Fedora 17

FreeIPA: Identity/Policy Management

Managing Identity and Authorization Policies for Linux-Based Infrastructures

Ella Deon Lackey
Identity and policy management — for both users and machines — is a core function for almost any enterprise environment. IPA provides a way to create an identity domain that allows machines to enroll to a domain and immediately access identity information required for single sign-on and authentication services, as well as policy settings that govern authorization and access. This manual covers all aspects of installing, configuring, and managing IPA domains, including both servers and clients. This guide is intended for IT and systems administrators.
5.4.1. From the Web UI ................................................................. 105
5.4.2. From the Command Line .................................................. 107
5.5. Unlocking User Accounts After Password Failures .................. 108
5.6. Managing User Private Groups ............................................. 108
  5.6.1. Disabling Private Groups for a Specific User ......................... 108
  5.6.2. Disabling Private Groups Globally ..................................... 108
5.7. Managing Unique UID and GID Number Assignments ............ 109
  5.7.1. About ID Range Assignments During Installation ................ 109
  5.7.2. Adding New Ranges ......................................................... 110
5.8. Managing User and Group Schema ....................................... 110
  5.8.2. Applying Custom Object Classes to New User Entries .......... 112
  5.8.3. Applying Custom Object Classes to New Group Entries .......... 115
5.9. Managing User Groups ......................................................... 117
  5.9.1. Creating User Groups ...................................................... 118
  5.9.2. Adding Group Members .................................................. 121
  5.9.3. Deleting User Groups ..................................................... 128
5.10. Searching for Users and Groups ......................................... 130
  5.10.1. With the UI ................................................................. 130
  5.10.2. With the Command Line ................................................ 130
5.11. Specifying Default User and Group Settings ......................... 132
  5.11.1. Viewing Settings from the Web UI ................................. 133
  5.11.2. Viewing Settings from the Command Line ......................... 134

6. Identity: Managing Hosts and Services ...................................... 137
  6.1. About Hosts, Services, and Machine Identity and Authentication 137
  6.2. Adding Host Entries .......................................................... 138
    6.2.1. Adding Host Entries from the Web UI ............................ 138
    6.2.2. Adding Host Entries from the Command Line ................. 141
  6.3. Enrolling Clients Manually ................................................ 142
    6.3.1. Performing a Split Enrollment ...................................... 142
  6.4. Manually Unconfiguring Client Machines ............................. 143
  6.5. Managing Services ........................................................... 144
    6.5.1. Adding and Editing Service Entries and Keytabs ............... 144
    6.5.2. Adding Services and Certificates for Services ................. 148
    6.5.3. Storing Certificates in NSS Databases ............................ 152
    6.5.4. Configuring Clustered Services ...................................... 152
    6.5.5. Using the Same Service Principal for Multiple Services .......... 153
  6.6. Disabling and Re-enabling Host and Service Entries ............... 153
    6.6.1. Disabling Host and Service Entries ................................ 154
    6.6.2. Re-enabling Hosts and Services .................................... 154
  6.7. Extending Access Permissions over Other Hosts and Services ...... 154
    6.7.1. Delegating Service Management ................................... 155
    6.7.2. Delegating Host Management ....................................... 156
    6.7.3. Delegating Host or Service Management in the Web UI ......... 156
    6.7.4. Accessing Delegated Services ...................................... 158
  6.8. Managing Public SSH Keys for Hosts ................................... 159
    6.8.1. About FreeIPA Clients and OpenSSH ............................ 159
    6.8.2. Adding Host Keys ...................................................... 160
    6.8.3. Removing Host Keys ................................................... 160
  6.9. Renaming Machines and Reconfiguring FreeIPA Client Configuration 161
  6.10. Managing Host Groups ..................................................... 162
    6.10.1. Creating Host Groups ............................................... 162
    6.10.2. Adding Group Members ............................................. 164
11. Policy: Defining Password Policies ................................................................. 251
   11.1. About Password Policies and Policy Attributes ........................................ 251
   11.2. Viewing Password Policies ..................................................................... 253
       11.2.1. Viewing the Global Password Policy ................................................ 253
       11.2.2. Viewing Group-Level Password Policies ......................................... 256
       11.2.3. Viewing the Password Policy in Effect for a User ........................... 258
   11.3. Creating and Editing Password Policies .................................................. 258
       11.3.1. Creating Password Policies in the Web UI ...................................... 258
       11.3.2. Creating Password Policies with the Command Line ..................... 261
       11.3.3. Editing Password Policies with the Command Line ......................... 261
   11.4. Managing Password Expirations .............................................................. 261
   11.5. Changing the Priority of Group Password Policies ................................... 262
   11.6. Setting Account Lockout Policies ............................................................ 263
       11.6.1. In the UI .................................................................................. 263
       11.6.2. In the CLI ............................................................................... 265
   11.7. Enabling a Password Change Dialog ...................................................... 266

