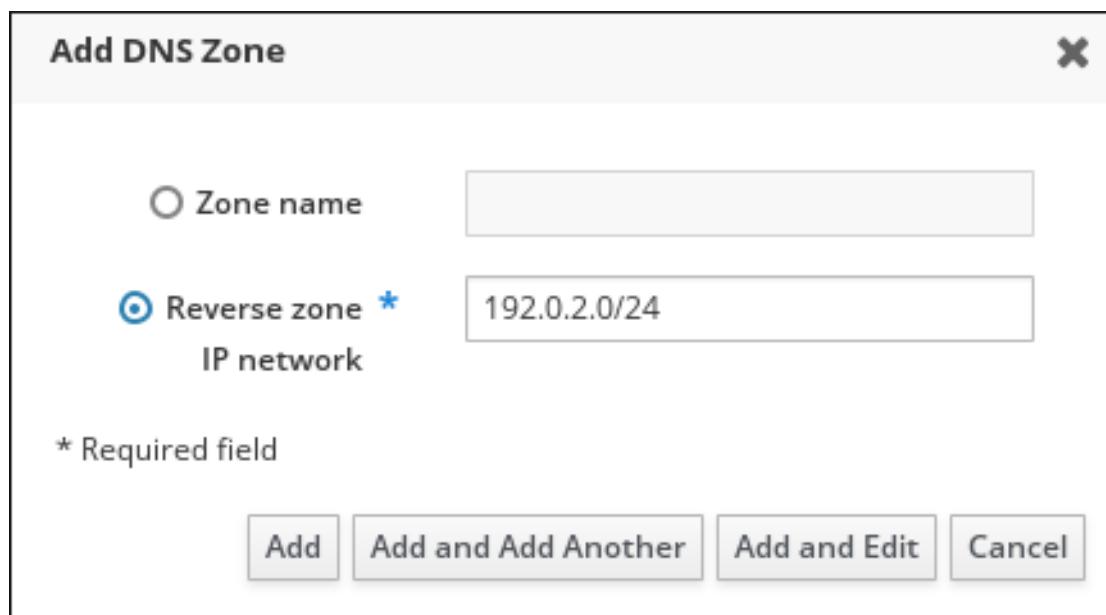


Figure 20.33. Creating a Reverse Zone by IP Network

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.

- 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 20.5, “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.

20.9. 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.

[4] For more information about GSS-TSIG, see [RFC 3545](#).

[5] For the full text of RFC 3007, see <http://tools.ietf.org/html/rfc3007>

[6] For more information, refer to the [BIND 9 Configuration Reference](#).

Part III. Defining Domain-wide System Policies

Chapter 21. 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 8.1, “Setting up User Home Directories”](#).

In IdM, automount works with the internal LDAP directory and also with DNS services if configured.

21.1. 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](#).

21.2. 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.

21.2.1. 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
```

If **--no-sssd** is used, the list of configuration files updated by **ipa-client-automount** is different:

- 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.

21.2.2. Configuring autofs Manually to Use SSSD and Identity Management

- 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"
```

- 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 21.4, “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/sssd/sssd.conf
```

- b. Add the autofs service to the list of services handled by SSSD.

```
[sssd]
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.

21.2.3. 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_VERSMAX=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=co
m
           -a
serviceSearchDescriptor=auto_master:automountMapName=auto.master,c
n=location,cn=automount,dc=example,dc=com?one
           -a
serviceSearchDescriptor=auto_home:automountMapName=auto_home,cn=lo
cation,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,c
n=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: /home
automountInformation: auto.home
```

- b. List a user's /**home** directory:

```
# ls /home/userName
```

21.3. 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.

21.3.1. 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 4.4.2, “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 15.1, “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/nfs-
server.example.com
Principal: NFS/nfs-server.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/idmapd.conf** file.

8. Edit the **/etc(exports** file and add the Kerberos information:

```
/export *(rw,sec=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 21.3.2, “Setting up a Kerberized NFS Client”](#).

21.3.2. 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 4.4.2, “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
--enablessdauth --enablemkhomedir
```

- b. Restart the OpenSSH client.

```
[root@nfs-client-server ~]# systemctl start sssh.service
```

- c. Edit the IdM domain section in the SSSD configuration file to set the keytab renewal options.

```
[root@nfs-client-server ~]# vim /etc/sssd/sssd.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
```

21.4. 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.

21.4.1. 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.

AUTOMOUNT LOCATIONS

Refresh

	Location
<input type="checkbox"/>	default
<input type="checkbox"/>	raleigh

4. Enter the name for the new location.

Add Automount Location

Location:

5. Click the **Add and Edit** button to go to the map configuration for the new location. Create maps, as described in [Section 21.5.1.1, “Configuring Direct Maps from the Web UI”](#) and [Section 21.5.2.1, “Configuring Indirect Maps from the Web UI”](#).

21.4.2. 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:
```

21.5. 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.

21.5.1. Configuring Direct Maps

Direct maps define exact locations, meaning absolute paths, to the file mount point. In the location entry, a direct map is identified by the preceding forward slash:

```
-----  
/etc/auto.direct:  
/shared/man server.example.com:/shared/man
```

21.5.1.1. 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.

The screenshot shows the 'Automount' tab selected under the 'Policy' section of the IPA Server interface. The main title is 'AUTOMOUNT LOCATIONS'. Below it are buttons for Refresh, Delete, and Add. A list of locations is shown with checkboxes: 'default' and 'raleigh'.

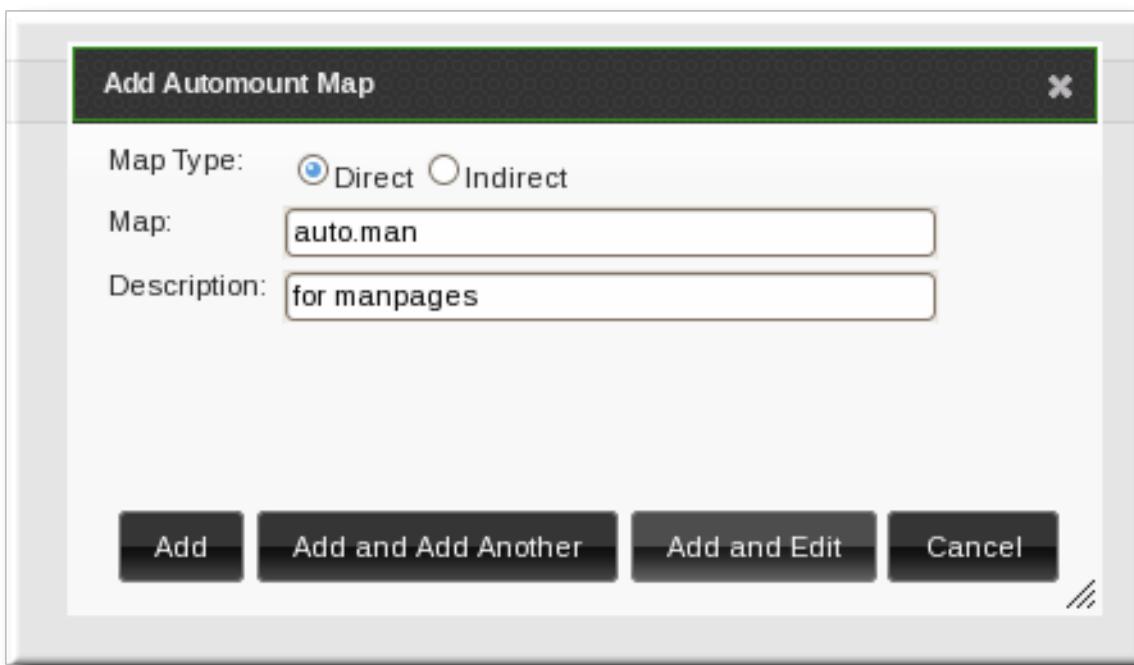
Location
<input type="checkbox"/> default
<input type="checkbox"/> raleigh

4. In the **Automount Maps** tab, click the **+ Add** link to create a new map.

The screenshot shows the 'Automount' tab selected under the 'Policy' section of the IPA Server interface. The path 'Autounmount Locations > boston' is shown above the title 'AUTOMOUNT LOCATION: boston'. Below it are tabs for 'Automount Maps' and 'Settings'. A list of maps is shown with checkboxes: 'auto.direct', 'auto.man', 'auto.master', and 'auto.share'.

Map
<input type="checkbox"/> auto.direct
<input type="checkbox"/> auto.man
<input type="checkbox"/> auto.master
<input type="checkbox"/> auto.share

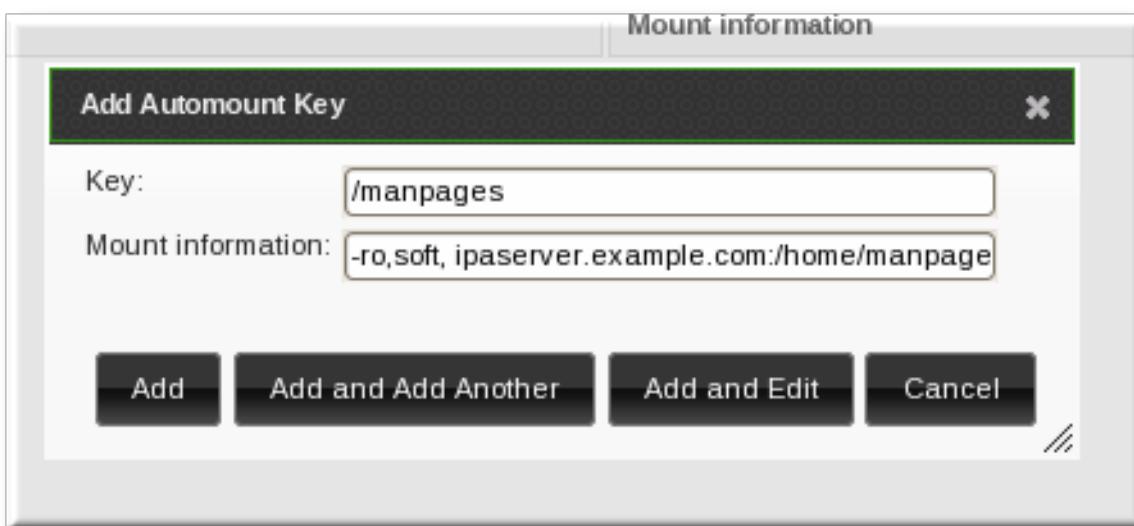
5. In pop-up window, select the **Direct** radio button and enter the name of the new map.



6. In the **Automount Keys** tab, click the **+ Add** link to create a new key for the map.

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

7. Enter the mount point. The key defines the actual mount point in the key name. The **Info** field sets the network location of the directory, as well as any **mount** options to use.



8. Click the **Add** button to save the new key.

21.5.1.2. 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 to 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
```

21.5.2. 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**.

21.5.2.1. 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.

The screenshot shows the IPA Server interface with the 'Automount' tab selected. The main title is 'AUTOMOUNT LOCATIONS'. Below it are buttons for Refresh, Delete, and Add. A list of locations is shown with checkboxes: 'default' and 'raleigh'.

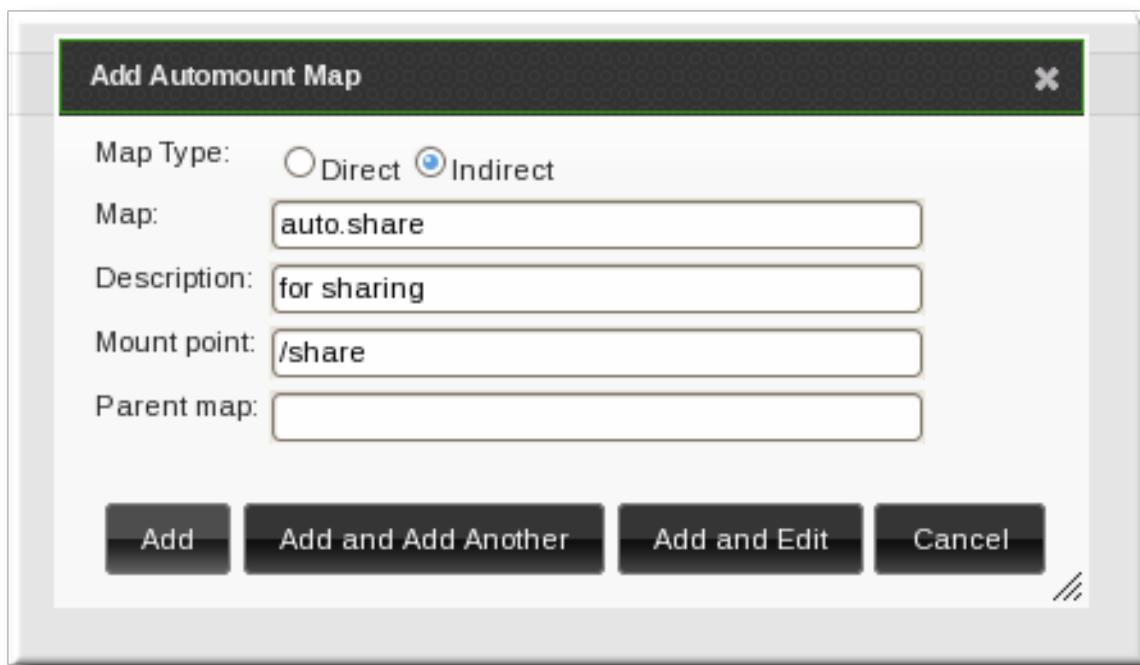
Location
<input type="checkbox"/> default
<input type="checkbox"/> raleigh

4. In the **Automount Maps** tab, click the **+ Add** link to create a new map.

The screenshot shows the IPA Server interface with the 'Automount' tab selected. The path 'Autounmount Locations > boston' is shown. The title is 'AUTOMOUNT LOCATION: boston'. Below it are tabs for 'Automount Maps' and 'Settings'. A list of maps is shown with checkboxes: 'Map', 'auto.direct', 'auto.man', 'auto.master', and 'auto.share'.

Map
<input type="checkbox"/> auto.direct
<input type="checkbox"/> auto.man
<input type="checkbox"/> auto.master
<input type="checkbox"/> auto.share

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.

21.5.2.2. Configuring Indirect Maps from the Command Line

The primary difference between a direct map and an indirect map is that there is no forward slash in front of an indirect key.

```
-----  
/etc/auto.share:  
man      ipa.example.com:/docs/man  
-----
```

1. Create an indirect map to set the base entry using the **automountmap-add-indirect** command. The **--mount** option sets the base directory to use for all the indirect map keys. The default parent entry is **auto.master**, but if another map exists which should be used, that can be specified using the **--parentmap** option.

```
$ ipa automountmap-add-indirect location mapName --mount=directory  
[- -parentmap=mapName]
```

For example:

```
$ ipa automountmap-add-indirect raleigh auto.share --mount=/share  
-----  
Added automount map "auto.share"  
-----
```

2. Add the indirect key for the mount location:

```
$ ipa automountkey-add raleigh auto.share --key=docs --info="ipa.example.com:/export/docs"
-----
Added automount key "docs"
-----
Key: docs
Mount information: ipa.example.com:/export/docs
```

3. To verify the configuration, check the location file list using **automountlocation-tofiles**:

```
$ ipa automountlocation-tofiles raleigh
/etc/auto.master:
/-      /etc/auto.direct
/share  /etc/auto.share
-----
/etc/auto.direct:
-----
/etc/auto.share:
man      ipa.example.com:/export/docs
```

On Solaris, add the indirect map using the **ldapclient** command to add the LDAP entry directly:

```
ldapclient -a
serviceSearchDescriptor=auto_share:automountMapName=auto.share,cn=location,cn=automount,dc=example,dc=com?one
```

21.5.3. 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 22. 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.

22.1. 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 22.1, “Password Policy Settings”](#).

Table 22.1. Password Policy Settings

Configuration Property Options for both the UI and CLI	Command-Line Option	Description
---	---------------------	-------------

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"> » Upper-case characters » Lower-case characters » Digits » Special characters (for example, punctuation) » 8-bit characters (characters whose decimal code starts at 128 or below) » Number of repeated characters <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.

Note

Even with the password history set to zero, users cannot reuse a *current* password.

Options for the CLI only

Priority	--priority	Sets the priority which determines which policy is in effect. The lower the number, the higher priority. Although this priority is required when the policy is first created in the UI, it cannot be reset in the UI. It can only be reset using the CLI.
Maximum Consecutive Failures	--maxfail	Specifies the maximum number of consecutive failures to input the correct password before the user's account is locked.
Fail Interval	--failinterval	Specifies the period (in seconds) after which the failure count will be reset.
Lockout Time	--lockouttime	Specifies the period (in seconds) for which a lockout is enforced.

22.2. 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 **pwpolicy-show** command. The CLI can also display the password policy in effect for a user.

22.2.1. 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 22.2, “Default Global Password Policy”](#).

Table 22.2. 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

22.2.1.1. 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.

The screenshot shows the IPA Server interface with a green header bar. The 'Policy' tab is selected. Below it, a navigation bar includes 'Host Based Access Control', 'Sudo', 'Automount', 'Password Policies' (which is highlighted with a blue border), and 'Kerberos Ticket Policy'. The main content area is titled 'PASSWORD POLICIES' and contains a sub-section for 'Group'. It lists four entries: 'engineering', 'global_policy', and 'ipausers', each preceded by a checkbox. A 'Delete' button is shown next to the first entry, and an 'Add' button is located below the list.

3. The global policy is displayed.

The screenshot shows the IPA Web UI interface. At the top, there are three tabs: 'Identity', 'Policy' (which is selected and highlighted in green), and 'IPA Server'. Below the tabs, a navigation bar includes links for 'Host Based Access Control', 'Sudo', 'Automount', 'Password Policies' (which is also highlighted in green), and 'Kerberos Ticket Policy'. A breadcrumb trail 'Password Policies » global_policy' is visible. The main title is 'PASSWORD POLICY: global_policy'. Below the title, there is a 'Settings' button and links for 'Refresh', 'Reset', and 'Update'. A section titled '▼ PASSWORD POLICY' contains various configuration fields:

- 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: (empty field)

22.2.1.2. 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

22.2.2. Viewing Group-Level Password Policies

22.2.2.1. 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.

The screenshot shows a web-based administrative interface for an IPA server. At the top, there is a navigation bar with tabs: 'Identity' (disabled), 'Policy' (selected and highlighted in green), and 'IPA Server'. Below the tabs is a sub-navigation bar with links: 'Host Based Access Control', 'Sudo', 'Automount', 'Password Policies' (which is the active subtab, indicated by a blue border), and 'Kerberos Ticket Policy'. The main content area has a title 'PASSWORD POLICIES' and two buttons: 'Delete' and '+ Add'. A list of groups is displayed, each with a checkbox and a link to its details. The groups listed are 'engineering', 'global_policy', and 'ipausers'.

3. The group policy is displayed.

The screenshot shows the IPA Management interface for defining password policies. The 'Policy' tab is active, and under it, the 'Password Policies' sub-tab is selected. The page title is 'PASSWORD POLICY: ipausers'. The 'Group' dropdown is set to 'engineering'. The configuration includes:

- Max lifetime (days): 90
- Min lifetime (hours): 48
- History size: 6
- Character classes: 3
- Min length: 8

22.2.2.2. 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
```

22.2.3. 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
```

22.3. 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.

22.3.1. Creating Password Policies in the Web UI

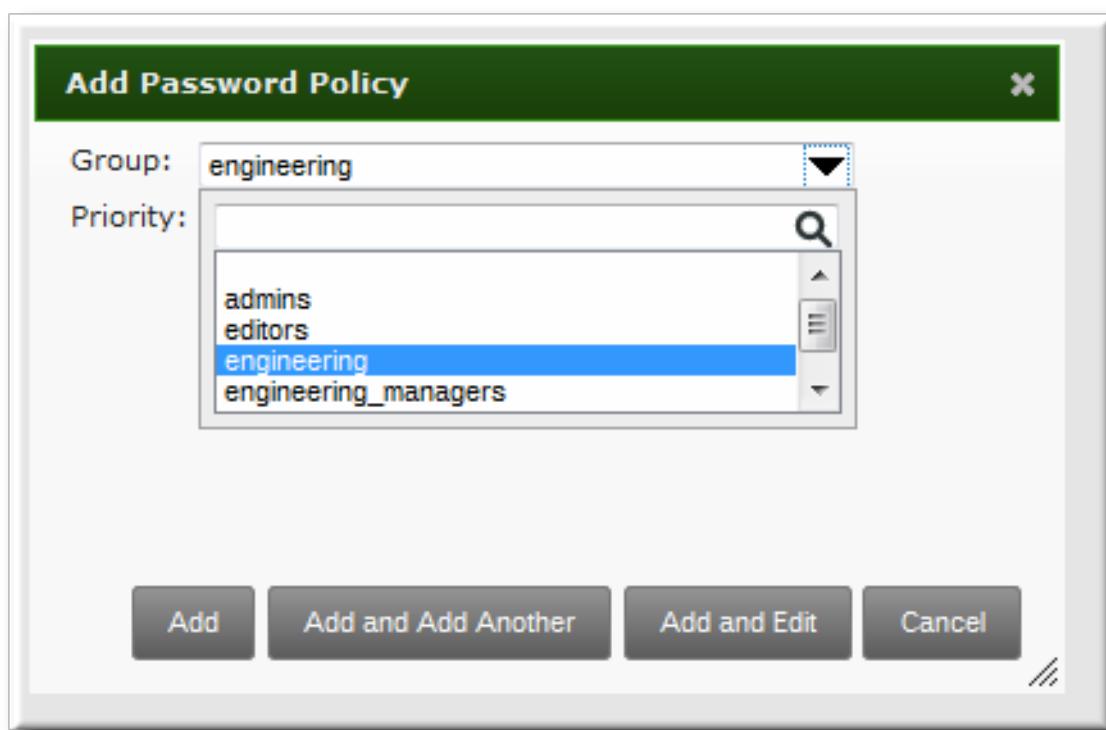
1. Click the **Policy** tab, and then click the **Password Policies** subtab.
2. All of the policies in the UI are listed by group. The global password policy is defined by the **global_policy** group. Click the group link.

The screenshot shows the IPA Server interface with the 'Policy' tab selected. Below it, the 'Password Policies' section displays a list of existing policies. The list includes:

- engineering
- global_policy
- ipausers
- Group

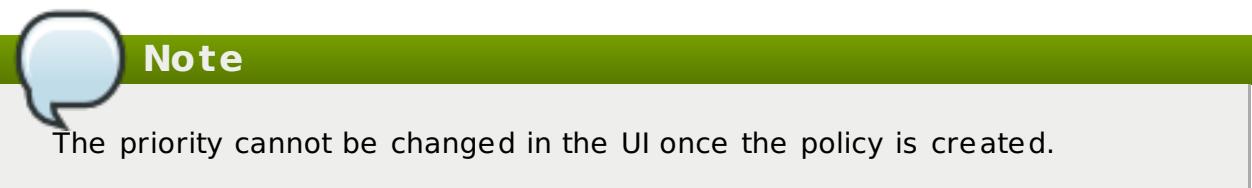
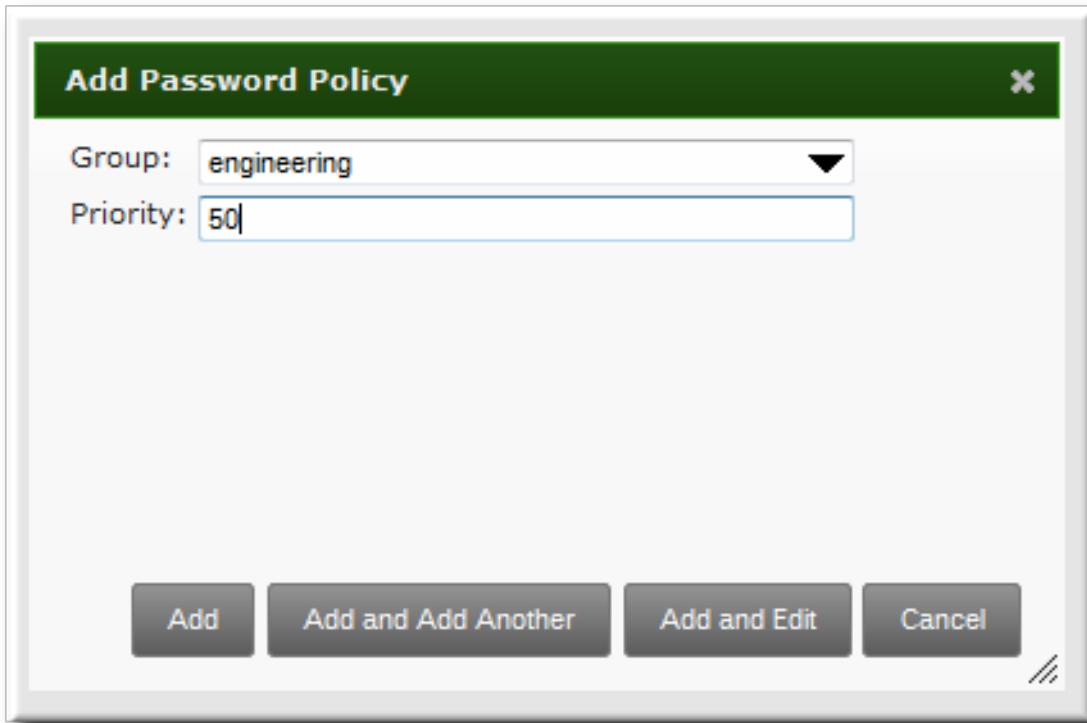
A delete link is located next to the 'engineering' entry.

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.



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 reuse 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 22.1, “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.

22.3.2. 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.

22.3.3. 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
```

22.4. 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 22.3, “Creating and Editing Password Policies”](#).