12. Policy: Managing the Kerberos Domain ......................................................... 267
   12.1. About Kerberos .................................................................................... 267
       12.1.1. About Principal Names ................................................................. 267
       12.1.2. About Protecting Keytabs ............................................................ 268
   12.2. Setting Kerberos Ticket Policies .............................................................. 268
       12.2.1. Setting Global Ticket Policies ....................................................... 268
       12.2.2. Setting User-Level Ticket Policies ................................................ 270
   12.3. Refreshing Kerberos Tickets .................................................................... 270
   12.4. Caching Kerberos Passwords ................................................................. 272
   12.5. Removing Keytabs .............................................................................. 272

    10.1. About Automount and FreeIPA .............................................................. 231
    10.2. Configuring Automount ......................................................................... 232
       10.2.1. Configuring autos on Fedora ......................................................... 232
       10.2.2. Configuring Automount on Solaris ............................................... 234
    10.3. Setting up a Kerberized NFS Server ...................................................... 235
       10.3.1. Setting up a Kerberized NFS Server ............................................. 235
       10.3.2. Setting up a Kerberized NFS Client ............................................. 237
    10.4. Configuring Kerberized CIFS ................................................................ 239
       10.4.1. Setting up Samba Groups in FreeIPA .......................................... 239
       10.4.2. Configuring the CIFS Client ......................................................... 241
    10.5. Configuring Locations ......................................................................... 241
       10.5.1. Configuring Locations through the Web UI ................................. 242
       10.5.2. Configuring Locations through the Command Line ...................... 243
    10.6. Configuring Maps ............................................................................... 243
       10.6.1. Configuring Direct Maps ............................................................... 244
       10.6.2. Configuring Indirect Maps ............................................................ 247
       10.6.3. Importing Automount Maps .......................................................... 250

    9.1. Configuring DNS .................................................................................. 227
       9.1.1. Defining DNS Queries ................................................................. 228
       9.1.2. Resolving Hostnames in the FreeIPA Domain .................................. 230
       9.1.3. Setting DNS Access Policies ........................................................ 230
       9.1.4. Synchronizing Forward and Reverse Zone Entries .......................... 230
       9.1.5. Defining DNS Queries .................................................................. 228
       9.1.6. Resolving Hostnames in the FreeIPA Domain .............................. 229
       9.1.7. Setting DNS Access Policies ........................................................ 229
       9.1.8. Synchronizing Forward and Reverse Zone Entries .......................... 229
       9.1.9. Defining DNS Queries .................................................................. 229
       9.1.10. Resolving Hostnames in the FreeIPA Domain .............................. 229
       9.1.11. Setting DNS Access Policies ........................................................ 229
       9.1.12. Synchronizing Forward and Reverse Zone Entries .......................... 230

8. Setting Replication Policies ....................................................................... 213
    8.1. Setting Global Replication Policies ....................................................... 213
    8.2. Setting Local Replication Policies .......................................................... 216
       8.2.1. Setting up Local Replication Policies ............................................ 216
       8.2.2. Setting up Local Replication Policies ............................................ 217
       8.2.3. Setting up Local Replication Policies ............................................ 218
       8.2.4. Setting up Local Replication Policies ............................................ 219

7. FreeIPA Policies ....................................................................................... 202
    7.1. About FreeIPA Policies and Policy Attributes ...................................... 202
    7.2. Viewing Policies ................................................................................... 203
       7.2.1. Viewing Global Policies ............................................................... 203
       7.2.2. Viewing Group-Level Policies ...................................................... 206
       7.2.3. Viewing the Policy in Effect for a User ........................................ 208
    7.3. Creating and Editing Policies ................................................................ 210
       7.3.1. Creating Global Policies .............................................................. 210
       7.3.2. Creating Group-Level Policies ..................................................... 213
       7.3.3. Creating the Policy in Effect for a User ........................................ 215
    7.4. Managing Policy Expirations ................................................................. 217
    7.5. Changing Policy Priorities ..................................................................... 219
    7.6. Setting Account Lockout Policies ........................................................... 221
       7.6.1. In the UI .................................................................................. 221
       7.6.2. In the CLI ............................................................................... 224
    7.7. Enabling a Policy Change Dialog ........................................................... 225

6. Setting DNS Access Policies .................................................................... 207
    6.1. About DNS Access Policies ................................................................ 207
    6.2. Creating and Editing DNS Access Policies .......................................... 209
       6.2.1. Creating Global DNS Access Policies ........................................ 209
       6.2.2. Creating Group-Level DNS Access Policies ............................... 212
       6.2.3. Creating the DNS Access Policy in Effect for a User ................. 214
    6.3. Managing DNS Access Expirations ....................................................... 216
    6.4. Changing DNS Access Priorities ........................................................... 218
    6.5. Setting Account Lockout Policies ........................................................... 220
       6.5.1. In the UI .................................................................................. 220
       6.5.2. In the CLI ............................................................................... 223
    6.6. Enabling a DNS Access Change Dialog ................................................ 224