With password expiration periods, there are two attributes that are related:

- » The maximum lifetime setting given in the password policy (**--maxlife**)

- The actual date that the password for a given user expires (***krbPasswordExpiration***)

Changing the password expiration time in the password policy does not affect the expiration date for a user, until the user password is changed. If the password expiration date needs to be changed immediately, it can be changed by editing the user entry.

To force the expiration date to change, reset the ***krbPasswordExpiration*** attribute value for the user. **This can only be done using `ldapmodify`**. For example, for a single user:

```
[bjensen@ipaserver ~]$ ldapmodify -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 **`ldapmodify`** command.

Note

If an administrator resets a password, it makes the previous password expired 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.

22.5. 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.

22.6. 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 9.4, “Unlocking User Accounts After Password Failures”](#).

22.6.1. 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.

The screenshot shows a software interface for managing password policies. At the top, there are three tabs: **Identity**, **Policy**, and **IPA Server**. The **Policy** tab is active. Below the tabs, there are several sub-links: Host Based Access Control, Sudo, Automount, **Password Policies** (which is highlighted with a blue border), and Kerberos Ticket Policy. The main content area is titled **PASSWORD POLICIES**. It includes a delete button (x) and an add button (+ Add). A list of password policies is displayed, each with a checkbox and a name: Group, engineering, global_policy, and ipausers.

3. Set the account lockout attribute values.

▼ **PASSWORD POLICY**

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:	

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 the 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.

22.6.2. 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 the 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
```

22.7. 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 23. 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.

23.1. 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.

23.1.1. About Principal Names

The principal identifies not only the user or service, but also the realm that the 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**.

23.1.2. 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**.

23.2. 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.

23.2.1. Setting Global Ticket Policies

23.2.1.1. 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.

23.2.1.2. 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
```

23.2.2. 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.

23.3. 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.

- 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.

23.4. 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.

23.4.1. 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.

The screenshot shows the 'Services' section of the FreeIPA web interface. At the top, there are tabs for Identity, Policy, Authentication, Network Services, and IPA Server. Under Network Services, there are sub-tabs for Users, User Groups, Hosts, Host Groups, Netgroups, and Services, with 'Services' being the active tab. A search bar and a refresh/delete/add button are at the top right. Below is a table with three rows:

	Principal
<input type="checkbox"/>	DNS/ipa.demo1.freeipa.org@DEMO1.FREEIPA.ORG
<input type="checkbox"/>	HTTP/ipa.demo1.freeipa.org@DEMO1.FREEIPA.ORG

Figure 23.1. List of Services

2. Click on the service to which you want to add the flag.
3. Check the **Trusted for delegation** option.

The screenshot shows the 'Service Settings' page for the DNS service. At the top, it displays the service name: DNS/ipa.demo1.freeipa.org@DEMO1.FREEIPA.ORG. Below this, there are tabs for Settings, Roles, and Hosts (1). Action buttons include Refresh, Reset, Update, and Actions. The main area contains the following settings:

- Principal:** DNS/ipa.demo1.freeipa.org@DEMO1.FREEIPA.ORG
- Service:** DNS
- Host Name:** ipa.demo1.freeipa.org
- PAC type:**
 - Inherited from server configuration
 - Override inherited settings
 - MS-PAC
 - PAD
- Trusted for delegation:** (This checkbox is highlighted with a red box)
- Undo:** A button to undo changes.

Figure 23.2. Adding the OK_AS_DELEGATE Flag

23.4.2. 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]
```

23.5. 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/sssd/sssd.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/sssd/sssd.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](#).

23.6. 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 24. Using sudo

Identity Management provides a mechanism for predictably and consistently applying **sudo** policies across the IdM domain. Every system in the IdM domain can be configured as a **sudo** client.

24.1. The sudo Utility in Identity Management

The **sudo** utility gives administrative access to specified users. When trusted users precede an administrative command with **sudo**, they are prompted for their own password. Then, when they have been authenticated and assuming that the command is permitted, the administrative command is executed as if they were the root user. For more information about **sudo**, see the [System Administrator's Guide](#).

24.1.1. The Identity Management LDAP Schema for sudo

IdM has a specialized LDAP schema for **sudo** entries. The schema supports:

- » Host groups as well as netgroups. Note that **sudo** only supports netgroups.
- » **sudo** command groups, which contain multiple commands.

Note

Because **sudo** does not support host groups or command groups, IdM translates the IdM **sudo** configuration into the native **sudo** configuration when the **sudo** rules are created. For example, IdM creates a corresponding shadow netgroup for every host group, which allows the IdM administrator to create **sudo** rules that reference host groups, while the local **sudo** command uses the corresponding netgroup.

By default, the **sudo** information is not available anonymously over LDAP. Therefore, IdM defines a default **sudo** user at **uid=sudo,cn=sysaccounts,cn=etc,\$SUFFIX**. You can change this user in the LDAP **sudo** configuration file at **/etc/sudo-ldap.conf**.

24.1.2. NIS Domain Name Requirements

The NIS domain name must be set for netgroups and **sudo** to work properly. The **sudo** configuration requires NIS-formatted netgroups and a NIS domain name for netgroups. However, IdM does not require the NIS domain to actually exist. It is also not required to have a NIS server installed.

Note

The **ipa-client-install** utility sets a NIS domain name automatically to the IdM domain name by default.

24.2. sudo Rules in Identity Management

Using **sudo** rules, you can define *who* can do *what*, *where*, and *as whom*.

- » *Who* are the users allowed to use **sudo**.
- » *What* are the commands that can be used with **sudo**.
- » *Where* are the target hosts on which the users are allowed to use **sudo**.
- » *As whom* is the system or other user identity which the users assume to perform tasks.

24.2.1. External Users and Hosts in sudo Rules

IdM accepts external entities in **sudo** rules. External entities are entities that are stored outside of the IdM domain, such as users or hosts that are not part of the IdM domain.

For example, you can use **sudo** rules to grant root access to a member of the IT group in IdM, where the root user is not a user defined in the IdM domain. Or, for another example, administrators can block access to certain hosts that are on a network but are not part of the IdM domain.

24.2.2. User Group Support for sudo Rules

You can use **sudo** to give access to whole user groups in IdM. IdM supports both Unix and non-POSIX groups. Note that creating non-POSIX groups can cause access problems because any users in a non-POSIX group inherit non-POSIX permissions from the group.

24.2.3. Support for sudoers Options

IdM supports **sudoers** options. For a complete list of the available **sudoers** options, see the **sudoers(5)** man page.

Note that IdM does not allow white spaces or line breaks in **sudoers** options. Therefore, instead of supplying multiple options in a comma-separated list, add them separately. For example, to add two **sudoers** options from the command line:

```
$ ipa sudorule-add-option sudo_rule_name
Sudo Option: first_option
$ ipa sudorule-add-option sudo_rule_name
Sudo Option: second_option
```

Similarly, make sure to supply long options on one line. For example, from the command line:

```
$ ipa sudorule-add-option sudo_rule_name
Sudo Option: env_keep="COLORS DISPLAY EDITOR HOSTNAME HISTSIZE INPUTRC
KDEDIR LESSSECURE LS_COLORS MAIL PATH PS1 PS2 XAUTHORITY"
```

24.3. Configuring the Location for Looking up sudo Policies

The centralized IdM database for **sudo** configuration makes the **sudo** policies defined in IdM globally available to all domain hosts. On Red Hat Enterprise Linux 7.1 systems and later, the **ipa-server-install** and **ipa-client-install** utilities automatically configure the system to use the IdM-defined policies by setting SSSD as the data provider for **sudo**.

The location for looking up the **sudo** policies is defined on the **sudoers** line of the **/etc/nsswitch.conf** file. On IdM systems running Red Hat Enterprise Linux 7.1 and later, the default **sudoers** configuration in **nsswitch.conf** is:

```
sudoers: files sss
```

The **files** option specifies that the system uses the **sudo** configuration defined in the **/etc/sudoers** local SSSD configuration file. The **sss** option specifies that the **sudo** configuration defined in IdM is used.

24.3.1. Configuring Hosts to Use IdM sudo Policies in Earlier Versions of IdM

To implement the IdM-defined **sudo** policies on IdM systems running Red Hat Enterprise Linux versions earlier than 7.1, configure the local machines manually. You can do this using SSSD or LDAP. Red Hat strongly recommends to use the SSSD-based configuration.

24.3.1.1. Applying the sudo Policies to Hosts Using SSSD

Follow these steps on each system that is required to use SSSD for **sudo** rules:

1. 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.

2. Add **sudo** to the list of services managed by the local SSSD client.

```
# vim /etc/sssd/sssd.conf
[sssd]
config_file_version = 2
services = nss, pam, sudo
domains = IPADOMAIN
```

3. 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.

```
# systemctl enable rhel-domainname.service
```

- b. Set the NIS domain name to use with the **sudo** rules.

```
# nisdomainname example.com
```

- c. Configure the system authentication settings to persist the NIS domain name. For example:

```
# echo "NISDOMAIN=example.com.com" >> /etc/sysconfig/network
```

This updates the `/etc/sysconfig/network` and `/etc/yp.conf` files with the NIS domain.

4. Optionally, enable debugging in SSSD to show what LDAP settings it is using.

```
[domain/IPDOMAIN]
debug_level = 6
...
```

The LDAP search base used by SSSD for operations is recorded in the `sssd_DOMAINNAME.log` log.

24.3.1.2. 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 24.3.1.1, “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.

24.4. Adding `sudo` Commands, Command Groups, and Rules

24.4.1. Adding `sudo` Commands

Adding `sudo` Commands in the Web UI

1. Under the **Policy** tab, click **Sudo** → **Sudo Commands**.
2. Click **Add** at the top of the list.
3. Fill out the information about the command. Enter the full system path to the command executable.

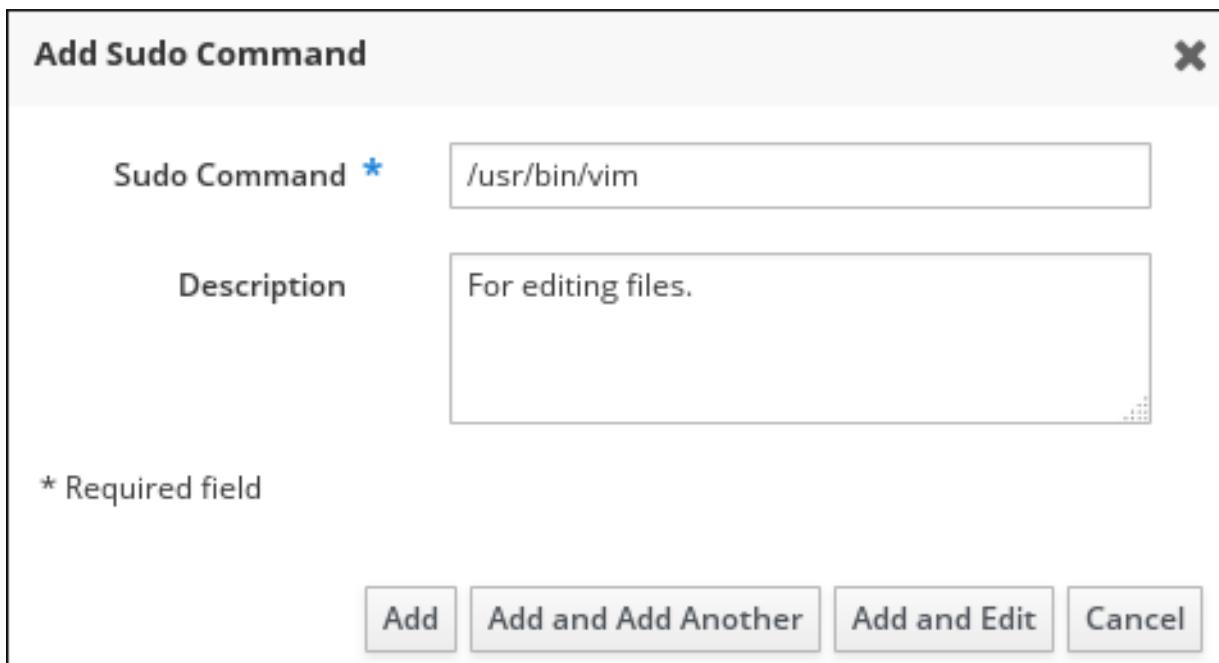


Figure 24.1. Adding a New sudo Command

- Click **Add**. Alternatively, click **Add and Add Another** to start adding another entry or **Add and Edit** to start editing the new entry.

Adding sudo Commands from the Command Line

To add a **sudo** command, use the **ipa sudocmd-add** command. Provide the full system path to the command executable. For example, to add the **/usr/bin/less** command and a description:

```
$ ipa sudocmd-add /usr/bin/less --desc="For reading log files"
-----
Added sudo command "/usr/bin/less"
-----
sudo Command: /usr/bin/less
Description: For reading log files
```

24.4.2. Adding sudo Command Groups

Adding sudo Command Groups in the Web UI

- Under the **Policy** tab, click **Sudo → Sudo Command Groups**.
- Click **Add** at the top of the list.
- Fill out the information about the command group.

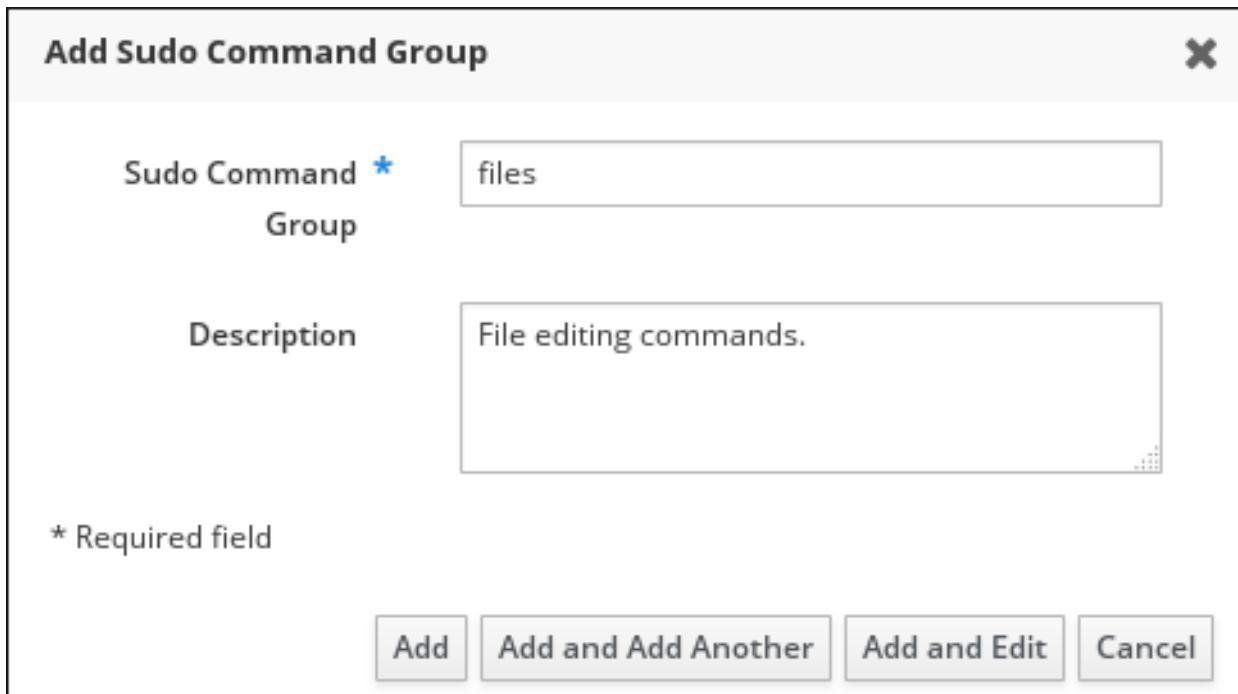


Figure 24.2. Adding a New sudo Command Group

4. Click **Add and Edit** to start editing the command group.
5. Under the **Sudo Commands** tab, click **Add** to add a **sudo** command to the group. Select the required commands and move them to the **Prospective** column using the **>** button.

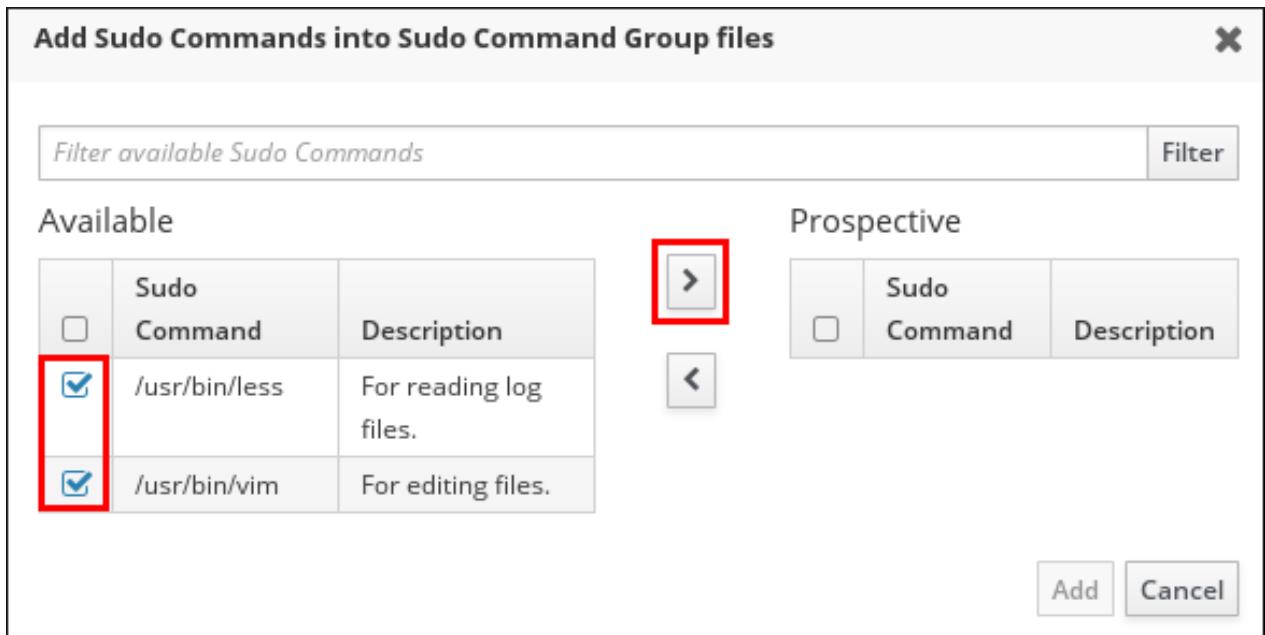


Figure 24.3. Adding Commands to a sudo Command Group

6. Click **Add**.

Adding sudo Command Groups from the Command Line

- Create the command group using the **ipa sudocmdgroup-add** command. For example, to create the **files** command group and add its description:

```
$ ipa sudocmdgroup-add files --desc="File editing commands"
-----
Added sudo command group "files"
-----
sudo Command Group: files
Description: File editing commands
```

- Include a **sudo** command in the group using the **ipa sudocmdgroup-add-member** command. Note that you can only include commands that have already been added to IdM, as described in [Section 24.4.1, “Adding sudo Commands”](#).

```
$ ipa sudocmdgroup-add-member files --sudocmds "/usr/bin/vim"
sudo Command Group: files
Description: File editing commands
Member sudo commands: /usr/bin/vim
-----
Number of members added 1
-----
```

24.4.3. Adding sudo Rules

Adding sudo Rules in the Web UI

- Under the **Policy** tab, click **Sudo** → **Sudo Rules**.
- Click **Add** at the top of the list.
- Enter the name for the rule.