5. Configuring DNS .................................................................................... 198
    5.1. About DNS ....................................................................................... 198
    5.2. Creating and Editing DNS Zones .......................................................... 200
       5.2.1. Creating Global DNS Zones ........................................................ 200
       5.2.2. Creating Group-Level DNS Zones .............................................. 203
       5.2.3. Creating the DNS Zone in Effect for a User ............................... 205
    5.3. Managing DNS Zone Expirations .......................................................... 207
    5.4. Changing DNS Zone Priorities ............................................................... 209
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13. Policy: Using sudo</td>
<td>275</td>
</tr>
<tr>
<td>13.1. About sudo and IPA</td>
<td>275</td>
</tr>
<tr>
<td>13.1.1. General sudo Configuration in FreeIPA</td>
<td>275</td>
</tr>
<tr>
<td>13.1.2. sudo and Netgroups</td>
<td>275</td>
</tr>
<tr>
<td>13.1.3. Supported sudo Clients</td>
<td>276</td>
</tr>
<tr>
<td>13.2. Setting up sudo Commands and Command Groups</td>
<td>276</td>
</tr>
<tr>
<td>13.2.1. Adding sudo Commands</td>
<td>276</td>
</tr>
<tr>
<td>13.2.2. Adding sudo Command Groups</td>
<td>277</td>
</tr>
<tr>
<td>13.3. Defining sudo Rules</td>
<td>281</td>
</tr>
<tr>
<td>13.3.1. About External Users and Hosts</td>
<td>281</td>
</tr>
<tr>
<td>13.3.2. About sudo Options Format</td>
<td>282</td>
</tr>
<tr>
<td>13.3.3. Defining sudo Rules in the Web UI</td>
<td>283</td>
</tr>
<tr>
<td>13.3.4. Defining sudo Rules in the Command Line</td>
<td>291</td>
</tr>
<tr>
<td>13.4. An Example of Configuring sudo</td>
<td>294</td>
</tr>
<tr>
<td>13.4.1. Server Configuration for sudo Rules</td>
<td>294</td>
</tr>
<tr>
<td>13.4.2. Client Configuration for sudo Rules</td>
<td>295</td>
</tr>
<tr>
<td>14. Policy: Configuring Host-Based Access Control</td>
<td>297</td>
</tr>
<tr>
<td>14.1. About Host-Based Access Control</td>
<td>297</td>
</tr>
<tr>
<td>14.2. Creating Host-Based Access Control Entries for Services and Service Groups</td>
<td>298</td>
</tr>
<tr>
<td>14.2.1. Adding HBAC Services</td>
<td>299</td>
</tr>
<tr>
<td>14.2.2. Adding Service Groups</td>
<td>300</td>
</tr>
<tr>
<td>14.3. Defining Host-Based Access Control Rules</td>
<td>304</td>
</tr>
<tr>
<td>14.3.1. Setting Host-Based Access Control Rules in the Web UI</td>
<td>304</td>
</tr>
<tr>
<td>14.3.2. Setting Host-Based Access Control Rules in the Command Line</td>
<td>311</td>
</tr>
<tr>
<td>14.4. Testing Host-Based Access Control Rules</td>
<td>314</td>
</tr>
<tr>
<td>14.4.1. The Limits of Host-Based Access Control Configuration</td>
<td>314</td>
</tr>
<tr>
<td>14.4.2. Test Scenarios for Host-Based Access Control (CLI-Based)</td>
<td>315</td>
</tr>
<tr>
<td>14.4.3. Testing Host-Based Access Control Rules in the UI</td>
<td>316</td>
</tr>
<tr>
<td>15. Policy: Defining SELinux User Maps</td>
<td>319</td>
</tr>
<tr>
<td>15.1. About FreeIPA, SELinux, and Mapping Users</td>
<td>319</td>
</tr>
<tr>
<td>15.2. Configuring SELinux Users in FreeIPA</td>
<td>322</td>
</tr>
<tr>
<td>15.2.1. In the Web UI</td>
<td>322</td>
</tr>
<tr>
<td>15.2.2. In the CLI</td>
<td>323</td>
</tr>
<tr>
<td>15.3. Mapping SELinux Users and FreeIPA Users</td>
<td>324</td>
</tr>
<tr>
<td>15.3.1. In the Web UI</td>
<td>325</td>
</tr>
<tr>
<td>15.3.2. In the CLI</td>
<td>328</td>
</tr>
<tr>
<td>15.4. Troubleshooting SELinux Login Problems</td>
<td>330</td>
</tr>
<tr>
<td>16. Policy: Defining Automatic Group Membership for Users and Hosts</td>
<td>331</td>
</tr>
<tr>
<td>16.1. About Automembership</td>
<td>331</td>
</tr>
<tr>
<td>16.2. Defining Automembership Rules (Basic Procedure)</td>
<td>332</td>
</tr>
<tr>
<td>16.2.1. From the Web UI</td>
<td>332</td>
</tr>
<tr>
<td>16.2.2. From the CLI</td>
<td>335</td>
</tr>
<tr>
<td>16.3. Examples of Using Automember Groups</td>
<td>336</td>
</tr>
<tr>
<td>16.3.1. Setting an All Users/Hosts Rule</td>
<td>337</td>
</tr>
<tr>
<td>16.3.2. Defining Default Automembership Groups</td>
<td>337</td>
</tr>
<tr>
<td>16.3.3. Using Automembership Groups with Windows Users</td>
<td>338</td>
</tr>
<tr>
<td>17. Configuration: Defining Access Control within FreeIPA</td>
<td>339</td>
</tr>
<tr>
<td>17.1. About Access Controls for FreeIPA Entries</td>
<td>339</td>
</tr>
<tr>
<td>17.1.1. A Brief Look at Access Control Concepts</td>
<td>339</td>
</tr>
<tr>
<td>17.1.2. Access Control Methods in FreeIPA</td>
<td>340</td>
</tr>
</tbody>
</table>
FreeIPA: Identity/Policy Management

B. Working with certmonger 399
   B.1. Requesting a Certificate with certmonger .......................................................... 399
   B.2. Storing Certificates in NSS Databases .................................................................. 400
   B.3. Tracking Certificates with certmonger .............................................................. 400

C. FreeIPA Tools Reference 401
   C.1. Using Special Characters with FreeIPA Tools ..................................................... 401
   C.2. ipa ....................................................................................................................... 401
      C.2.1. Location ...................................................................................................... 401
      C.2.2. Syntax ......................................................................................................... 401
      C.2.3. Help Topics ................................................................................................. 403
      C.2.4. Global Options ............................................................................................ 404
      C.2.5. Managing Attributes with --setattr, --addattr, and --delattr ..................... 405
      C.2.6. Return Codes .............................................................................................. 406
      C.2.7. Commands .................................................................................................. 406
   C.3. ipa DNS Commands .......................................................................................... 407
      C.3.1. ipa dnszone-add ......................................................................................... 407
      C.3.2. ipa dnsrecord-add ..................................................................................... 408
   C.4. ipa Host Commands ......................................................................................... 415
      C.4.1. ipa host-add ............................................................................................... 415
   C.5. Server Scripts .................................................................................................... 416
      C.5.1. A Quick Summary of Configuration Scripts ................................................. 416
      C.5.2. ipa-replica-install ..................................................................................... 418
      C.5.3. ipa-replica-prepare .................................................................................... 420
      C.5.4. ipa-server-install ....................................................................................... 421
   C.6. Client Scripts ..................................................................................................... 426
      C.6.1. ipa-client-install ........................................................................................ 426

Glossary 431

Index 445
Preface

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

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

1. Audience and Purpose

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

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

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

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

2. Examples and Formatting

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

2.1. Brackets

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

2.2. Client Tool Information

The tools for FreeIPA are located in the /usr/bin and the /usr/sbin directories.
Preface

The LDAP tools used to edit the FreeIPA directory services, such as `ldapmodify` and `ldapsearch`, are from OpenLDAP. OpenLDAP tools use SASL connections by default. To perform a simple bind using a username and password, use the `-x` argument to disable SASL.

2.3. Text Formatting and Styles

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

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

Other formatting styles draw attention to important text.

**NOTE**

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

**IMPORTANT**

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

**WARNING**

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

3. Giving Feedback

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

1. Select the Community group and the freeIPA product.
2. Set the component to Documentation.
3. Set the version number to 2.2.
4. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.
   For enhancements, put in what information needs to be added and why.
5. Give a clear title for the bug. For example, "Incorrect command example for setup script options" is better than "Bad example".

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact the Fedora docs team at docs@lists.fedoraproject.org.

4. Document Change History

Revision: July 22, 2011  Ella Deon Lackey d1ackey@redhat.com
2.1.0-2
Completing first round of content revisions on the chapters for server installation, client installation, DNS, basic usage, managing clients, and the preface.
Beginning content revisions on the chapters for users, Kerberos, automount, and managing servers.
Bare initial draft of a tools appendix.
Bugzilla work: 646226, 646240, 646257, 646267, 68173, 693843, 701465, 709385, 714603, 715015
FreeIPA.org tickets: 1183, 1359, 1449, 1058, 1335, 1107, 1355, 1430, 803, 991, 615, 969, 594, 593
freeipa-guide trac tickets: 18, 19, 16, 17

Revision: May 10, 2011  Ella Deon Lackey d1ackey@redhat.com
2.1.0-1
Beginning draft for the Fedora docs project.
Introduction to FreeIPA

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

FreeIPA provides a unifying skin for standards-defined, common network services, including PAM, LDAP, Kerberos, DNS, NTP, and certificate services, and it allows Fedora systems to serve as the domain controllers.