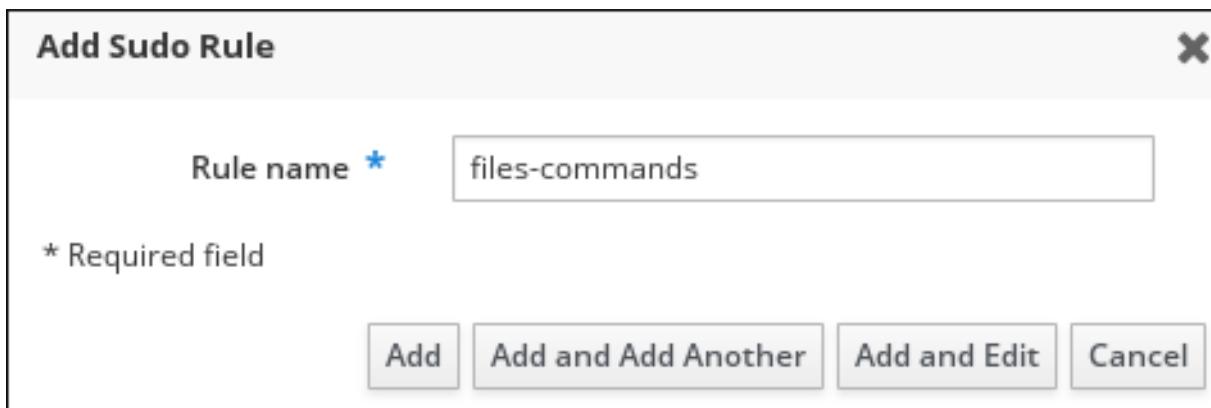


Figure 24.4. Naming a New sudo Rule

- Click **Add**. Alternatively, click **Add and Add Another** to start adding another entry or **Add and Edit** to start editing the new entry.

For information on how to edit the new **sudo** rule, see [Section 24.6, “Modifying sudo Rules”](#).

Adding sudo Rules from the Command Line

To add a new **sudo** rule, use the **ipa sudorule-add** command. For example, to add a rule named **files-commands**:

```
$ ipa sudorule-add files-commands
-----
Added Sudo Rule "files-commands"
-----
Rule name: files-commands
Enabled: TRUE
```

For more information on using **ipa sudorule-add** and the options it accepts, run the command with the **--help** option added.

For information on how to edit the new **sudo** rule, see [Section 24.6, “Modifying sudo Rules”](#).

For a complete example of adding a new **sudo** rule and editing it from the command line, see [Example 24.1, “Adding and Modifying a New sudo Rule from the Command Line”](#).

24.5. Modifying sudo Commands and Command Groups

Modifying sudo Commands and Command Groups in the Web UI

1. Under the **Policy** tab, click **Sudo → Sudo Commands** or **Sudo → Sudo Command Groups**.
2. Click the name of the command or command group to display its configuration page.
3. Change the settings as required. On some configuration pages, the **Save** button is available at the top of the page. On these pages, you must click the button to confirm the changes.

Modifying sudo Commands and Command Groups from the Command Line

To modify a command or command group, use the following commands:

- » **ipa sudocmd-mod**
- » **ipa sudocmdgroup-mod**

Add command-line options to the above-mentioned commands to update the **sudo** command or command group attributes. For example, to add a new description for the **/usr/bin/less** command:

```
$ ipa sudocmd-mod /usr/bin/less --desc="For reading log files"
-----
Modified Sudo Command "/usr/bin/less"
-----
Sudo Command: /usr/bin/less
Description: For reading log files
Sudo Command Groups: files
```

For more information about these commands and the options they accept, run them with the **--help** option added.

24.6. Modifying sudo Rules

Modifying sudo Rules in the Web UI

- Under the **Policy** tab, click **Sudo → Sudo Rules**.
- Click the name of the rule to display its configuration page.
- Change the settings as required. On some configuration pages, the **Save** button is available at the top of the page. On these pages, click the button to confirm the changes.

The **sudo** rule configuration page includes several configuration areas:

The General area

In this area, you can modify the rule's description and **sudo order**. The **sudo order** field accepts integers and defines the order in which IdM evaluates the rules. The rule with the highest **sudo order** value is evaluated first.

The Options area

In this area, you can add **sudoers** options to the rule.

- Click **Add** above the options list.

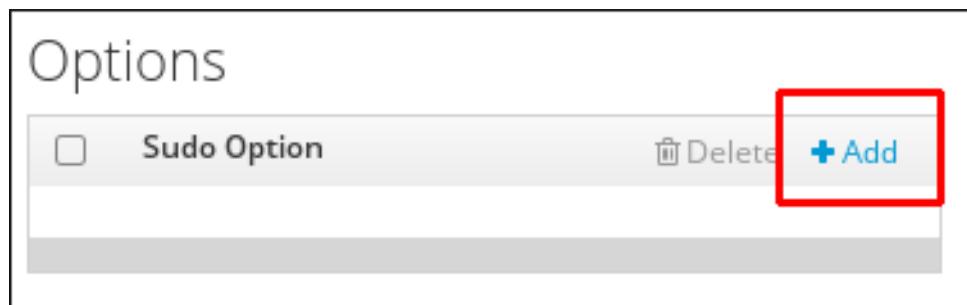


Figure 24.5. Adding a sudo Option

- Enter the **sudoers** option. For example, to specify that **sudo** will not prompt the user to authenticate, add the **!authenticate** option:



Figure 24.6. Entering a sudoers Option

For more information on **sudoers** options, see the **sudoers(5)** man page.

- Click **Add**.

The Who area

In this area, you can select the users or user groups to which the **sudo** rule will be applied. These users will be entitled to use **sudo** as defined in the rule.

To specify that all system users will be able to use **sudo** as defined in the rule, select **Anyone**.

To apply the rule to specific users or groups only, select **Specified Users and Groups** and then follow these steps:

1. Click **Add** above the users or user groups list.

The screenshot shows a 'Who' configuration page with two tables for adding users and groups. The top table is for 'Users' and the bottom table is for 'User Groups'. Each table has columns for a checkbox, the user/group name, and an 'External' field. A red box highlights the '+ Add' button in both tables.

Users		External	Delete	+ Add
<input type="checkbox"/>	manager			
<input type="checkbox"/>	employee			
<input type="checkbox"/>	helpdesk			

User Groups		External	Delete	+ Add
<input type="checkbox"/>	admins			

Figure 24.7. Adding Users to a sudo Rule

2. Select the users or user groups to add to the rule, and click the > arrow button to move them to the **Prospective** column. To add an external user, specify the user in the **External** field, and then click the > arrow button.

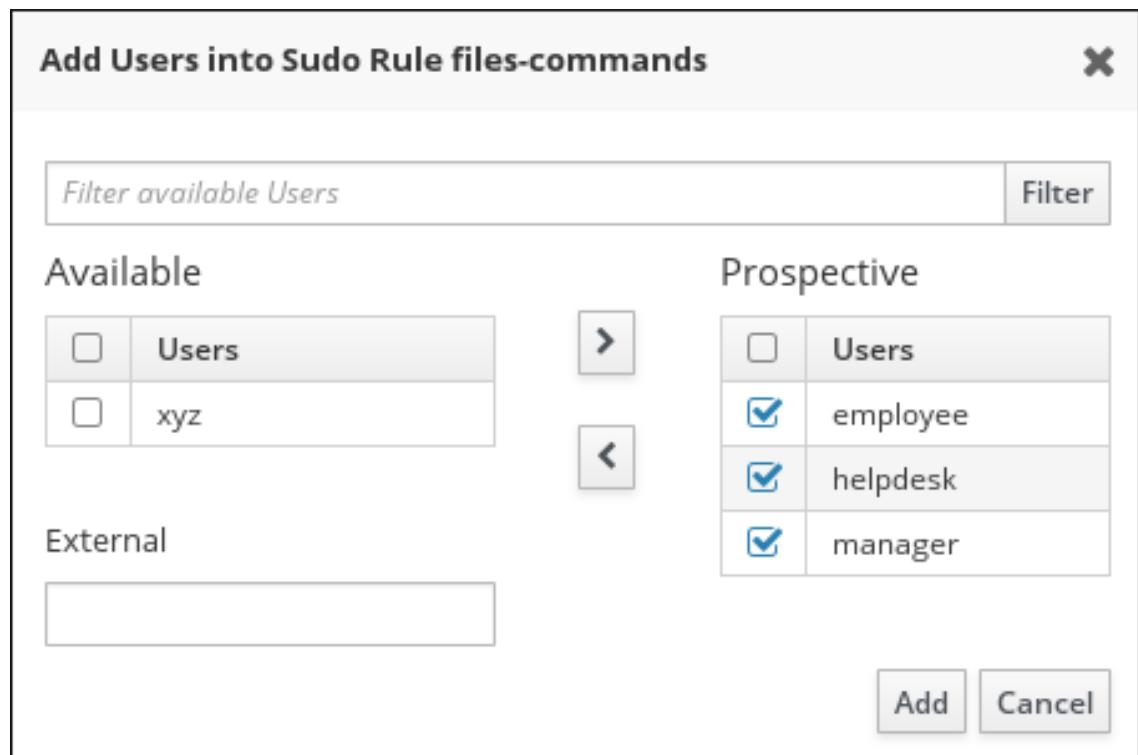


Figure 24.8. Selecting Users for a sudo Rule

- Click **Add**.

The Access This Host area

In this area, you can select the hosts on which the **sudo** rule will be in effect. These are the hosts where the users will be granted **sudo** permissions.

To specify that the rule will be in effect on all hosts, select **Anyone**.

To apply the rule to specific hosts or host groups only, select **Specified Hosts and Groups** and then follow these steps:

- Click **Add** above the hosts list.

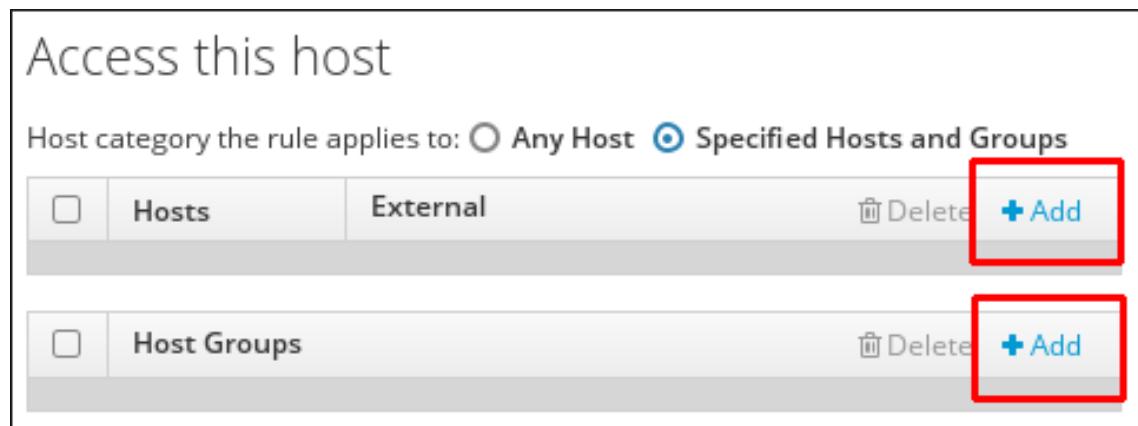


Figure 24.9. Adding Hosts to a sudo Rule

2. Select the hosts or host groups to include with the rule, and click the > arrow button to move them to the **Prospective** column. To add an external host, specify the user in the **External** field, and then click the > arrow button.

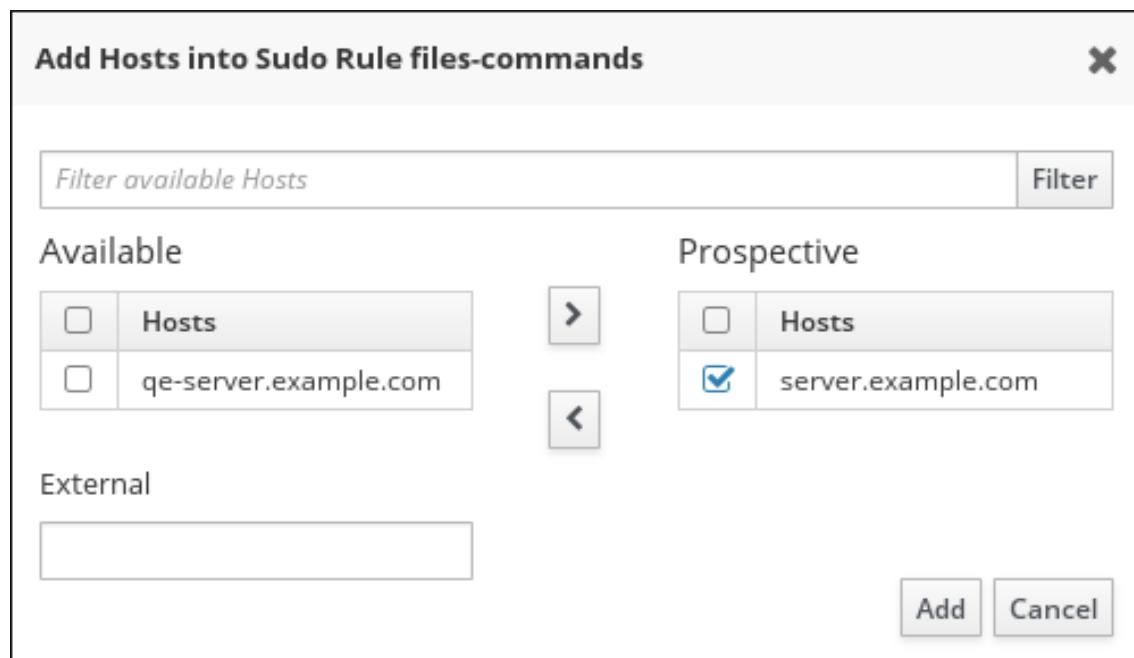


Figure 24.10. Selecting Hosts for a sudo Rule

- ### 3. Click Add.

The Run Commands area

In this area, you can select the commands to be included in the **sudo** rule. You can specify that users will be either allowed or denied to use specific commands.

To specify that users will be allowed to use any command with **sudo**, select **Any Command**.

To associate the rule with specific commands or command groups, select **Specified Commands and Groups** and then follow these steps:

1. Click one of the **Add** buttons to add a command or a command group.

To specify allowed commands or command groups, use the **Allow** area. To specify denied commands or command groups, use the **Deny** area.

Run Commands

Command category the rule applies to: Any Command Specified Commands and Groups

Allow

<input type="checkbox"/>	Sudo Allow Commands	<input type="button" value="Delete"/>	<input type="button" value="Add"/>
<input type="checkbox"/>	Sudo Allow Command Groups	<input type="button" value="Delete"/>	<input type="button" value="Add"/>

Deny

<input type="checkbox"/>	Sudo Deny Commands	<input type="button" value="Delete"/>	<input type="button" value="Add"/>
<input type="checkbox"/>	Sudo Deny Command Groups	<input type="button" value="Delete"/>	<input type="button" value="Add"/>

Figure 24.11. Adding Commands to a sudo Rule

2. Select the commands or command groups to include with the rule, and click the > arrow button to move them to the **Prospective** column.

Add Allow Sudo Commands into Sudo Rule files-commands X

Filter available Sudo Commands

Available		Prospective	
<input type="checkbox"/>	Sudo Commands	<input type="checkbox"/>	Sudo Commands
<input type="checkbox"/>	editing	<input checked="" type="checkbox"/>	files
<input type="checkbox"/>	log-files	<input type="button" value="<"/>	
<input type="checkbox"/>	login	<input type="button" value=">"/>	

Figure 24.12. Selecting Commands for a sudo Rule

3. Click **Add**.

The As Whom area

In this area, you can configure the **sudo** rule to run the given commands as a specific, non-root user.

Note that if you add a group of RunAs users, UIDs of the members of the group will be used to run the command. If you add a RunAs group, the GID of the group will be used to run the command.

To specify that the rule will be run as any user on the system, select **Anyone**. To specify that the rule will be run as any group on the system, select **Any Group**.

1. Click **Add** above the users list.

The screenshot shows the 'As Whom' configuration interface. It includes three main sections:

- RunAs User category the rule applies to:** Contains two rows:
 - RunAs Users (radio button selected)
 - Groups of RunAs UsersEach row has an 'Add' button (blue with a plus sign) and a 'Delete' button.
- RunAs Group category the rule applies to:** Contains one row:
 - RunAs Groups (radio button selected)It also has an 'Add' button and a 'Delete' button.
- A bottom section with a single 'Add' button (blue with a plus sign) highlighted by a red box.

Figure 24.13. Configuring sudo Rules to Execute Commands as a Specific User

2. Select the required users or groups, and use the > arrow button to move them to the **Prospective** column. To add an external entity, specify it in the **External** field, and then click the > arrow button.

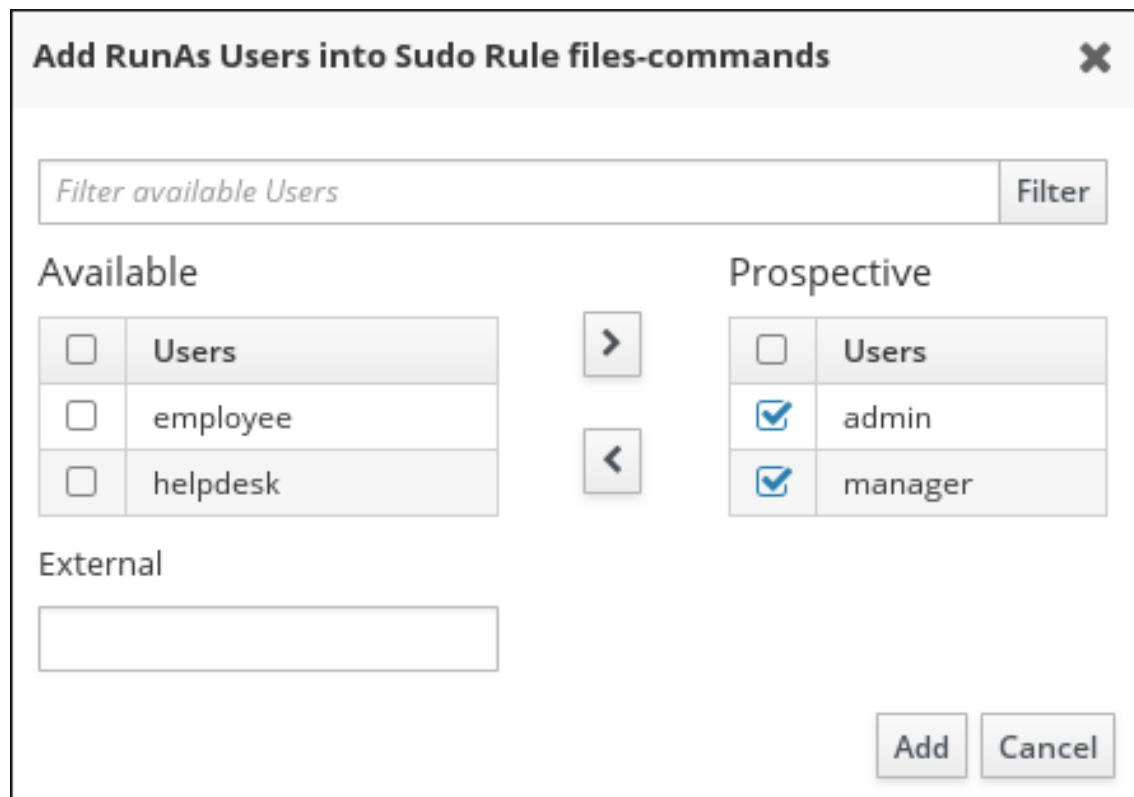


Figure 24.14. Selecting Users for the Command

3. Click **Add**.

Modifying sudo Rules from the Command Line

The IdM command-line utilities allow you to configure several **sudo** rule areas:

General sudo rules management

To change the general configuration for a **sudo** rule, use the **ipa sudorule-mod** command. The most common options accepted by the command are:

- » The **--desc** option to change the **sudo** rule description. For example:

```
$ ipa sudorule-mod sudo_rule_name --desc="sudo_rule_description"
```

- » The **--order** option to define the order of the specified rule. For example:

```
$ ipa sudorule-mod sudo_rule_name --order=3
```

- » Options to specify a category of entities: **--usercat** (user category), **--hostcat** (host category), **--cmdcat** (command category), **--runasusercat** (run-as user category), and **--runasgroupcat** (run-as group category). These options only accept the **all** value that associates the rule with all users, hosts, commands, run-as users, or run-as groups.

For example, to specify that all users will be able to use **sudo** as defined in the **sudo_rule** rule:

```
$ ipa sudorule-mod sudo_rule --usercat=all
```

Note that if the rule is already associated with a specific entity, you must remove it before defining the corresponding **all** category. For example, if **sudo_rule** was previously associated with a specific user using the **ipa sudorule-add-user** command, you must first use the **ipa sudorule-remove-user** command to remove the user.

For more details and a complete list of options accepted by **ipa sudorule-mod**, run the command with the **--help** option added.

Managing sudo options

To add a **sudoers** option, use the **ipa sudorule-add-option** command.

For example, to specify that users using **sudo** based on the **files-commands** rule will not be required to authenticate, add the **!authenticate** option:

```
$ ipa sudorule-add-option files-commands
Sudo Option: !authenticate
-----
Added option "!authenticate" to Sudo Rule "files-commands"
-----
```

For more information on **sudoers** options, see the **sudoers(5)** man page.

To remove a **sudoers** option, use the **ipa sudorule-remove-option** command. For example:

```
$ ipa sudorule-remove-option files-commands
Sudo Option: authenticate
-----
Removed option "authenticate" from Sudo Rule "files-commands"
-----
```

Managing who is granted the permission to use sudo

To specify an individual user, add the **--users** option to the **ipa sudorule-add-user** command. To specify a user group, add the **--groups** option to **ipa sudorule-add-user**.

For example, to add **user** and **user_group** to the **files-commands** rule:

```
$ ipa sudorule-add-user files-commands --users=user --
groups=user_group
...
-----
Number of members added 2
-----
```

To remove an individual user or group, use the **ipa sudorule-remove-user**. For example, to remove a user:

```
$ ipa sudorule-remove-user files-commands
[member user]: user
[member group]:
...
-----
```

```
-----  
Number of members removed 1  
-----
```

Managing where the users are granted the sudo permissions

To specify a host, add the **--hosts** option to the **ipa sudorule-add-host** command. To specify a host group, add the **--hostgroups** option to **ipa sudorule-add-host**.