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

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

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

1.1.1. A Working Definition for FreeIPA

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

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

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

FreeIPA does three things:

• Create a Linux-based and Linux-controlled domain. Both FreeIPA servers and FreeIPA clients are Linux or Unix machines. While FreeIPA can synchronize data with an Active Directory domain to allow integration with Windows servers, it is not an administrative tools for Windows machines and it does not support Windows clients. FreeIPA is a management tool for Linux domains.

• Centralize identity management and identity policies.

• Build on existing, native Linux applications and protocols. While FreeIPA has its own processes and configuration, its underlying technologies are familiar and trusted by Linux administrators and are well established on Linux systems.
In a sense, FreeIPA isn’t making administrators do something new; it is helping them do it better. There are a few ways to illustrate that.

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

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

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

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

1.1.2. Contrasting FreeIPA with a Standard LDAP Directory

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

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

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

FreeIPA, on the other hand, has a very specific purpose and fits a very specific application. It is not a general LDAP directory, it is not a backend, and it is not a general policy server. It is not generic.
FreeIPA focuses on identities (user and machine) and policies that relate to those identities and their interactions. While it uses an LDAP backend to store its data, FreeIPA has a highly-customized and specific set of schema that defines a particular set of identity-related entries and defines them in detail. It has a relatively flat and simple directory tree because it has only a handful of entry types and relationships that are relevant to its purpose. It has rules and limitations on how the FreeIPA server can be deployed because it can only be deployed for a specific purpose: managing identities.

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

### Table 1.1. FreeIPA Compared to 389 Directory Server

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

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

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

### 1.2. Bringing Linux Services Together

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

A FreeIPA server is, at its core, an identity and authentication server. The primary FreeIPA server, essentially a domain controller, uses a Kerberos server and KDC for authentication. An LDAP backend contains all of the domain information, including users, client machines, and domain configuration.
Figure 1.1. The FreeIPA Server: Unifying Services

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

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

1.2.1. Authentication: Kerberos KDC

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

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

In FreeIPA, the Kerberos administration server is set up on the FreeIPA domain controller, and all of the Kerberos data are stored in FreeIPA's backend Directory Server. The Directory Server instance defines and enforces access controls for the Kerberos data.
1.2.2. Data Storage: 389 Directory Server

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

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

1.2.3. Authentication: Dogtag Certificate System

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

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

1.2.4. Server/Client Discovery: DNS

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

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

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

1.2.5. Management: NTP

Many services require that servers and clients have the same system time, within a certain variance. For example, Kerberos tickets use time stamps to determine their validity. If the times between the server and client skew outside the allowed range, then any Kerberos tickets are invalidated.
Chapter 1. Introduction to FreeIPA

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

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

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

1.3. Relationships Between Servers and Clients

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

1.3.1. About FreeIPA Servers and Replicas

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

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

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

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

**NOTE**

The only real different between a FreeIPA server and a FreeIPA replica is that a server is a new installation and a replica is based on an existing server. Once the instance is configured, servers and replicas are basically identical in functionality and behavior within the FreeIPA domain.

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

Servers and replicas both use underlying LDAP directories to store user and host entries, configuration data, policy configuration, and keytabs, certificates, and keys. Servers and replicas propagate data among each other through multi-master replication agreements. Both servers and replicas are masters in the replication topology.
The replication topology essentially creates a cloud of FreeIPA servers. One benefit of a server domain is automatic load balancing, using the SRV records in DNS. The SRV record priority sets the order that servers and replicas are contacted, while weight distributed the load between servers/replicas with the same priority. The server and replica DNS entries can be edited to change the load balancing, which is covered in Example 9.4, “SRV Record” and Section 9.15, “Changing Load Balancing for FreeIPA Servers and Replicas”.