For example, to add **example.com** and **host_group** to the **files-commands** rule:

```
$ ipa sudorule-add-host files-commands --hosts=example.com --  
hostgroups=host_group  
...  
-----  
Number of members added 2  
-----
```

To remove a host or host group, use the **ipa sudorule-remove-host** command. For example:

```
$ ipa sudorule-remove-host files-commands  
[member host]: example.com  
[member host group]:  
...  
-----  
Number of members removed 1  
-----
```

Managing what commands can be used with sudo

You can specify that users will be either allowed or denied to use specific commands.

To specify an allowed command or command group, add the **--sudocmds** or **--sudocmdgroups** option to the **ipa sudorule-add-allow-command**. To specify a denied command or command group, add the **--sudocmds** or **--sudocmdgroups** option to the **ipa sudorule-add-deny-command** command.

For example, to add the **/usr/bin/less** command and the **files** command group as allowed to the **files-commands** rule:

```
$ ipa sudorule-add-allow-command files-commands --  
sudocmds=/usr/bin/less --sudocmdgroups=files  
...  
-----  
Number of members added 2  
-----
```

To remove a command or command group from a rule, use the **ipa sudorule-remove-allow-command** or **ipa sudorule-remove-deny-command** commands. For example:

```
$ ipa sudorule-remove-allow-command files-commands  
[member sudo command]: /usr/bin/less
```

```
[member sudo command group]:
```

```
...
```

```
-----
```

Number of members removed 1

```
-----
```

Note that the **--sudocmds** option only accepts commands added to IdM, as described in [Section 24.4.1, “Adding sudo Commands”](#).

Managing as whom the sudo commands are run

To use the UIDs of an individual user or users in a group as the identity under which the commands are run, use the **--users** or **--groups** options with the **ipa sudorule-add-runasuser** command.

To use the GID of a user group as the identity for the commands, use the **ipa sudorule-add-runasgroup --groups** command.

If you specify no user or group, **sudo** commands will be run as root.

For example, to specify that the identity of **user** will be used to execute the commands in the **sudo** rule:

```
$ ipa sudorule-add-runasuser files-commands --users=user
...
RunAs Users: user
...
```

For more information on the **ipa sudorule-*** commands, see the output of the **ipa help sudorule** command or run a particular command with the **--help** option added.

Example 24.1. Adding and Modifying a New sudo Rule from the Command Line

To allow a specific user group to use **sudo** with any command on selected servers:

1. Obtain a Kerberos ticket for the **admin** user or any other user allowed to manage **sudo** rules.

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

2. Add a new **sudo** rule to IdM.

```
$ ipa sudorule-add new_sudo_rule --desc="Rule for user_group"
-----
Added Sudo Rule "new_sudo_rule"
-----
Rule name: new_sudo_rule
Description: Rule for user_group
Enabled: TRUE
```

3. Define the **who:** specify the group of users who will be entitled to use the **sudo** rule.

```
$ ipa sudorule-add-user new_sudo_rule --groups=user_group
```

```

Rule name: new_sudo_rule
Description: Rule for user_group
Enabled: TRUE
User Groups: user_group
-----
Number of members added 1
-----
```

- Define the *where*: specify the group of hosts where the users will be granted the **sudo** permissions.

```

$ ipa sudorule-add-host new_sudo_rule --hostgroups=host_group
Rule name: new_sudo_rule
Description: Rule for user_group
Enabled: TRUE
User Groups: user_group
Host Groups: host_group
-----
Number of members added 1
-----
```

- Define the *what*: to allow the users to run any **sudo** command, add the **all** command category to the rule.

```

$ ipa sudorule-mod new_sudo_rule --cmdcat=all
-----
Modified Sudo Rule "new_sudo_rule"
-----
Rule name: new_sudo_rule
Description: Rule for user_group
Enabled: TRUE
Command category: all
User Groups: user_group
Host Groups: host_group
```

- To let the **sudo** commands be executed as root, do not specify any run-as users or groups.
- Add the **!authenticate sudoers** option to specify that the users will not be required to authenticate when using the **sudo** command.

```

$ ipa sudorule-add-option new_sudo_rule
Sudo Option: !authenticate
-----
Added option "!authenticate" to Sudo Rule "new_sudo_rule"
-----
Rule name: new_sudo_rule
Description: Rule for user_group
Enabled: TRUE
Command category: all
User Groups: user_group
Host Groups: host_group
Sudo Option: !authenticate
```

- Display the new **sudo** rule configuration to verify it is correct.

```
$ ipa sudorule-show new_sudo_rule
  Rule name: new_sudo_rule
  Description: Rule for user_group
  Enabled: TRUE
  Command category: all
  User Groups: user_group
  Host Groups: host_group
  Sudo Option: !authenticate
```

24.7. Listing and Displaying sudo Commands, Command Groups, and Rules

Listing and Displaying sudo Commands, Command Groups, and Rules in the Web UI

1. Under the **Policy** tab, click **Sudo** and select **Sudo Rules**, **Sudo Commands**, or **Sudo Command Groups**.
2. Click the name of the rule, command, or command group to display its configuration page.

Listing and Displaying sudo Commands, Command Groups, and Rules from the Command Line

To list all commands, command groups, and rules, use the following commands:

- » **ipa sudocmd-find**
- » **ipa sudocmdgroup-find**
- » **ipa sudorule-find**

To display information about a particular command, command group, or rule, use the following commands:

- » **ipa sudocmd-show**
- » **ipa sudocmdgroup-show**
- » **ipa sudorule-show**

For example, to display information about the **/usr/bin/less** command:

```
$ ipa sudocmd-show /usr/bin/less
  Sudo Command: /usr/bin/less
  Description: For reading log files.
  Sudo Command Groups: files
```

For more information about these commands and the options they accept, run them with the **--help** option added.

24.8. Disabling and Enabling sudo Rules

Disabling a **sudo** rule temporarily deactivates it. A disabled rule is not removed from IdM and can be enabled again.

Disabling and Enabling sudo Rules from the Web UI

1. Under the **Policy** tab, click **Sudo** → **Sudo Rule**.
2. Select the rule to disable and click **Disable** or **Enable**.

<input type="checkbox"/>	Rule name	Status	Description
<input checked="" type="checkbox"/>	files-commands	✓ Enabled	

Showing 1 to 1 of 1 entries.

Figure 24.15. Disabling or Enabling a sudo Rule

Disabling and Enabling sudo Rules from the Command Line

To disable a rule, use the **ipa sudorule-disable** command.

```
$ ipa sudorule-disable sudo_rule_name
-----
Disabled Sudo Rule "sudo_rule_name"
```

To re-enable a rule, use the **ipa sudorule-enable** command.

```
$ ipa sudorule-enable sudo_rule_name
-----
Enabled Sudo Rule "sudo_rule_name"
```

24.9. Removing sudo Commands, Command Groups, and Rules

Removing sudo Commands, Command Groups, and Rules in the Web UI

1. Under the **Policy** tab, click **Sudo** and select **Sudo Rules**, **Sudo Commands**, or **Sudo Command Groups**.
2. Select the command, command group, or rule to delete, and click **Delete**.

Sudo Commands		
Search		
	Sudo Command	Description
<input checked="" type="checkbox"/>	/usr/bin/less	For reading log files.
<input type="checkbox"/>	/usr/bin/vim	For editing files.

Showing 1 to 2 of 2 entries.

Figure 24.16. Deleting a sudo Command

Removing sudo Commands, Command Groups, and Rules from the Command Line

To delete a command, command group, or rule, use the following commands:

- » **ipa sudocmd-del**
- » **ipa sudocmdgroup-del**
- » **ipa sudorule-del**

For more information about these commands and the options they accept, run them with the **--help** option added.

Chapter 25. 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.

25.1. About Host-Based Access Control

Host-based access control rules 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.

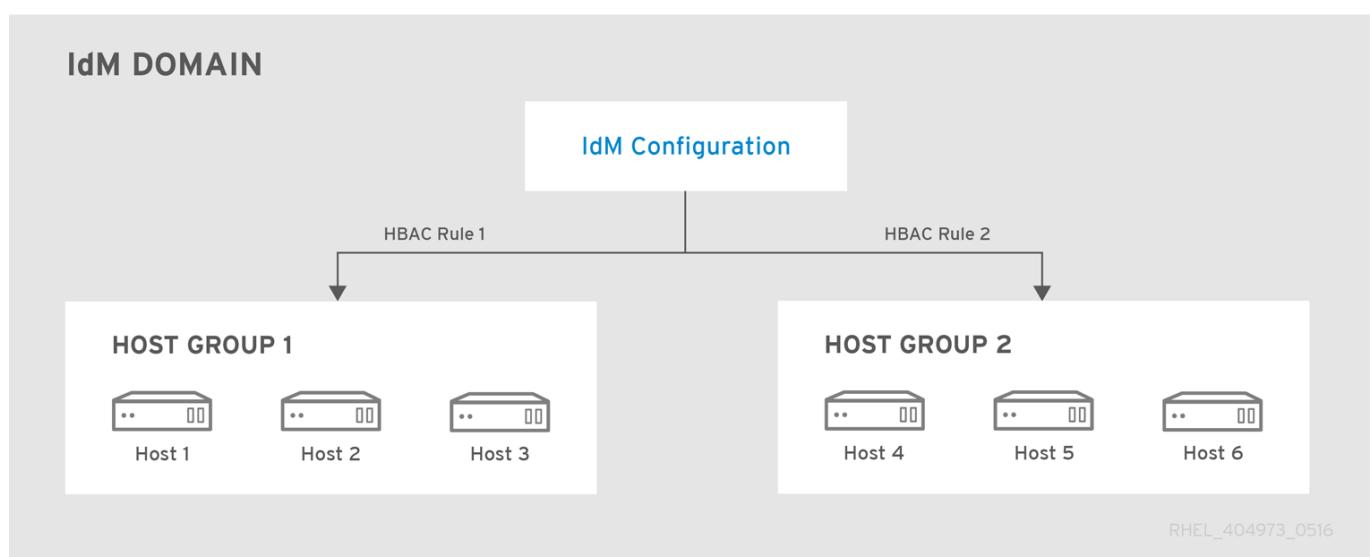


Figure 25.1. Host Groups and Host-Based Access Control



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.

25.2. Creating Host-Based Access Control Entries for Services and Service Groups

Any PAM service can be added 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.

25.2.1. Adding HBAC Services

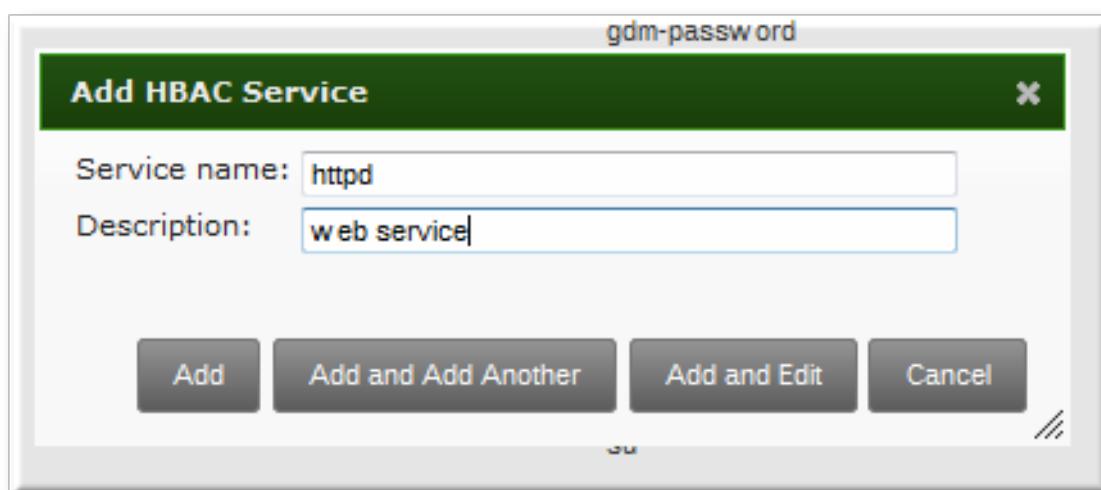
25.2.1.1. Adding HBAC Services in the Web UI

1. Click the **Policy** tab.
2. Click the **Host-Based Access Control** subtab, and then select the **HBAC Services** link.

- Click the **Add** link at the top of the list of services.

	Service name	Description
<input type="checkbox"/>	ftp	ftp
<input type="checkbox"/>	gdm	gdm
<input type="checkbox"/>	gdm-password	gdm-password
<input type="checkbox"/>	gssftp	gssftp
<input type="checkbox"/>	kdm	kdm

- Enter the service name and a description.



- Click the **Add** button to save the new service.

- If a service group already exists, then add the service to the desired group, as described in [Section 25.2.2.1, “Adding Service Groups in the Web UI”](#).

25.2.1.2. Adding Services in the Command Line

The service is added to the access control system using the **hbacsrv-add** command, specifying the service by the name that PAM uses to evaluate the service.

For example, this adds the **tftp** service:

```
# ipa hbacsrv-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 **hbacsvcgroup-add-member** command, as in [Section 25.2.2.2, “Adding Service Groups in the Command Line”](#).

25.2.2. Adding Service Groups

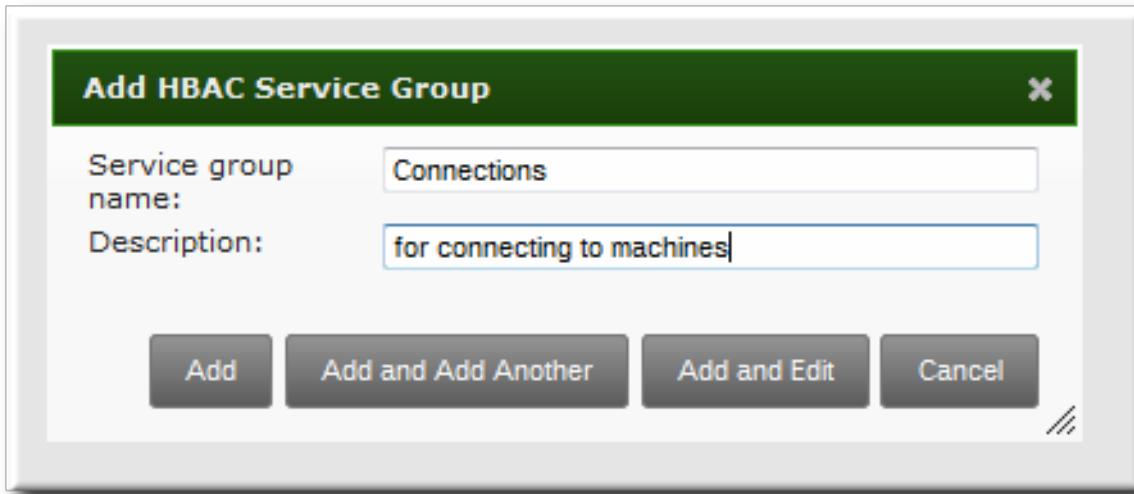
Once the individual service is added, it can be added to the access control rule. However, if there is a large number of services, then it can require frequent updates to the access control rules as services change. Identity Management also allows groups of services to be added to access control rules. 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.

25.2.2.1. 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.

	Service group name	Description
<input type="checkbox"/>	ftp	Default group of ftp related services
<input type="checkbox"/>	Sudo	Default group of Sudo related services

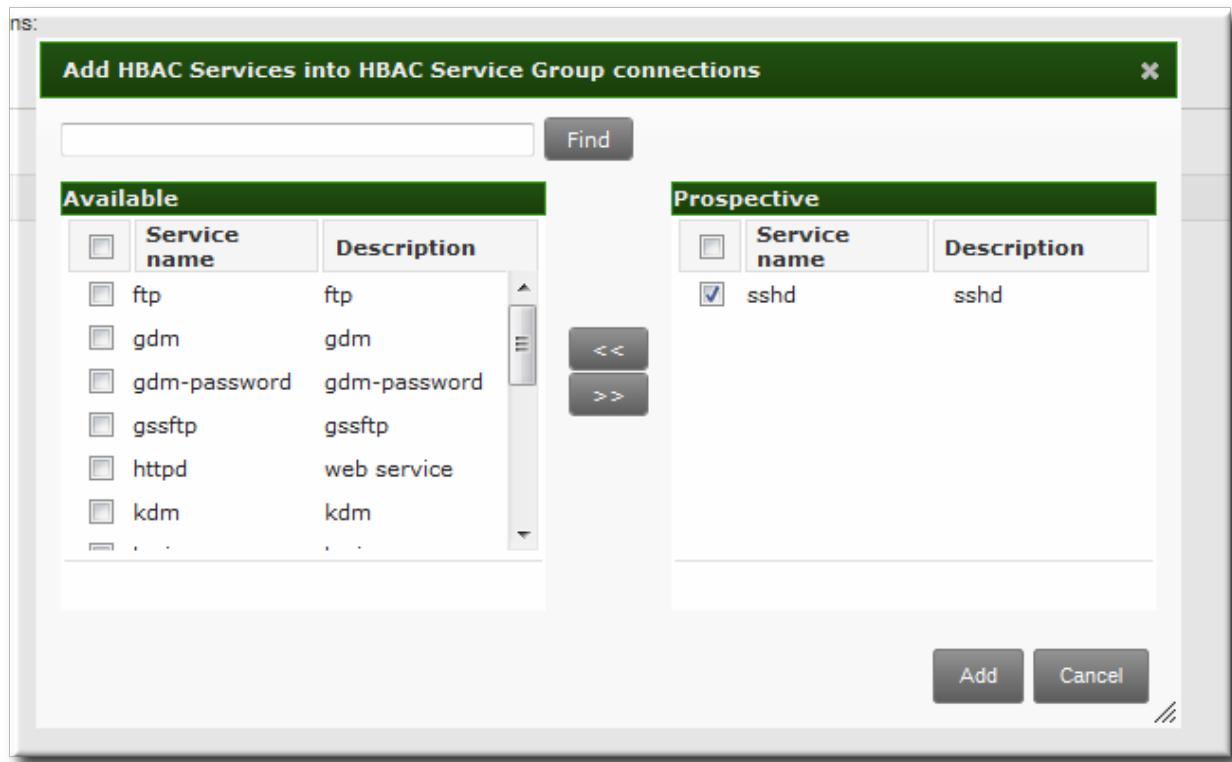
4. Enter the service group name and a description.



5. Click the **Add and Edit** button to go immediately to the service group configuration page.
6. At the top of the **HBAC Services** tab, click the **Add** link.

<input type="checkbox"/>	Service name	Description
<input type="checkbox"/>	httpd	web service
<input type="checkbox"/>	login	login

7. Click the checkbox by the names of the services to add, and click the right arrows button, **>>**, to move the command to the selection box.



- Click the **Add** button to save the group membership.

25.2.2.2. 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 hbacsvcgroup-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 hbacsvcgroup-add-member --hbacservcs=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.

25.3. 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.

25.3.1. 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.

Rule name	Status	Description
allow_all	✓ Enabled	Allow all users to access any host from any host
rule01	✓ Enabled	Test HBAC Rule 01
rule02	✓ Enabled	Test HBAC Rule 02

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 three basic elements are *who* the rule applies to, what hosts allow access (the target), and, optionally,

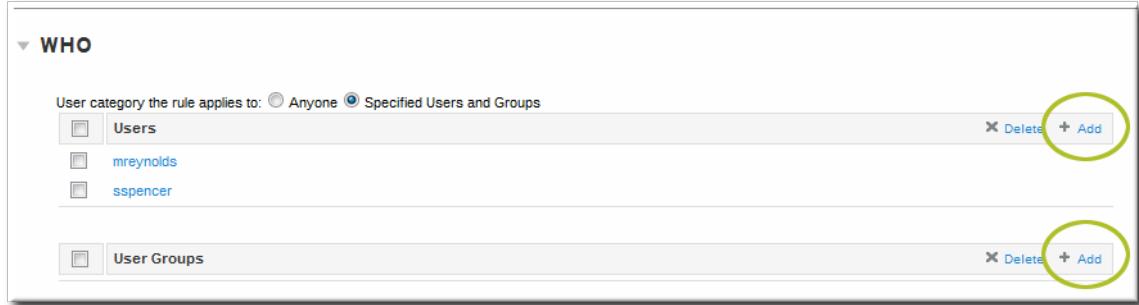
what services can be accessed.

- In the **Who** area, select the users or user groups to which the access control rule is applied.

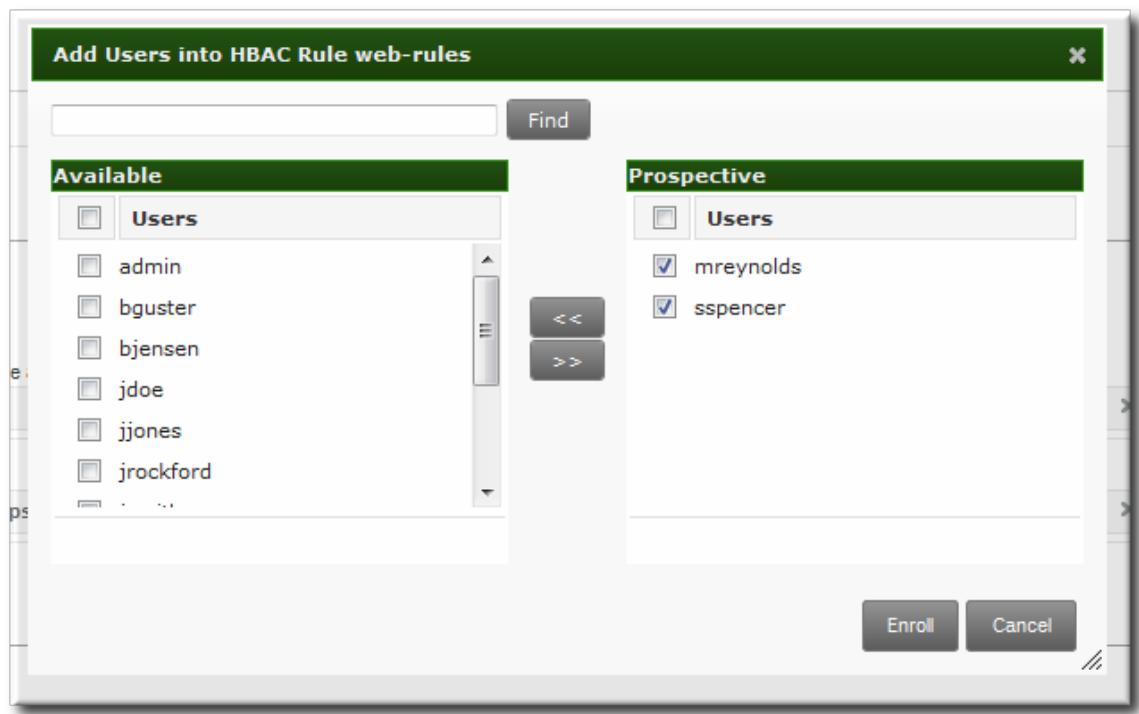
To apply the rule to all IdM users, select the **Anyone** radio button.

To apply the rule to a specific set of users or user groups:

- Select the **Specified Users and Groups** radio button.
- Click the **+ Add** link at the right of the users list.



- Click the checkbox by the users to add to the rule, and click the right arrows button, **>>**, to move the users to the selection box.



- Click **Add**.

- In the **Accessing** area, select the target hosts which can be accessed through this access control rule.

To apply the rule to all IdM hosts, select the **Any Host** radio button.

To apply the rule to a specific set of hosts or host groups:

- Select the **Specified Hosts and Groups** radio button.
- Click the **+ Add** link at the right of the hosts list.

ACCESSING

Host category the rule applies to: Any Host Specified Hosts and Groups

		X Delete	+ Add
<input type="checkbox"/>	Hosts		
<input type="checkbox"/>	server.example.com		

		X Delete	+ Add
<input type="checkbox"/>	Host Groups		

- 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.

Add Hosts into HBAC Rule web-rules

Find

Available		Prospective	
<input type="checkbox"/>	Hosts	<input checked="" type="checkbox"/>	Hosts
<input type="checkbox"/>	qe-server.example.com	<input checked="" type="checkbox"/>	server.example.com

<< >>

Add Cancel

- d. Click **Add**.
8. In the **Via Service** area, select specific services on the target hosts which the users are allowed to use to access target machines.

To apply the rule to all IdM hosts, select the **Any Service** radio button.

To apply the rule to a specific set of hosts or host groups:

- Select the **Specified Services and Groups** radio button.
- Click the **+ Add** link at the right of the commands list.

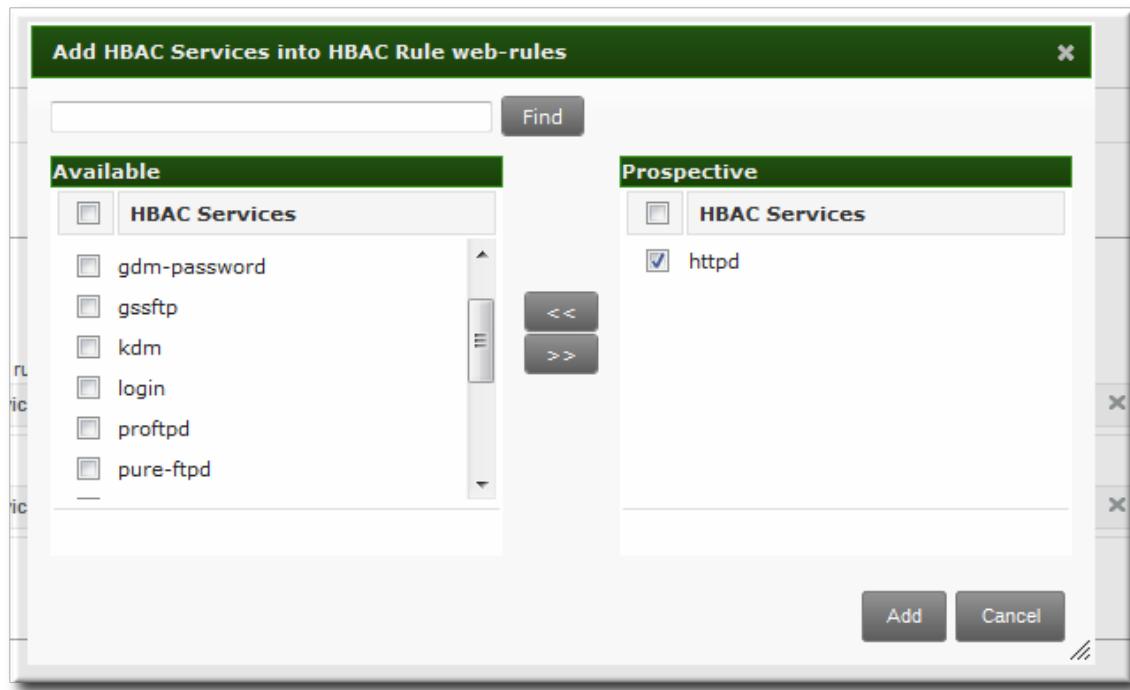
VIA SERVICE

Service category the rule applies to: Any Service Specified Services and Groups

		X Delete	+ Add
<input type="checkbox"/>	HBAC Services		
<input type="checkbox"/>	httpd		
<input type="checkbox"/>	ftp		
<input type="checkbox"/>	sshd		

		X Delete	+ Add
<input type="checkbox"/>	HBAC Service Groups		

- 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**.

25.3.2. Setting Host-Based Access Control Rules in the Command Line

Access control rules are created using the **hbacruler-*** commands (listed in [Table 25.1, "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 hbacruler-add* options ruleName
```