1.3.2. About FreeIPA Clients

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

For Fedora systems, a certain number of platform tools are available for FreeIPA to use, such as SSSD. These are FreeIPA-enabled platform applications, which is simply a way of saying that they are aspects of the underlying platform that work with FreeIPA services. Other tools, like certain PAM and NSS modules and FreeIPA command-line utilities, are provided as FreeIPA-specific packages that must be installed on the machine. These are FreeIPA-related components.
FreeIPA uses the local storage (cache) on a client to improve performance in a few ways:

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

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

- Identity information (about users, machines, and groups) is stored in the LDB database, which uses the same syntax as an LDAP directory. This identity information is originally stored in the FreeIPA server's 389 Directory Server instance. Because this information changes frequently and is referenced frequently, it is important to be able to call the more current information quickly, which is possible using an LDB database on the client and the Directory Server on the server.
- Policy information is more static than identity information, and it can include configuration for SELinux or sudo. These policies are set globally on the server and then are propagated to the clients. On the client, the policy information is stored in the filesystem in XML files which can be downloaded and converted into a native file for whatever service is being managed.

A specific set of services on the FreeIPA server interact with a subset of services and modules on the FreeIPA client. A client is any machine (a host) which can retrieve a keytab or certificates from the FreeIPA domain.
Figure 1.4. Interactions Between FreeIPA Services
Chapter 1. Introduction to FreeIPA

Figure 1.4, "Interactions Between FreeIPA Services" shows that Fedora uses two native daemons to interact with the FreeIPA server:

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

When a Fedora client is added to the domain (enrolled), its SSSD and `certmonger` are configured to connect to the FreeIPA server and the required Kerberos keytab and host certificates are created. (The host certificate is not used directly by FreeIPA; it may be used by other services, such as a web server.)
Chapter 2.

Installing a FreeIPA Server

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

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

2.1. Preparing to Install the FreeIPA Server

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

2.1.1. Hardware Recommendations

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

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


2.1.2. Software Requirements

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

- Kerberos 1.9
- The named and bind-dyndb-ldap packages for DNS

2.1.3. Supported Web Browsers

The only supported browser to access the FreeIPA web UI is Firefox 3.x or 4.x.
Chapter 2. Installing a FreeIPA Server

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

2.1.4.1. Hostname and IP Address Requirements
Regardless of whether the DNS is within the FreeIPA server or external, the server host must have DNS properly configured:

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

  IMPORTANT

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

• The hostname must be all lower-case.

• The server’s A record must be set and resolve to its public IP address.

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

• The server's hostname and IP address must be in its own /etc/hosts file.

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

  TIP

  If the FreeIPA server is configured to host its own DNS server, any previous existing DNS ignored. A records and PTR records do not need to match for the FreeIPA server machine, and the machine can have any configured IP address.

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

2.1.4.3. System Files
The server script overwrites system files to set up the FreeIPA domain. The system should be clean, without custom configuration for services like DNS and Kerberos, before configuring the FreeIPA server.
2.1.4.4. System Ports
FreIIPA uses a number of ports to communicate with its services. These ports, listed in Table 2.1, “FreeIPA Ports”, must be open and available for FreIIPA to work. They cannot be in use by another service or blocked by a firewall. To make sure that these ports are available, try `iptables` to list the available ports or `nc`, `telnet`, or `nmap` to connect to a port or run a port scan.

To open a port:

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

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

<table>
<thead>
<tr>
<th>Service</th>
<th>Ports</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/HTTPS</td>
<td>80, 443</td>
<td>TCP</td>
</tr>
<tr>
<td>LDAP/LDAPS</td>
<td>389, 636</td>
<td>TCP</td>
</tr>
<tr>
<td>Kerberos</td>
<td>88, 464</td>
<td>TCP and UDP</td>
</tr>
<tr>
<td>DNS</td>
<td>53</td>
<td>TCP and UDP</td>
</tr>
<tr>
<td>NTP</td>
<td>123</td>
<td>UDP</td>
</tr>
<tr>
<td>Dogtag Certificate System -</td>
<td>7389</td>
<td>TCP</td>
</tr>
<tr>
<td>LDAP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.1.4.5. NTP
If a server is being installed on a virtual machine, that server should not run an NTP server. To disable NTP for FreIIPA, use the `--no-ntp` option.