Note

To set every user or every host as a target, use the category options, such as **--usercat=all**.

Example 25.1. 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 hbacruler-add --usercat=all allGroup
```

```
-----  
Added HBAC rule "allGroup"  
-----
```

```
Rule name: allGroup
User category: all
Enabled: TRUE
```

The second command adds the target host to the HBAC rule:

```
$ 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 25.2. 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 **hbacruler-add-user** command:

```
$ ipa hbacruler-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 **hbacsrv-add** command.) This is the service that the user can use to connect to the machine.

```
$ ipa hbacruler-add-service --hbacsrvs=sshd sshd-jsmith
```

Example 25.3. 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 **hbacruler-add-service** command, only with the **--hbacsrvgroups** option that specifies the group name.

```
$ ipa hbacruler-add-service --hbacsrvgroups=login loginRule
```

Table 25.1. 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"> ✖ <code>--usercat=all</code>, which applies the rule to every user ✖ <code>--hostcat=all</code>, which sets every host as an allowed target server ✖ <code>--servicecat=all</code>, which sets every configured service as an allowed target service ✖ <code>ruleName</code>, which is the required unique identifier for the new rule 	
hbacrule-add-host	Adds a target host to the access control rule. A target host can be accessed by other servers and users in the domain.	<ul style="list-style-type: none"> ✖ <code>--hosts</code>, which adds an individual server or comma-separated list of servers as an allowed target server ✖ <code>--hostgroups</code>, which adds a host group to the rule and every host within the host group is an allowed target server ✖ <code>ruleName</code>, which is the rule to which to add the target server 	Target

Command	Description	Arguments	Source or Target Entry
hbacrule-add-service	Adds a service type to the rule.	<ul style="list-style-type: none"> ✿ --hbacsrvcs, which adds an individual service type or a list of service types as an allowed target service 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>. ✿ --hbacsvcgroups, which adds a service group to the rule and every service within the service group is an allowed target service 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>. ✿ <i>ruleName</i>, which is the rule to which to add the target service 	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"> ✖ --users, which adds an individual user or comma-separated list of users to the rule ✖ --groups, which adds a user group to the rule and, thus, every user within the group ✖ <i>ruleName</i>, which is the rule to which to add the user 	Source
hbacrule-disable hbacrule-enable	Disables or enables a host-based access control rule. Rules can be disabled if their behavior needs to be evaluated (for troubleshooting or to test a new rule).	<i>ruleName</i> , which is the rule to disable or enable	

25.4. 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.

25.4.1. 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.

25.4.2. 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 **--nodelay**, 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 25.4. 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 25.5. 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 25.6. 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.

25.4.3. Testing Host-Based Access Control Rules in the UI

As [Section 25.4.1, “The Limits of Host-Based Access Control Configuration”](#) details, misconfiguring a host-based access-control rule can result in unpredictable behavior when users or services attempt to connect to a remote host.

Testing host-based access control can help confirm that the rule performs as expected before it is deployed or to troubleshoot a rule once it is already active.

Note

The **hbactest** command does not work with trusted Active Directory users. Active Directory user/group associations are determined dynamically, as a user logs in, and those data are not stored in the IdM LDAP directory. The **hbactest** command, then, is unable to resolve the group memberships to check how access control rules will be applied.

By the nature of host-based access control rules, a test must define and verify a very specific set of criteria. A test run defines:

- » The user to run the operation as to test the rule performance for that user (**Who**).
- » To target host Z (**Accessing**).
- » Using the login client Y (**Via Service**).
- » The rule to test; if this is not used, then all enabled rules are tested (**Rules**).

The test environment is defined on the **HBAC TEST** page in the **Host Based Access Control** tab under **Policy**. A series of tabs is set up for each configuration step.

The screenshot shows the 'HBAC TEST' configuration interface. At the top, there are three tabs: 'Identity', 'Policy' (which is selected), and 'IPA Server'. Below these are several sub-tabs: 'Host Based Access Control' (selected), 'Sudo', 'Automount', 'Password Policies', 'Kerberos Ticket Policy', 'SELinux User Maps', and 'Automember'. Under the 'Host Based Access Control' tab, there are four sub-tabs: 'HBAC RULES', 'HBAC SERVICES', 'HBAC SERVICE GROUPS', and 'HBAC TEST' (selected). The 'FROM' tab is currently active, indicated by a green border. It contains a row of buttons: 'Who', 'Accessing', 'Via Service', 'From', 'Rules', and 'Run Test'. Below this is a search bar labeled 'FROM' with a magnifying glass icon. The search results table has columns for 'Host name', 'Description', and 'Enrolled?'. It shows two entries: 'dev.example.com' (selected) and 'test.example.com'. At the bottom of the table are buttons for 'Prev', 'Next', and 'Page: 1 / 1'. There is also a link 'Specify external Host:' and a note 'Showing 1 to 2 of 2 entries.'

Figure 25.2. 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 whether access was granted or denied to the users and also display the rules which matched the given parameters.

The screenshot shows a web-based interface for managing Host-Based Access Control (HBAC) rules. At the top, there are tabs for Identity, Policy, and IPA Server, with Policy selected. Below the tabs, a navigation bar includes Host Based Access Control, Sudo, Automount, Password Policies, Kerberos Ticket Policy, SELinux User Maps, and Automember. Under the Policy tab, sub-links for HBAC RULES, HBAC SERVICES, HBAC SERVICE GROUPS, and HBAC TEST are present, with HBAC TEST selected. A large "RUN TEST" button is visible. Below it is a horizontal form with fields: Who, Accessing, Via Service, From, Rules, and Run Test. The "Run Test" button is highlighted. To its right, the message "ACCESS GRANTED" is displayed. A table titled "RULES" lists 21 entries, with checkboxes for Matched and Unmatched. The table columns are Rule name, Matched, Status, and Description. The "Matched" column contains mostly checked boxes, while the "Status" column shows various states like Enabled, Disabled, and Pending. The "Description" column provides a brief summary for each rule. At the bottom of the table, it says "Showing 1 to 20 of 21 entries." and "Prev Next Page: 1 / 2". Below the table are "Prev" and "New Test" buttons.

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	

Figure 25.3. 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 26. 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.

26.1. 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) and a multi-category security context (MCS). The MLS/MCS contexts *confine* users to what processes, files, and operations they can access on the system.

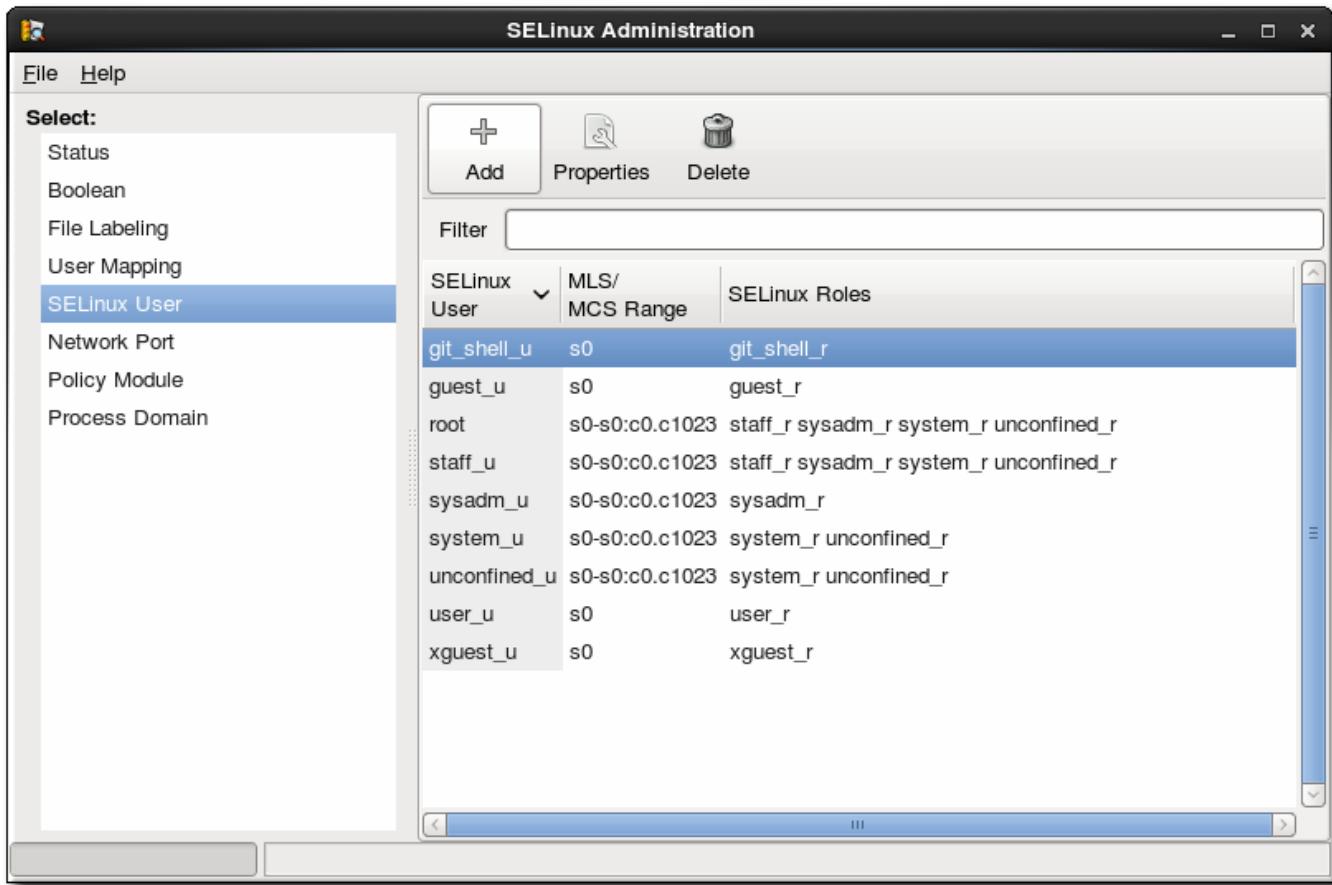


Figure 26.1. SELinux Users in the SELinux Manager

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, 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 25, 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.

26.2. 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 26.3, “Mapping SELinux Users and IdM Users”](#).

26.2.1. 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 interface. At the top, there are tabs for Identity, Policy, and IPA Server, with IPA Server selected. Below the tabs, there are sub-tabs: Role Based Access Control, Self Service Permissions, Delegations, and Configuration, with Configuration selected. The main area is titled "CONFIGURATION". It contains three expandable sections: "SEARCH OPTIONS", "USER OPTIONS", and "GROUP OPTIONS". Under "SEARCH OPTIONS", there is a section for "SELINUX OPTIONS" which is expanded. It shows two fields: "SELinux user map order:" containing "guest_u:s0\$guest_u:s0\$user_u:s0-s0:c0.c1023" and "Default SELinux user:" containing "guest_u:s0". There are also "Refresh" and "Reset" buttons at the top of this section.

- Click the **Update** link at the top of the page to save the changes.

26.2.2. 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 26.1. List of SELinux Users

The complete list of SELinux users is passed in the **--ipaselinuxusermaporder** 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 --  
ipaselinuxusermaporder="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 26.2. 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 **--ipaselinuxusermapdefault**. For example:

```
[jsmith@server ~]$ ipa config-mod --  
ipaselinuxusermapdefault="guest_u:s0"
```

26.3. 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.

26.3.1. 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.

The screenshot shows the 'SELinux User Maps' page. At the top, there are tabs for Identity, Policy, and IPA Server. Below them is a navigation bar with links for Host Based Access Control, Sudo, Automount, Password Policies, Kerberos Ticket Policy, SELinux User Maps (which is selected), and Automember. The main area is titled 'SELINUX USER MAPS' and contains a table with two rows:

Rule name	SELinux User	Status	Description
system_unconfined	unconfined_u:s0-s0:c0.c1023	✓ Enabled	description
test01	xguest_u:s0	— Disabled	Test SELinux User Map 01

- 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]*.

The dialog box is titled 'Add SELinux User Map'. It contains two fields: 'Rule name:' with value 'example-map' and 'SELinux User:' with value 'sysadmin_u:s0'. Both fields have a red asterisk indicating they are required. Below the fields is a note: '* Required field'. At the bottom are four buttons: 'Add', 'Add and Add Another', 'Add and Edit' (which is highlighted in grey), and 'Cancel'.

- Click **Add and Edit** to add the IdM user information.
- 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-based configuration interface for a SELinux User Map named 'user'. The top navigation bar includes a back arrow, the title 'SELinux User Maps > user', and a search bar. Below the title is a 'Settings' tab and a row of buttons: Refresh, Reset, and Update. A 'GENERAL' section is expanded, showing the rule name 'user', an empty 'Description' field, and the SELinux User set to 'unconfined_u:s0-s0:c0.c1023'. The HBAC Rule is set to 'web_admin', and the status is 'Enabled'. There is also an 'undo' button.

SELinux User Maps > user

SELINUX USER MAP: user

Settings

Refresh Reset Update

▼ GENERAL

Rule name: **user**

Description:

SELinux User: *

HBAC Rule:

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.

The screenshot shows the SELinux User Map configuration interface. At the top, it displays the SELinux User Map name: "SELINUX USER MAP: user_u". Below this, there are tabs for "Settings" and "Refresh", and status indicators for "Enabled" and "Disabled".

USER Section:

- User category the rule applies to: Specified Users and Groups (radio button selected).
- Users: A list containing "jsmith".
- Add button (+ Add) is highlighted with a yellow box.

HOST Section:

- Host category the rule applies to: Specified Hosts and Groups (radio button selected).
- Hosts: A list containing "test.example.com".
- Add button (+ Add) is highlighted with a yellow box.
- Host Groups: An empty list.
- Add button (+ Add) is highlighted with a yellow box.

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.

This dialog box allows selecting users to add to the SELinux User Map. It has two main sections: "Available" and "Prospective".

- Available Section:** Contains a list of users: bguster, bjensen, jrockford, ljones, mreynolds (selected), and sspencer.
- Prospective Section:** Contains a list of users: (empty).
- Buttons between the sections: ">>" and "<<".
- Bottom buttons: "Add" and "Cancel".

**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.

26.3.2. 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 26.3. 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 26.4. 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 25, Configuring Host-Based Access Control](#).

Example 26.5. 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 26.6. 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=jsmithselinux1
```

Chapter 27. Defining Automatic Group Membership for Users and Hosts

Most of the policies and configuration within the Identity Management domain are based on *groups*. Various settings, such as sudo rules, automount, or access control, are defined for groups. These settings are then applied to individual 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.

27.1. 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 class entered by the administrator.

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 of 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.

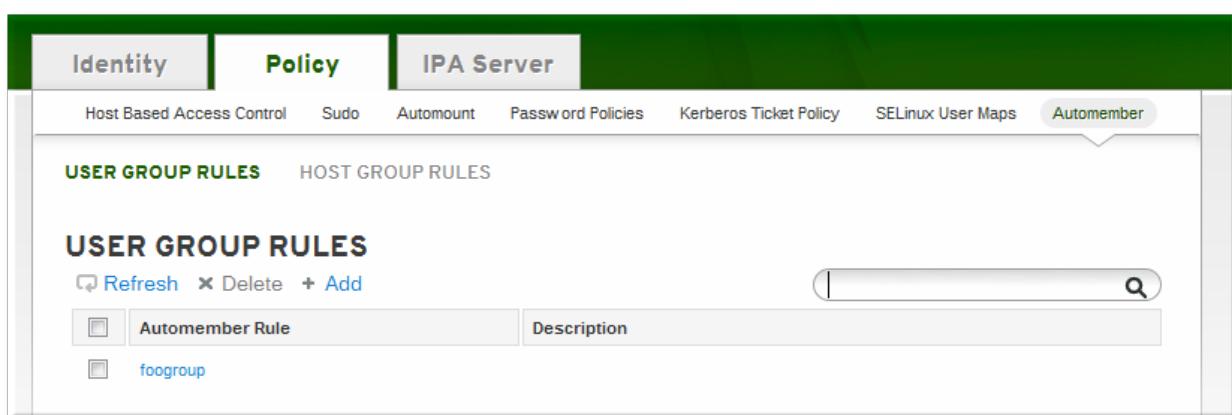
Once the structure is in place, then several things are clear:

- » 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

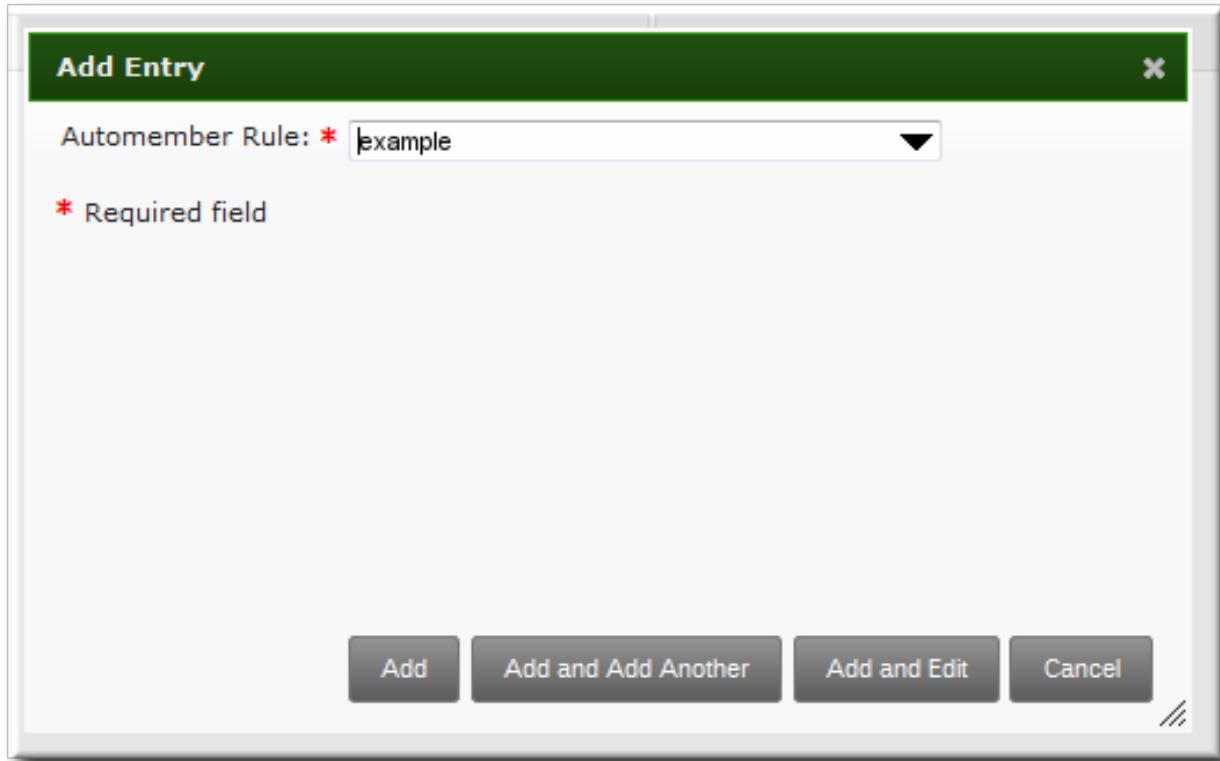
27.2. Defining Automembership Rules (Basic Procedure)

27.2.1. From the Web UI

1. Create the user group ([Section 10.2.2.1, “Creating User Groups”](#)) or host group ([Section 14.6.1.1, “Creating Host Groups from the Web UI”](#)).
2. Open the **Policy** tab, and select the **Automembers** subtab.
3. In the top of the **Automembers** area, select the type of autogroup to create, either **USER GROUP RULES** or **HOST GROUP RULES**.



4. In the drop-down menu, select the group for which to create the automember rule.



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.

USER GROUP RULE: foogroup

GENERAL

Attribute	Expression	Action
cn	^user[0-9]+	+ Add

EXCLUSIVE

Attribute	Expression	Action
cn	^user5	+ Add
cn	^user6	+ Add

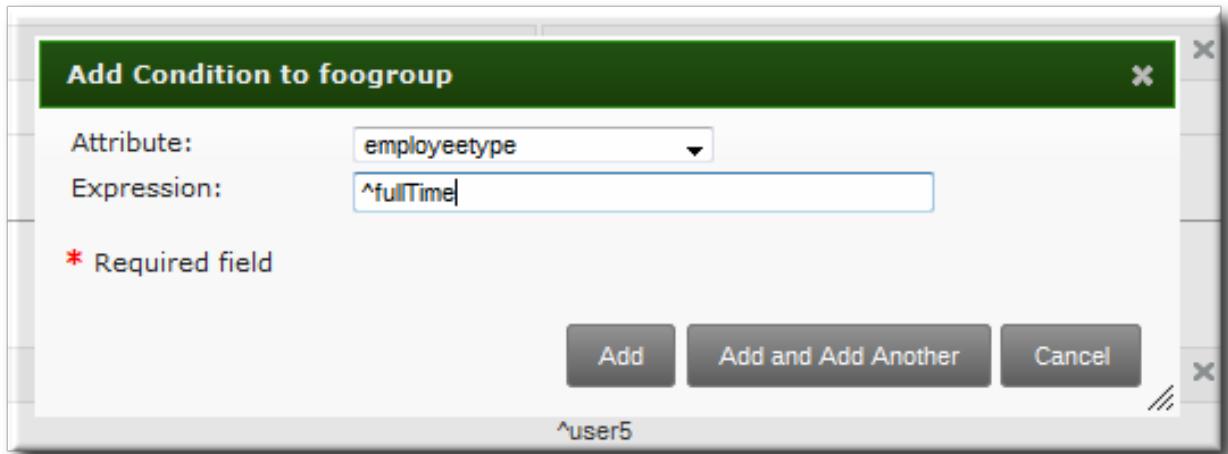
7. Select the attribute to use as the basis for the search and then set the regular expression to use to match the attribute value.

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.



- 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.

27.2.2. 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:

- Create the user group ([Section 10.2.2.1.2, “With the Command Line”](#)) or host group ([Section 14.6.1.2, “Creating Host Groups from the Command Line”](#)).
- 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
```

- 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 webservers
```

To remove the entire rule, simply run the **automember-del** command.

27.3. 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 not match an automember group and would, therefore, not be added to the fallback group.

27.3.1. 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](#).

27.3.2. 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
```

27.3.3. 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
```

Part IV. Configuring the Identity Management Server

Chapter 28. 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.

28.1. 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.

28.1.1. 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.

28.2. 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.

28.2.1. 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.

The screenshot shows the IPA Server interface with the 'Identity' tab selected. In the 'Role Based Access Control' section, the 'Self Service Permissions' option is highlighted. On the right, there is a list of permissions with three buttons at the bottom: 'Refresh', 'Delete', and a redboxed '+ Add' button.

Permissions
<input type="checkbox"/> Users can manage their own SSH public keys
<input type="checkbox"/> Self can write own password
<input type="checkbox"/> User Self service

Figure 28.1. Adding a New Self-Service Rule

3. Enter the name of the rule in the pop-up window. Spaces are allowed.

The dialog box has a title 'Add Self Service Permission' and a close button. It contains a 'Self-service * name' field with 'Adding Personal Info' and an 'Attributes *' section with a 'Filter' input and an 'Add' button. Below are two columns of attributes with checkboxes:

Attributes *	
<input type="checkbox"/> audio	<input type="checkbox"/> businesscategory
<input type="checkbox"/> carlicense	<input type="checkbox"/> cn
<input type="checkbox"/> departmentnumber	<input type="checkbox"/> description
<input type="checkbox"/> homedirectory	<input type="checkbox"/> homephone
<input type="checkbox"/> homepostaladdress	<input type="checkbox"/> inetuserhttppurl
<input type="checkbox"/> inetuserstatus	<input checked="" type="checkbox"/> initials
<input type="checkbox"/> internationalisdnnumber	<input type="checkbox"/> ipasshpubkey
<input type="checkbox"/> ipatokenradiusconfiglink	<input type="checkbox"/> ipatokenradiususername
<input type="checkbox"/> ipauniqueid	<input type="checkbox"/> ipauserauthtype
<input checked="" type="checkbox"/> jpegphoto	<input type="checkbox"/> krbcanonicalname

* Required field

Buttons at the bottom: Add, Add and Add Another, Add and Edit, Cancel.

Figure 28.2. 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.

28.2.2. 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
```

28.2.3. 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

Attributes *	
<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"/> inetuserhttppurl
<input type="checkbox"/> inetuserstatus	<input checked="" type="checkbox"/> initials

Figure 28.3. 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.

28.3. 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.

28.3.1. 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.

The screenshot shows the IPA Server interface with the 'IPA Server' tab selected. In the 'Role Based Access Control' dropdown, 'Delegations' is highlighted. On the right, there are buttons for 'Refresh', 'Delete', and '+ Add'. The '+ Add' button is highlighted with a red box.

Figure 28.4. 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 *	managers																		
Permissions	<input checked="" type="checkbox"/> read <input checked="" type="checkbox"/> write																		
User group *	managers																		
Member user group	employees																		
Attributes *	<input type="text" value="Filter"/> <input type="button" value="Add"/> <table border="0"> <tr> <td><input type="checkbox"/> audio</td> <td><input checked="" type="checkbox"/> businesscategory</td> </tr> <tr> <td><input checked="" type="checkbox"/> carlicense</td> <td><input type="checkbox"/> cn</td> </tr> <tr> <td><input checked="" type="checkbox"/> departmentnumber</td> <td><input type="checkbox"/> description</td> </tr> <tr> <td><input type="checkbox"/> destinationindicator</td> <td><input type="checkbox"/> displayname</td> </tr> <tr> <td><input type="checkbox"/> employeenumber</td> <td><input checked="" type="checkbox"/> employeetype</td> </tr> <tr> <td><input checked="" type="checkbox"/> facsimiletelephonenumber</td> <td><input type="checkbox"/> gecos</td> </tr> <tr> <td><input type="checkbox"/> gidnumber</td> <td><input checked="" type="checkbox"/> givenname</td> </tr> <tr> <td><input type="checkbox"/> homedirectory</td> <td><input type="checkbox"/> homephone</td> </tr> <tr> <td><input type="checkbox"/> homepostaladdress</td> <td><input type="checkbox"/> inetuserhttpurl</td> </tr> </table>	<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory	<input checked="" type="checkbox"/> carlicense	<input type="checkbox"/> cn	<input checked="" type="checkbox"/> departmentnumber	<input type="checkbox"/> description	<input type="checkbox"/> destinationindicator	<input type="checkbox"/> displayname	<input type="checkbox"/> employeenumber	<input checked="" type="checkbox"/> employeetype	<input checked="" type="checkbox"/> facsimiletelephonenumber	<input type="checkbox"/> gecos	<input type="checkbox"/> gidnumber	<input checked="" type="checkbox"/> givenname	<input type="checkbox"/> homedirectory	<input type="checkbox"/> homephone	<input type="checkbox"/> homepostaladdress	<input type="checkbox"/> inetuserhttpurl
<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory																		
<input checked="" type="checkbox"/> carlicense	<input type="checkbox"/> cn																		
<input checked="" type="checkbox"/> departmentnumber	<input type="checkbox"/> description																		
<input type="checkbox"/> destinationindicator	<input type="checkbox"/> displayname																		
<input type="checkbox"/> employeenumber	<input checked="" type="checkbox"/> employeetype																		
<input checked="" type="checkbox"/> facsimiletelephonenumber	<input type="checkbox"/> gecos																		
<input type="checkbox"/> gidnumber	<input checked="" type="checkbox"/> givenname																		
<input type="checkbox"/> homedirectory	<input type="checkbox"/> homephone																		
<input type="checkbox"/> homepostaladdress	<input type="checkbox"/> inetuserhttpurl																		

* Required field

Figure 28.5. Form for Adding a Delegation

- In the **Member user group** drop-down menu, select the group whose entries can be edited by members of the delegation group.
- In the attributes box, select the checkboxes by the attributes to which the member user group is being granted permission.
- Click the **Add** button to save the new delegation ACI.

28.3.2. 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.

28.4. 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.

28.4.1. Roles

28.4.1.1. 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.

<input type="checkbox"/>	Role name	Description
<input type="checkbox"/>	IT Security Specialist	IT Security Specialist
<input type="checkbox"/>	IT Specialist	IT Specialist
<input type="checkbox"/>	Security Architect	Security Architect

Figure 28.6. Adding a New Role

3. Enter the role name and a description.

Add Role

Role name *

Example Role

Description

For engineers

* Required field

Add Add and Add Another Add and Edit Cancel

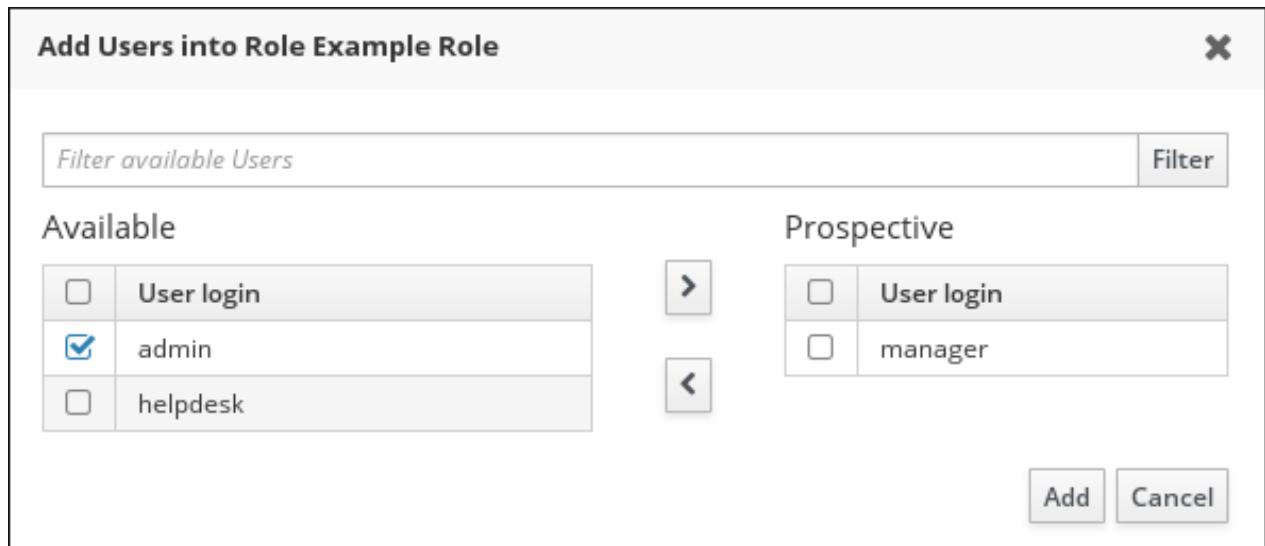
Figure 28.7. Form for Adding a Role

4. Click the **Add and Edit** button to save the new role and go to the configuration page.
5. At the top of the **Users** tab, or in the **Users Groups** tab when adding groups, click **Add**.

Example Role members:					
Users (1)	User Groups	Hosts	Host Groups	Services	Privileges
<input type="checkbox"/> Refresh	<input type="checkbox"/> Delete	<input type="checkbox"/> + Add			
<input type="checkbox"/> User login					
<input type="checkbox"/> employee					
Showing 1 to 1 of 1 entries.					

Figure 28.8. Adding Users

6. Select the users on the left and use the > button to move them to the **Prospective** column.

**Figure 28.9. Selecting Users**

- At the top of the **Privileges** tab, click **Add**.

Example Role members:						
<input type="button" value="Users (1)"/>	<input type="button" value="User Groups"/>	<input type="button" value="Hosts"/>	<input type="button" value="Host Groups"/>	<input type="button" value="Services"/>	<input type="button" value="Privileges (4)"/>	<input type="button" value="Settings"/>
<input type="button" value="Refresh"/>	<input type="button" value="Delete"/>	<input style="border: 2px solid red;" type="button" value="+ Add"/>				
<input type="checkbox"/>	Privilege name					
<input type="checkbox"/>	Automount Administrators					
<input type="checkbox"/>	Certificate Administrators					
<input type="checkbox"/>	DNS Administrators					

Figure 28.10. Adding Privileges

- Select the privileges on the left and use the **>** button to move them to the **Prospective** column.

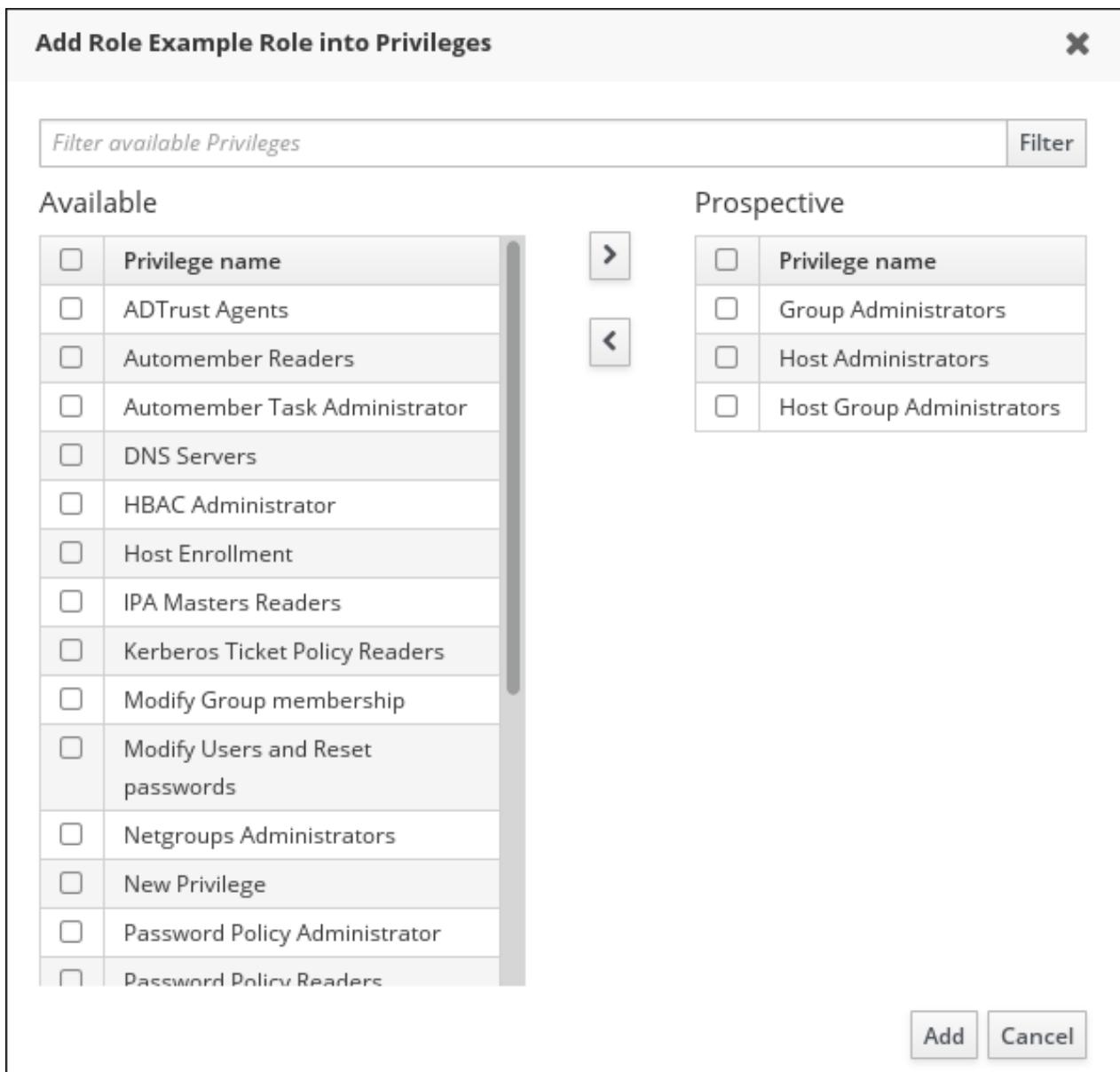


Figure 28.11. Selecting Privileges

- Click the **Add** button to save.

28.4.1.2. Creating Roles in the Command Line

- 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
```

- 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
```

28.4.2. Permissions

28.4.2.1. 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.

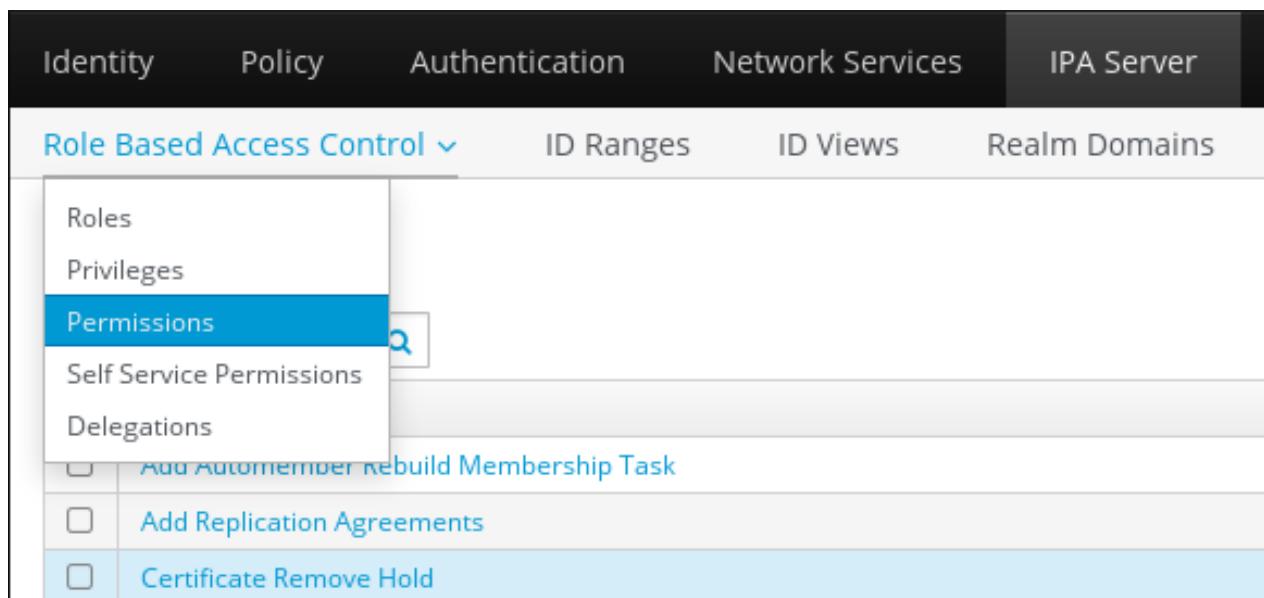


Figure 28.12. Permissions Task

3. Click the **Add** button at the top of the list of permissions.

Permissions	
<input type="text"/> Search	
	Refresh
	Delete
	Add
<input type="checkbox"/>	Permission name
<input type="checkbox"/>	Add Automember Rebuild Membership Task
<input type="checkbox"/>	Add Replication Agreements
<input type="checkbox"/>	Certificate Remove Hold

Figure 28.13. Adding a New Permission

4. Define the properties for the new permission in the form that shows up.

Add Permission

Permission name *

Bind rule type permission all anonymous

Granted rights * read search compare
 write add delete
 all

Type

Subtree *

Extra target filter

Target DN

Member of group

Effective attributes

* Required field

Figure 28.14. 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.

28.4.2.2. 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 option 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=testrealm,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.

28.4.2.3. 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

<input type="checkbox"/> read	<input type="checkbox"/> search	<input type="checkbox"/> compare	<input checked="" type="checkbox"/> write
<input type="checkbox"/> add	<input type="checkbox"/> delete	<input type="checkbox"/> all	

Figure 28.15. 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"
```

28.4.2.4. 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.

28.4.3. Privileges

28.4.3.1. Creating New Privileges from the Web UI

- Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
- Select the **Privileges** task link.

	Description
<input type="checkbox"/> Password Policy Readers	Read password policies
<input type="checkbox"/> RBAC Readers	Read roles, privileges, permissions and ACIs
<input type="checkbox"/> Replication Administrators	Replication Administrators

Figure 28.16. Privileges Task

3. Click the **Add** link at the top of the list of privileges.

Privileges		
Search		
<input type="checkbox"/>	Privilege name	Description
<input type="checkbox"/>	Password Policy Readers	Read password policies
<input type="checkbox"/>	RBAC Readers	Read roles, privileges, permissions and ACIs
<input type="checkbox"/>	Replication Administrators	Replication Administrators

Figure 28.17. Adding a New Privilege

4. Enter the name and a description of the privilege.

The form has two main sections: 'Privilege name *' with a value of 'New Privilege' and 'Description' with a value of 'For employees'. Below the form is a note '* Required field'. At the bottom are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

Privilege name *	New Privilege
Description	For employees

* Required field

Add Add and Add Another Add and Edit Cancel

Figure 28.18. Form for Adding a Privilege

5. Click the **Add and Edit** button to go to the privilege configuration page to add permissions.
6. Select the **Permissions** tab.
7. Click **Add** at the top of the list of permissions to add permission to the privilege.

Privilege: New Privilege

The screenshot shows a user interface for managing privileges. At the top, there are three tabs: 'Permissions (5)', 'Settings', and 'Roles'. Below the tabs is a toolbar with buttons for 'Refresh' and 'Delete', followed by a prominent red-bordered 'Add' button labeled '+ Add'. The main area displays a list of permissions with checkboxes next to them. The permissions listed are: 'System: Add Groups', 'System: Add HBAC Rule', 'System: Add Hosts', 'System: Add Privileges', and 'System: Add Roles'. All these items are currently in the 'Available' state, indicated by blue text.

	Permission name
<input type="checkbox"/>	System: Add Groups
<input type="checkbox"/>	System: Add HBAC Rule
<input type="checkbox"/>	System: Add Hosts
<input type="checkbox"/>	System: Add Privileges
<input type="checkbox"/>	System: Add Roles

Figure 28.19. 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.

This screenshot shows a modal dialog titled 'Add Privilege New Privilege into Permissions'. At the top, there is a search bar labeled 'Filter available Permissions' with a 'Filter' button. Below the search bar are two columns: 'Available' on the left and 'Prospective' on the right. The 'Available' column lists various system permissions with checkboxes. The 'Prospective' column lists the same permissions without checkboxes, indicating they have been moved. Between the two columns are two large grey arrows: a right-pointing arrow above a left-pointing arrow, used for moving items between the lists.

	Available	Prospective
<input type="checkbox"/>	Policy costemplate	<input type="checkbox"/> Permission name
<input type="checkbox"/>	System: Add HBAC Service Groups	<input type="checkbox"/> System: Add Services
<input type="checkbox"/>	System: Add HBAC Services	<input type="checkbox"/> System: Add Sudo Command
<input type="checkbox"/>	System: Add Hostgroups	Command
<input type="checkbox"/>	System: Add krbPrincipalName to a Host	
<input type="checkbox"/>	System: Add Netgroups	
<input type="checkbox"/>	System: Add SELinux User Maps	
<input type="checkbox"/>	System: Add Sudo Command Group	
<input type="checkbox"/>	System: Delete Group Password Policy	

Figure 28.20. Selecting Permissions

9. Click the **Add** button to save.

28.4.3.2. 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 29. 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.

29.1. A Reference of IdM Server Configuration Files and Directories

Table 29.1. IdM Server Configuration Files and Directories

Directory or File	Description
Server Configuration	
/etc/ipa/	The main IdM configuration directory.
/etc/ipa/default.conf	The primary configuration file for IdM.
/etc/ipa/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 ipa 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 <i>system user</i> 's home directory the first time the user runs an IdM command.
IdM Logs	
~/.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.
System Services	
/etc/rc.d/init.d/ipa/	The IdM server init script.
Web UI	
/etc/ipa/html/	A symlink directory in the main configuration directory for the HTML files used by the IdM web UI.

Directory or File	Description
/etc/httpd/conf.d/ipa.conf	The configuration files used by the Apache host for the web UI application.
/etc/httpd/conf.d/ipa-rewrite.conf	
/etc/httpd/conf/ipa.keytab	The keytab file used by the web UI service.
/usr/share/ipa/	The main directory for all of the HTML files, scripts, and stylesheets used by the web UI.
/usr/share/ipa/ipa-rewrite.conf	The configuration files used by the Apache host for the web UI application.
/usr/share/ipa/ipa.conf	
/usr/share/ipa/updates/	Contains any updated files, schema, and other elements for Identity Management.
/usr/share/ipa/html/	Contains the HTML files, JavaScript files, and stylesheets used by the web UI.
/usr/share/ipa/ipaclient/	Contains the JavaScript files used to access Firefox's autoconfiguration feature and set up the Firefox browser to work in the IdM Kerberos realm.
/usr/share/ipa/migration/	Contains HTML pages, stylesheets, and Python scripts used for running the IdM server in migration mode.
/usr/share/ipa/ui/	Contains all of the scripts used by the UI to perform IdM operations.
/var/log/httpd/	The log files for the Apache web server.
Kerberos	
/etc/krb5.conf	The Kerberos service configuration file.
SSSD	
/usr/share/sssd/sssd.api.d/sssd-ipa.conf	The configuration file used to identify the IdM server, IdM Directory Server, and other IdM services used by SSSD.
/var/log/sssd/	The log files for SSSD.
389 Directory Server	
/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.
Dogtag Certificate System	
/etc/pki-ca/	The main directory for the IdM CA instance.
/var/lib/pki/pki-tomcat/conf/ca/CS.cfg	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.
Cache Files	

Directory or File	Description
/var/cache/ipa/	Cache files for the IdM server and the IdM Kerberos password daemon.
System Backups	
/var/lib/ipa/sysrestore/	Contains backups of all of the system files and scripts that were reconfigured when the IdM server was installed. These include the original .conf files for NSS, Kerberos (both krb5.conf and kdc.conf), and NTP.
/var/lib/ipa-client/sysrestore/	Contains backups of all of the system files and scripts that were reconfigured when the IdM client was installed. Commonly, this is the sssd.conf file for SSSD authentication services.

29.2. 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)
- » tomcat
- » 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 29.1. 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: the policies for the **named** and **tomcat** services. 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 **tomcat** 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.

29.3. 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.

29.4. 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 29.2. 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>	These are standard access and error logs for Apache servers. Both the web UI and the XML-RPC command-line interface use Apache, so some IdM-specific messages will be recorded in the error log along with the Apache messages.	Apache log chapter
	<code>/var/log/httpd/error_log</code>		
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>	These logs mainly relate to certificate operations. In IdM, this is used for service principals, hosts, and other entities which use certificates.	Logging chapter
	<code>/var/log/pki-ca/system</code>		
	<code>/var/log/pki-ca/transactions</code>		
	<code>/var/log/pki-ca/signedAudit</code>		

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. » Monitoring servers and databases » Log entries explained
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. » Monitoring servers and databases » Log entries explained
Kerberos	/var/log/krb5libs.log	This is the primary log file for Kerberos connections.	This location is configured in the krb5.conf file, so it could be different on some systems.
Kerberos	/var/log/krb5kdc.log	This is the primary log file for the Kerberos KDC server.	This location is configured in the krb5.conf file, so it could be different on some systems.
Kerberos	/var/log/kadmind.log	This is the primary log file for the Kerberos administration server.	This location is configured in the krb5.conf file, so it could be different on some systems.

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 querylog command:</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <code>/usr/sbin/rndc querylog</code> </div> <p>This begins writing log messages to the system's /var/log/messages file. To turn off logging, run the querylog command again.</p>

29.4.1. 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`

29.4.2. 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+gAwIBBaEZGxdSSFRTLKV0Ry5CT1MuUkVESEFULKNPTaI5MDegAwIBA6EwMC4bBEhUVFAb
JmRlbGwtcGUx0DUwLTAxLnJodHMuZW5nLmJvcy5yZWRoYXQuY29to4IBIDCCARygAwIBEqED
AgECooIBDgSCAQpV2YEWv03X+SENDUBf0hMFGc3Fvnd51nELV0rIB1tfGVjpNlkuQxXKSfFK
VD3vyAUqkii255T0mnXyTwayE93W1U4s0shetmG50zeU4KDmhupzQZScb5xB0KPU4HMDvP1
UnDFJUGCk9tcqDJiYE+lJrEc8H+Vxxvl+nP6yIdUQKqoEuNhJaLWIiT8ieAzk8zvmDlDzpF
YtInGWe9D5ko1Bb7Npu0SEpdVJB2gnB5vszCIdLlzHM4JUqX8p21AZV0UYA6QZ0WX90XhqHd
ElKcuHCN2s9FBRoFYK83gf1voS7xSFlzzAfsEGHNmdA0qXbzREKGqUr8fmWmNvBGpDir2ILQ
ep091L56JqSCA8owggPGoAMCARKigg09BIIDuarbB67zjmBu9Ax2K+0k1SD99pNv97h9yxol
8c6NGLB4CmE8Mo39rL4MMXHe0S00Cbn+TD97XVGLu+cgkfVcuIG4PMMBoajuSnPmIf7qDvfa
8IYDFlDDnRB7I//IXtCc/Z4rBbaxk0SMIRLrsKf5wha7aWtN1JbizBMQw+J0UlN8JjsWxu0L
s75hBtIDbPf3fva3vwBf7kTBChBsheelSalck9qUglyNxAOdgFVvRrXbfkw51Uo++9qHnhh+
zFSWepfv7US7RYK0Kx0Fd+uaY1ES+xlnMvK18ap2pcy0odBkKu1kwJDND0JXUdSY08MxK2
zb/UGWrVEf6GIIsBgu122UGiHp6C+0fEu+nRrvt0RY65Bgi8E1vm55fbb/9dQWNcQheL9m6QJ
WPw0rgc+E5S0990N6x3Vv2+Zk17EmbZXinPd2tDe7fJ9cS9o/z7Qjw8z8vvSzHL4GX7FKi2H
JdBST3nEg0C8Pq046UnAJcA8pf1ZkwCK9xDWH+5PSph6WnvpRquqgf/6cq+3jk3MEjCrx+JB
J8QL6AgN3oEB4kvjZpAC+FftkdX59VLDwfL/r0gMw3Znk0nLLCLkiYUMTEHZBzJw9kFbsX3
LmS8qQQA6rQ2L782DYisElywfZ/0Sax8J0/G62Zvy7BHy7SQSGIVcdA0afeNyfxaWm1vTpVs
h0GrnllYfs3FhZAKnVcLYrlPTapR23uLgRMv+0c9wAbwuUDfKg0015vAd1j55VUyapgDEzi/
URsLdVdcF4zigt4KrTByCwU2/pI6FmEPqB2tsjM2A8JmqA+9Nt8bjanNdNwCOWE0dF50zeL9P

```

```

800dWGkbRZLk4DLIurpCW1d6IyTBhPQ5qZqHJWeoGiFa5y94zBpp27goMPmE0BskXT0JQmve
Yfl0eKEMSzyiWPL2mwi7KEMtfgCpwTIGP2LRE/QxNvPGkwFf0+PDjZGVw+APKkMKqclVXxht
JA/2NmBr01pZIIJ9R+41sR/QoACcXIUXJnhrTwwR1viKCB5Tec87gN+e0Cf0g+fmZuXNRscw
JfhYQJYwJqdYzGtZW+h8jDWqa2EPcDwIQwyFAgXNQ/aMvh1yNTECpLEgrMhYmFAUDLQzI2BD
nfbDftIs0rXjSC0oZn/Uaoqdr4F5sy0rYAxH47bS6MW8CxxylreH8nt2qXjenakLFHcNjt4
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+iejb4oAMCARKiQRvVl5x6Zt9PbWN
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>krbprincipalname</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>196680004</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>196680004</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 30. 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.

30.1. 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
```

30.2. 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.

30.3. 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 and replicas to update them with the information about the new certificate from LDAP. You must run **ipa-certupdate** on every client and replica separately.

To manually renew the Dogtag CA certificate signed by an external CA:

1. Run the **ipa-cacert-manage renew --external-ca** 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 2.3.5, “Installing a Server with an External CA as the Root 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 and replicas to update them with the information about the new certificate from LDAP. You must run **ipa-certupdate** on every client and replica 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
```

30.4. Manual CA Certificate Installation

The **ipa-cacert-manage install** command allows you to install a new certificate to IdM. For example, this enables you to change the current certificate when it is nearing its 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.

30.5. 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 30.3, “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**.

30.6. 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
```

30.7. 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
```

30.8. 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' '(&(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: (&(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
```

30.9. 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.

30.9.1. 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
```

30.9.2. 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
```

Chapter 31. 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 32. Managing Replicas and Replication Agreements

This chapter provides details on replication agreements and describes how to manage them.

Note

For guidelines on setting up additional replication agreements, see [Section 3.2, “Recommendations for Planning the Replica Topologies”](#).

32.1. Explaining Replication Agreements

Replicas are joined in a *replication agreement* that copies data between them. Replication agreements are bilateral: the data is replicated from the first replica to the other one as well as from the other replica to the first one.

Note

An initial replication agreement is set up between two replicas by the **ipa-replica-install** script. See [Chapter 3, Setting up IdM Replicas](#) for details on installing the initial replica.

Types of Replication Agreements

Identity Management supports the following three types of replication agreements:

- ▶ Replication agreements to replicate directory data, such as users, groups, and policies. You can manage these agreements using the **ipa-replica-manage** utility.
- ▶ Replication agreements to replicate certificate server data. You can manage these agreements using the **ipa-csreplica-manage** utility.
- ▶ Synchronization agreements to replicate user information with an Active Directory server. These agreements are not described in this guide. For documentation on synchronizing IdM and Active Directory, see the [Windows Integration Guide](#).

The **ipa-replica-manage** and **ipa-csreplica-manage** utilities use the same format and arguments. The following sections of this chapter describe the most notable replication management operations performed using these utilities. For detailed information about the utilities, see the **ipa-replica-manage(1)** and **ipa-csreplica-manage(1)** man pages.

32.2. Listing Replication Agreements

To list the directory data replication agreements currently configured for a replica, use the **ipa-replica-manage list** command:

1. Run **ipa-replica-manage list** without any arguments to list all replicas in the replication topology. In the output, locate the required replica:

```
$ ipa-replica-manage list
server1.example.com: master
server2.example.com
server3.example.com
server4.example.com
```

2. Add the replica's host name to **ipa-replica-manage list** to list the replication agreements.

```
$ ipa-replica-manage list server1.example.com
server2.example.com
server3.example.com
```

The output displays the replicas to which **server1.example.com** sends updates.

To list certificate server replication agreements, use the **ipa-csreplica-manage list** command.

32.3. Creating and Removing Replication Agreements

Creating Replication Agreements

To create a new replication agreement, use the **ipa-replica-manage connect** command:

```
$ ipa-replica-manage connect server1.example.com server2.example.com
```

The command creates a new bilateral replication agreement going from *server1.example.com* to *server2.example.com* and from *server2.example.com* to *server1.example.com*.

If you only specify one server with **ipa-replica-manage connect**, IdM creates a replication agreement between the local host and the specified server.

To create a new certificate server replication agreement, use the **ipa-csreplica-manage connect** command.

Removing Replication Agreements

To remove a replication agreement, use the **ipa-replica-manage disconnect** command:

```
$ ipa-replica-manage disconnect server1.example.com server4.example.com
```

This command disables replication from *server1.example.com* to *server4.example.com* and from *server4.example.com* to *server1.example.com*.

The **ipa-replica-manage disconnect** command only removes the replication agreement. It leaves both servers in the Identity Management replication topology. To remove all replication agreements and data related to a replica, use the **ipa-replica-manage del** command, which removes the replica entirely from the Identity Management domain.

```
$ ipa-replica-manage del server2.example.com
```

To remove a certificate server replication agreement, use the **ipa-csreplica-manage disconnect** command. Similarly, to remove all certificate replication agreements and data between two servers, use the **ipa-csreplica-manage del** command.

32.4. Initiating a Manual Replication Update

Data changes between replicas with direct replication agreements between each other are replicated almost instantaneously. However, replicas that are not joined in a direct replication agreement do not receive updates as quickly.

In some situations, it might be necessary to manually initiate an unplanned replication update. For example, before taking a replica offline for maintenance, all the queued changes waiting for the planned update must be sent to one or more other replicas. In this situation, you can initiate a manual replication update before taking the replica offline.

To manually initiate a replication update, use the **ipa-replica-manage force-sync** command. The local host on which you run the command is the replica that receives the update. To specify the replica that sends the update, use the **--from** option.

```
$ ipa-replica-manage force-sync --from server1.example.com
```

To initiate a replication update for certificate server data, use the **ipa-csreplica-manage force-sync** command.

32.5. Re-initializing a Replica

If a replica has been offline for a long period of time or its database has been corrupted, you can *re-initialize* it. Re-initialization is analogous to initialization, which is described in [Section 3.4, “Creating the Replica”](#). Re-initialization refreshes the replica with an updated set of data.

Note

Waiting for a regular replication update or initiating a manual replication update will not help in this situation. During these replication updates, replicas only send changed entries to each other. Unlike re-initialization, replication updates do not refresh the whole database.

To re-initialize a data replication agreement on a replica, use the **ipa-replica-manage re-initialize** command. The local host on which you run the command is the re-initialized replica. To specify the replica from which the data is obtained, use the **--from** option:

```
$ ipa-replica-manage re-initialize --from server1.example.com
```

To re-initialize a certificate server replication agreement, use the **ipa-csreplica-manage re-initialize** command.

32.6. Removing a Replica

Deleting or *demoting* a replica removes the IdM replica from the topology so that it no longer processes IdM requests. It also removes the host machine itself from the IdM domain.

To delete a replica, perform these steps on the replica:

1. List all replication agreements for the IdM domain. In the output, note the host name of the replica.

```
$ ipa-replica-manage list
server1.example.com: master
server2.example.com: master
server3.example.com: master
server4.example.com: master
```

2. Use the **ipa-replica-manage del** command to remove all agreements configured for the replica as well as all data about the replica.

```
$ ipa-replica-manage del server3.example.com
```

3. If the replica was configured with its own CA, then also use the **ipa-csreplica-manage del** command to remove all certificate server replication agreements.

```
$ ipa-csreplica-manage del server3.example.com
```

Note

This step is only required if the replica itself was configured with a Red Hat Certificate System CA. It is not required if only the master server or other replicas were configured with a CA.

4. Uninstall the IdM server package.

```
$ ipa-server-install --uninstall -U
```

32.7. Renaming a Server Host System

It is not possible to change the host name of an IdM server after it was set up. However, you can replace the server with a replica with a different name.

1. Create a new replica of the server, with a CA and with the new required host name or IP address. This is described in [Chapter 3, Setting up IdM Replicas](#).
2. Stop the initial IdM server instance.

```
[root@old_server ~]# ipactl stop
```

3. Verify that all other replicas and clients are working as before.
4. Uninstall the initial IdM server, as described in [Section 2.4, “Uninstalling an IdM Server”](#).

Chapter 33. 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.

33.1. 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 33.1.2, “Planning Password Migration”](#).

33.1.1. 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.

33.1.1.1. 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.

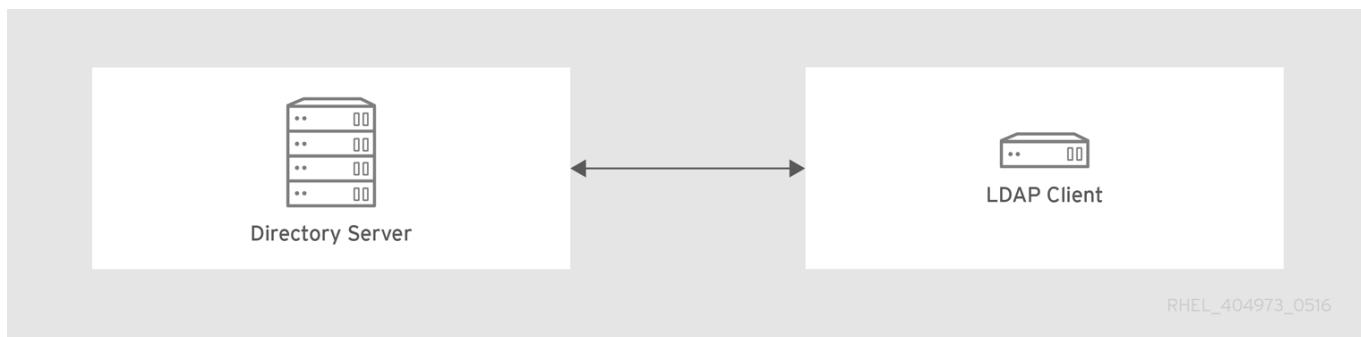


Figure 33.1. Basic LDAP Directory and Client Configuration

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 1.1.2, “Contrasting Identity Management with a Standard LDAP Directory”](#).) 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.

33.1.1.2. 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.

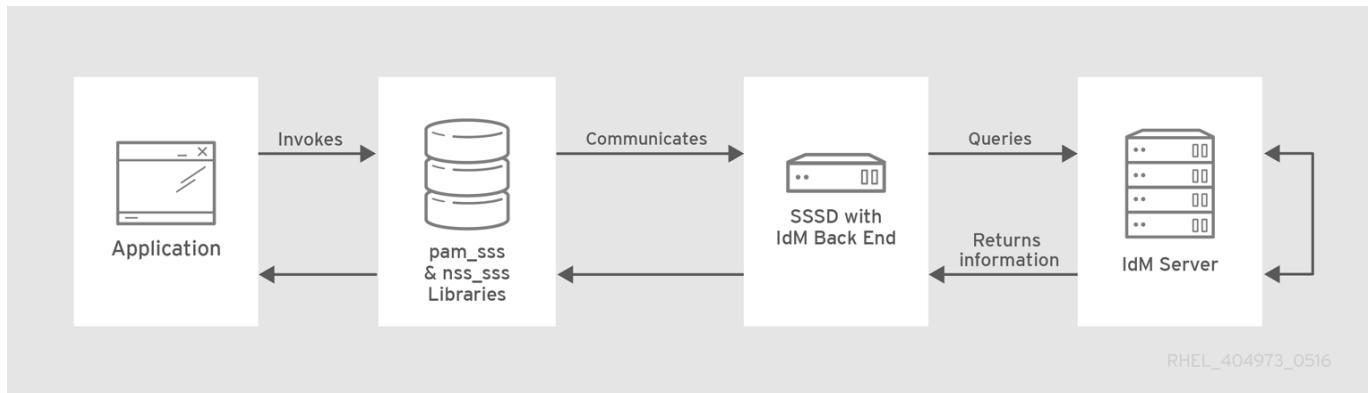


Figure 33.2. Clients and SSSD with an IdM Backend

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 33.1.1.3, “Alternative Supported Configuration”](#).

33.1.1.3. 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`).

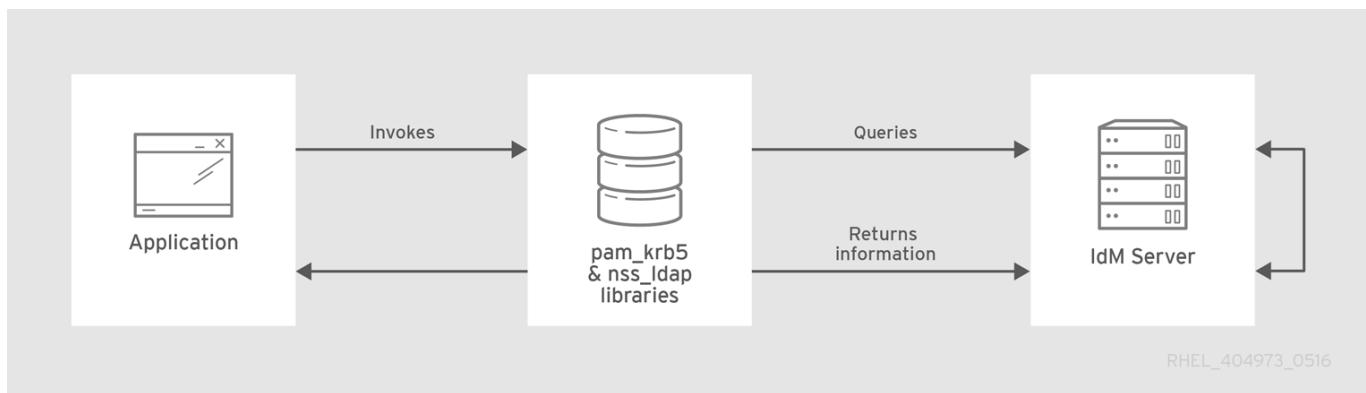


Figure 33.3. Clients and IdM with LDAP and Kerberos

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).

33.1.2. 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 [7], 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.

33.1.2.1. 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.

33.1.2.2. 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.

33.1.2.3. 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.

33.1.2.4. 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 33.1.2.2, “Method 2: Using the Migration Web Page”](#) and [Section 33.1.2.3, “Method 3: Using SSSD \(Recommended\)”](#).

33.1.2.5. 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:
```

33.1.3. 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.

33.1.3.1. 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.

33.1.3.2. 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.

33.1.3.3. 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.

33.1.3.4. 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 **--bind-dn** 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.

33.1.3.5. 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.

33.1.3.6. 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.

33.2. 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` command 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.

Note

By default, `ipa migrate-ds` uses the bind DN "`cn=Directory Manager`". Pass the `-bind-dn` option to the command to specify an individual bind DN. See [Section 33.1.3.4, “Migration Tools”](#) for further information.

33.2.1. 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.

33.2.2. 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
```

33.2.3. Excluding Entry Attributes

By default, every attribute and object class for a user or group entry is migrated. There are some cases where that may not be realistic, either because of bandwidth and network constraints or because the attribute data are no longer relevant. For example, if users are going to be assigned new user certificates as they join the IdM domain, then there is no reason to migrate the **userCertificate** attribute.

Specific object classes and attributes can be ignored by the **migrate-ds** by using any of several different options:

- » **--user-ignore-objectclass**
- » **--user-ignore-attribute**
- » **--group-ignore-objectclass**
- » **--group-ignore-attribute**

For example, to exclude the **userCertificate** attribute and **strongAuthenticationUser** object class for users and the **groupOfCertificates** object class for groups:

```
[root@ipaserver ~]# ipa migrate-ds --user-ignore-attribute=userCertificate --user-ignore-objectclass=strongAuthenticationUser --group-ignore-objectclass=groupOfCertificates ldap://ldap.example.com:389
```

Note

Make sure not to ignore any required attributes. Also, when excluding object classes, make sure to exclude any attributes which are only supported by that object class.

33.2.4. 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
```

33.3. 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 [8], 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 33.2, “Examples for Using migrate-ds”](#).

Once the information is exported, the script adds all required IdM object classes and attributes and converts DNs 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.

33.4. 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 [9], 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 33.2, “Examples for Using migrate-ds”](#).

Once the information is exported, the script adds all required IdM object classes and attributes and converts DNs 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 33.1.2, “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.

33.5. Scenario 3: Migrating over SSL

Both migrating using SSSD ([Section 33.3, “Scenario 1: Using SSSD as Part of Migration”](#)) and migrating directly from LDAP ([Section 33.4, “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.

- Restart the IdM Apache instance. The SSL configuration is loaded through the Apache server.

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

- Go through any required migration preparation and run the **ipa migrate-ds** script, as described in [Section 33.3, “Scenario 1: Using SSSD as Part of Migration”](#) and [Section 33.4, “Scenario 2: Migrating an LDAP Server Directly to Identity Management”](#).
- 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.
- Restart the IdM Apache instance to load the updated SSL configuration.

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

[7] 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.

[8] There is limited support for custom user and group schema in Identity Management.

[9] There is limited support for custom user and group schema in Identity Management.

Appendix A. Troubleshooting Identity Management

For troubleshooting advice for:

- » servers, see [Section A.1, “Identity Management Servers”](#)
- » replicas, see [Section A.2, “Identity Management Replicas”](#)
- » clients, see [Section A.3, “Identity Management Clients”](#)
- » authentication, see [Section A.4, “Logging In and Authentication Problems”](#)

A.1. Identity Management Servers

A.1.1. External CA Installation Fails

The `ipa-server-install --external-ca` command fails 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
```

The `env|grep proxy` command displays variables such as the following:

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

What this means: The *_proxy environmental variables are preventing the server from being installed.

To fix 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
```

4. Retry running `ipa-server-install --external-ca`.

A.1.2. named Daemon Fails to Start

After installing an IdM server with integrated DNS, the **named-pkcs11** fails to start. The **/var/log/messages** file includes an error message 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
```

What this means: The **bind-chroot** package is installed and is preventing the **named-pkcs11** service from starting.

To fix the problem:

1. Uninstall the **bind-chroot** package.

```
# yum remove bind-chroot
```

2. Restart the IdM server.

```
# ipactl restart
```

A.2. Identity Management Replicas

This guide describes common replication problems for Identity Management in Red Hat Enterprise Linux.

Additional resources:

- ▶ For troubleshooting advice on replication in Red Hat Directory Server, see [Section “Solving Common Replication Conflicts”](#) in the *Directory Server Administration Guide*.
- ▶ For advice on how to test that replication is working, see [Section 3.4.5, “Testing the New Replica”](#).
- ▶ The Directory Server **repl-monitor** script shows in-progress status of replication, which can help you troubleshoot replication problems. For documentation on the script, see [Section “repl-monitor \(Monitors Replication Status\)”](#) in the *Directory Server Administration Guide*.

A.2.1. Replica Starts with SASL, GSS-API, and Kerberos Errors in the Directory Server Logs

When the replica starts, a series of SASL bind errors are recorded in the Directory Server (DS) logs. The errors state 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))  
...
```

Additionally, other messages can occur stating that the server could not obtain Kerberos credentials for the host principal:

```
set_krb5_creds - Could not get initial credentials for principal [ldap/
replica.example.com] in keytab [WRFILE:/etc/dirsrv/ds.keytab]: -
1765328324 (Generic error)
```

What this means: IdM uses GSS-API for Kerberos connections. The DS instance keeps the Kerberos credentials cache in memory. When the DS process ends, such as when the IdM replica stops, the credentials cache is destroyed.

When the replica restarts, DS starts before the KDC server starts. Because of this start order, the Kerberos credentials are not yet saved in the credentials cache when DS starts, which is what causes the errors.

After the initial failure, DS re-attempts to establish the GSS-API connection after the KDC starts. This second attempt is successful and ensures that the replica works as expected.

You can ignore the described startup errors as long as the GSS-API connection is successfully established and the replica works as expected. The following message shows that the connection was successful:

```
Replication bind with GSSAPI auth resumed
```

A.2.2. The DNS forward Record Does Not Match the Reverse Address

When configuring a new replica, installation fails with a series of certificate errors, followed by a DNS error stating the DNS forward record does not match the reverse address.

```
ipa: DEBUG: approved_usage = SSLServer intended_usage = SSLServer
ipa: DEBUG: cert valid True for "CN=replica.example.com,O=EXAMPLE.COM"
ipa: DEBUG: handshake complete, peer = 192.0.2.2:9444
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 replica.example.com. does not match the reverse
address replica.example.org
```

What this means: Multiple host names are used for a single PTR record. The DNS standard allows such configuration, but it causes an IdM replica installation to fail.

To fix the problem: Verify the DNS configuration, as described in [Section 2.1.3, “Verifying the Forward and Reverse DNS Configuration”](#).

A.2.3. Serial Numbers Not Found Errors

An error stating that a certificate serial number was not found appears on a replicated server:

```
Certificate operation cannot be completed: EXCEPTION (Certificate serial
number 0x2d not found)
```

What this means: A certificate replication agreement between two replicas has been removed but a data replication agreement is still in place. Both replicas are still issuing certificates, but information about the certificates is no longer replicated.

Example situation:

1. Replica A issues a certificate to a host.
2. The certificate is not replicated to replica B, because the replicas have no certificate replication agreement established.
3. A user attempts to use replica B to manage the host.
4. Replica B returns an error that it cannot verify the host's certificate serial number. This is because replica B has information about the host in its data directory, but it does not have the host certificate in its certificate directory.

To fix the problem:

1. Enable certificate server replication between the two replicas using the **ipa-csreplica-manage connect** command. See [Section 32.3, “Creating and Removing Replication Agreements”](#).
2. Re-initialize one of the replicas from the other to synchronize them. See [Section 32.5, “Re-initializing a Replica”](#).



Warning

Re-initializing overwrites data on the re-initialized replica with the data from the other replica. Some information might be lost.

A.2.4. Cleaning Replica Update Vector (RUV) Errors

After a replica has been removed from the IdM topology, obsolete RUV records are now present on one or more remaining replicas.

Possible causes:

- » The replica has been removed without properly removing its replication agreements first, as described in [Section 32.3, “Removing Replication Agreements”](#).
- » The replica has been removed when another replica was offline.

What this means: The other replicas still expect to receive updates from the removed replica.

Note

The correct procedure for removing a replica is described in [Section 32.6, “Removing a Replica”](#).

To fix the problem: Clean the RUV records on the replica that expects to receive the updates.

- List the details about the obsolete RUVs using the **ipa-replica-manage list-ruv** command. The command displays the replica IDs:

```
# ipa-replica-manage list-ruv
server1.example.com:389: 6
server2.example.com:389: 5
server3.example.com:389: 4
server4.example.com:389: 12
```

- Clear the corrupt RUVs using the **ipa-replica-manage clean-ruv *replica_ID*** command. The command removes any RUVs associated with the specified replica.

Repeat the command for every replica with obsolete RUVs. For example:

```
# ipa-replica-manage clean-ruv 6
# ipa-replica-manage clean-ruv 5
# ipa-replica-manage clean-ruv 4
# ipa-replica-manage clean-ruv 12
```



Warning

Proceed with extreme caution when using **ipa-replica-manage clean-ruv**. Running the command against a valid replica ID will corrupt all the data associated with that replica in the replication database.

If this happens, re-initialize the replica from another replica as described in [Section 32.5, “Re-initializing a Replica”](#).

- Run **ipa-replica-manage list-ruv** again.

- If the command no longer displays any corrupt RUVs, the records have been successfully cleaned.
- If the command still displays corrupt RUVs, clear them manually using this task:

```
dn: cn=clean replica_ID, cn=cleanallruv, cn=tasks, cn=config
objectclass: extensibleObject
replica-base-dn: dc=example,dc=com
replica-id: replica_ID
replica-force-cleaning: no
cn: clean replica_ID
```

If you are not sure on which replica to clean the RUVs:

- Search all your servers for active replica IDs. Make a list of uncorrupted and reliable replica IDs.

To find the IDs of valid replicas, run this LDAP query for all the nodes in your topology:

```
# ldapsearch -p 389 -h IdM_node -D "cn=directory manager" -W -b
"cn=config" "(objectclass=nsds5replica)" nsDS5ReplicaId
```

2. Run `ipa-replica-manage list-ruv` on every server. Note any replica IDs that are not on the list of uncorrupted replica IDs.
3. Run `ipa-replica-manage clean-ruv replica_ID` for every corrupted replica ID.

A.2.5. Recovering a Lost CA Server

You only had one server with CA installed. This server failed and is now lost.

What this means: The CA configuration for your IdM domain is no longer available.

To fix the problem:

If you have a backup of the original CA server available, you can restore the server and install the CA on a replica.

1. Recover the CA server from backup. See [Section 7.2, “Restoring a Backup”](#) for details.
This makes the CA server available to the replica.
2. Delete the replication agreements between the initial server and the replica to avoid replication conflicts. See [Section 32.3, “Creating and Removing Replication Agreements”](#).
3. Install the CA on the replica. See [Section 30.8, “Promoting a Replica to a Master CA Server”](#).
4. Decommission the original CA server. See [Section 32.6, “Removing a Replica”](#).

If you do not have a backup of the original CA server, the CA configuration was lost when the server failed and cannot be recovered.

A.3. Identity Management Clients

A.3.1. The Client Is Unable to Resolve Reverse Lookups when Using an External DNS

An external DNS server returns a wrong host name for the IdM server. The following errors related to the IdM server appear in the Kerberos database:

```
Jun 30 11:11:48 server1 krb5kdc[1279](info): AS_REQ (4 etypes {18 17 16 23}) 192.0.2.1: 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}) 192.0.2.1: 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}) 192.0.2.1: UNKNOWN_SERVER: authtime 0, admin EXAMPLE COM for HTTP/server1.wrong.example.com@EXAMPLE.COM, Server not found in Kerberos database
```

What this means: The external DNS name server returns the wrong host name for the IdM server or returns no answer at all.

To fix the problem:

1. Verify your DNS configuration, and make sure the DNS domains used by IdM are properly delegated. See [Section 2.1.3, “Host Name and DNS Configuration”](#) for details.
2. Verify your reverse (PTR) DNS records settings. See [Chapter 20, Managing DNS](#) for details.

A.3.2. The Client Is Not Added to the DNS Zone

When running the `ipa-client-install` utility, the `nsupdate` utility fails to add the client to the DNS zone.

What this means: The DNS configuration is incorrect.

To fix the problem:

1. Verify your configuration for DNS delegation from the parent zone to IdM. See [Section 2.1.3, “Host Name and DNS Configuration”](#) for details.
2. Make sure that dynamic updates are allowed in the IdM zone. See [Section 20.6.1, “Enabling Dynamic DNS Updates”](#) for details.

For details on managing DNS in IdM, see [Section 20.8, “Managing Reverse DNS Zones”](#). For details on managing DNS in Red Hat Enterprise Linux, see [Section 11.2.3, “Editing Zone Files”](#) in the *Networking Guide*.

A.3.3. Client Connection Problems

Users cannot log in to a machine. Attempts to access user and group information, such as with the `getent passwd admin` command, fail.

What this means: Client authentication problems often indicate problems with the System Security Services Daemon (SSSD) service.

To fix the problem: Examine the SSSD logs in the `/var/log/sssd/` directory. The directory includes a log file for the DNS domain, such as `sssd_example.com.log`.

If the logs do not include enough information, increase the log level:

1. In the `/etc/sssd/sssd.conf` file, look up the `[domain/example.com]` section. Adjust the `debug_level` option to record more information in the logs.

```
debug_level = 9
```

2. Restart the `sssd` service.

```
# systemctl start sssd
```

3. Examine `sssd_example.com.log` again. The file now includes more error messages.

A.4. Logging In and Authentication Problems

A.4.1. Kerberos GSS Failures When Running ipa Commands

Immediately after installing a server, Kerberos errors occur when attempting 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)
```

What this means: DNS is not properly configured.

To fix the problem: Verify your DNS configuration.

- ▶ See [Section 2.1.3, “Host Name and DNS Configuration”](#) for DNS requirements for IdM servers.
- ▶ See [Section 5.2.2, “DNS and Realm Settings”](#) in the *Windows Integration Guide* for DNS requirements for Active Directory trust.

A.4.2. Kerberos Authentication Not Working in the UI

Turn on verbose logging for the authentication process to help diagnose the problem. For advice on how to do this in Firefox, see [Section “Firefox Configuration for Kerberos Troubleshooting”](#) in the *System-Level Authentication Guide*.

A.4.3. SSH Connection Fails when Using GSS-API

Users are unable to log in to IdM machines using SSH.

What this means: When SSH attempts to connect to an IdM resource using GSS-API as the security method, GSS-API first verifies the DNS records. SSH failures are often caused by incorrect reverse DNS entries. The incorrect records prevent SSH from locating the IdM resource.

To fix the problem: Verify your DNS configuration as described in [Section 2.1.3, “Host Name and DNS Configuration”](#).

As a temporary workaround, you can also disable reverse DNS lookups in the SSH configuration. To do this, set the **GSSAPITrustDNS** to **no** in the `/etc/ssh/ssh_config` file. Instead of using reverse DNS records, SSH will pass the given user name directly to GSS-API.

A.4.4. OTP Token Out of Sync

Authentication using OTP fails because the token is desynchronized.

To fix the problem: Resynchronize the token. Any user can resynchronize their tokens regardless of the token type and whether or not the user has permission to modify the token settings.

1. In the IdM web UI: Click **Sync OTP Token** on the login page.

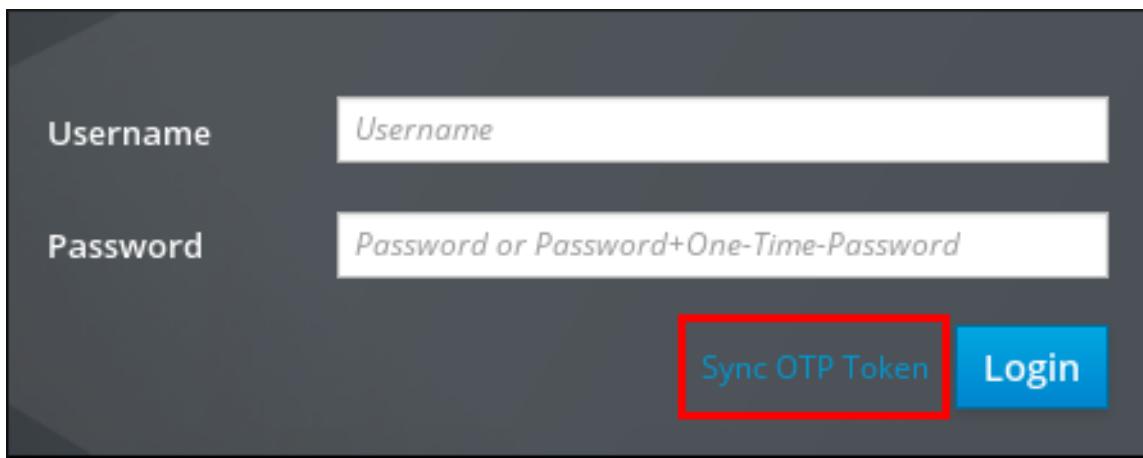
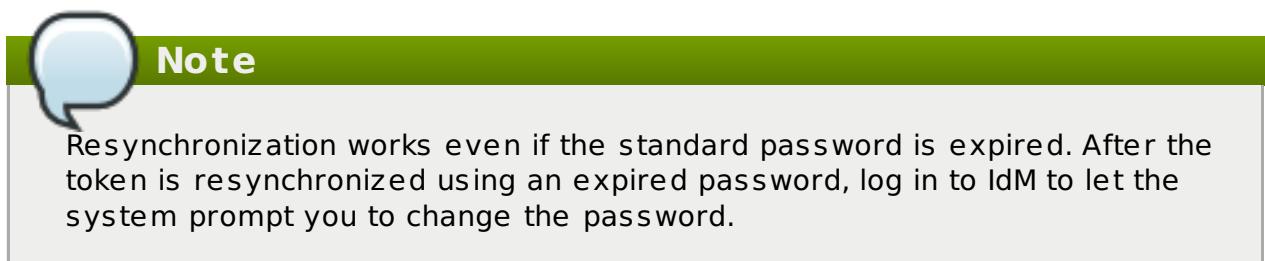


Figure A.1. Sync OTP Token

From the command line: Run the `ipa otptoken-sync` command.

2. Provide the information required to resynchronize the token. For example, IdM will ask you to provide your standard password and two subsequent token codes generated by the token.



Appendix B. Revision History

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Enterprise Linux.

Revision 7.0-22	Fri Jul 29 2016	Aneta Petrová
Async update: added a chapter on using vaults.		
Revision 7.0-21	Thu Jul 28 2016	Marc Muehlfeld
Async update: updated introduction, other minor fixes.		
Revision 7.0-19	Tue Jun 28 2016	Aneta Petrová
Updated diagrams. Added a section on benefits of using IdM to the intro chapter. Other minor fixes and tweaks.		
Revision 7.0-18	Fri Jun 10 2016	Aneta Petrová
Updated introduction, server installation, and troubleshooting chapters. Added a chapter for user, host, and service certificates. Merged changing domain DNS config chapter into other chapters. Other minor fixes.		
Revision 7.0-17	Fri May 27 2016	Aneta Petrová
Added a diagram for user lifecycle.		
Revision 7.0-16	Thu Mar 24 2016	Aneta Petrová
Added user lifecycle. Updated the User Accounts, User Authentication, and Managing Replicas chapters.		
Revision 7.0-15	Thu Mar 03 2016	Aneta Petrová
Updated several DNS sections. Moved restricting domains for PAM services to the System-Level Authentication Guide.		
Revision 7.0-14	Tue Feb 09 2016	Aneta Petrová
Added smart cards, ID views, and OTP. Updated some web UI screenshots and the Basics of Management and Restricting Domains chapters. Moved uninstallation procedures into installation chapters. Commented out index. Other minor updates.		
Revision 7.0-13	Thu Nov 19 2015	Aneta Petrová
Minor updates to certificate profile management and promoting a replica to master.		
Revision 7.0-12	Fri Nov 13 2015	Aneta Petrová
Version for 7.2 GA release with updates to DNS and other sections.		
Revision 7.0-11	Thu Nov 12 2015	Aneta Petrová
Version for 7.2 GA release.		
Revision 7.0-10	Fri Mar 13 2015	Tomáš Čapek
Async update with last-minute edits for 7.1.		
Revision 7.0-8	Wed Feb 25 2015	Tomáš Čapek
Version for 7.1 GA release.		
Revision 7.0-6	Fri Dec 05 2014	Tomáš Čapek

Rebuild to update the sort order on the splash page.

Revision 7.0-4

Wed Jun 11 2014

Ella Deon Ballard

Initial release.