2.1.4.6. NSCD

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

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

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

```
positive-time-to-live   group           3600
negative-time-to-live   group           60
positive-time-to-live   hosts           3600
negative-time-to-live   hosts           20
```
2.1.5. Networking

2.1.5.1. Configuring Networking Services

The default networking service used by Fedora is NetworkManager, and due to the way this service works, it can cause problems with FreeIPA and the KDC. Consequently, it is highly recommended that you use the network service to manage the networking requirements in a FreeIPA environment and disable the NetworkManager service.

1. Boot the machine into single-user mode and run the following commands:

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

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

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

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

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

4. Ensure that static networking is correctly configured.

5. Restart the system.

2.1.5.2. Configuring the /etc/hosts File

You need to ensure that your /etc/hosts file is configured correctly. A misconfigured file can prevent the FreeIPA command-line tools from functioning correctly and can prevent the FreeIPA web interface from connecting to the FreeIPA server.

Configure the /etc/hosts file to list the FQDN for the FreeIPA server before any aliases. Also ensure that the hostname is not part of the localhost entry. The following is an example of a valid hosts file:

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

**Important**

Do not omit the IPv4 entry in the /etc/hosts file. This entry is required by the FreeIPA web service.

2.2. Installing the FreeIPA Server Packages

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

All of these packages can be installed using the yum command:
Creating a FreeIPA Server Instance

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

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

After the packages are installed, the server instance must be created using the `ipa-server-install` command. The options for configuring the new server instance are described in Section 2.3, “Creating a FreeIPA Server Instance”.

### 2.3. Creating a FreeIPA Server Instance

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

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

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

**NOTE**

The port numbers and directory locations used by FreeIPA are all defined automatically, as defined in Section 2.1.4.4, “System Ports” and . These ports and directories cannot be changed or customized.

### 2.3.1. About `ipa-server-install`

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

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

Table 2.2, “`ipa-server-install` Options” lists some common arguments with `ipa-server-install`, while Section 2.3.3, “Examples of Creating the FreeIPA Server” has examples of some common installation scenarios. The full list of options are in Section C.5.4, “`ipa-server-install`”. In real life, the
**ipa-server-install** options are versatile enough to be customized to the specific deployment environment.

### Table 2.2. ipa-server-install Options

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a <strong>ipa_admin_password</strong></td>
<td>The password for the FreeIPA administrator. This is used for the admin user to authenticate to the Kerberos realm.</td>
</tr>
<tr>
<td>--hostname=<strong>hostname</strong></td>
<td>The fully-qualified domain name of the FreeIPA server machine.</td>
</tr>
<tr>
<td><strong>IMPORTANT</strong></td>
<td>This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures. Additionaly, the hostname must all be lower-case. No capital letters are allowed.</td>
</tr>
<tr>
<td>-n <strong>domain_name</strong></td>
<td>The name of the LDAP server domain to use for the FreeIPA domain. This is usually based on the FreeIPA server's hostname.</td>
</tr>
<tr>
<td>-p <strong>directory_manager_password</strong></td>
<td>The password for the superuser, <code>cn=Directory Manager</code>, for the LDAP service.</td>
</tr>
<tr>
<td>-r <strong>realm_name</strong></td>
<td>The name of the Kerberos realm to create for the FreeIPA domain.</td>
</tr>
<tr>
<td>--subject=<strong>subject_DN</strong></td>
<td>Sets the base element for the subject DN of the issued certificates. This defaults to <code>O=realm</code>.</td>
</tr>
<tr>
<td>--forwarder=<strong>forwarder</strong></td>
<td>Gives a DNS forwarder to use with the DNS service. To specify more than one forwarder, use this option multiple times.</td>
</tr>
<tr>
<td>--no-forwarders</td>
<td>Uses root servers with the DNS service instead of forwarders.</td>
</tr>
<tr>
<td>--no-reverse</td>
<td>Does not create a reverse DNS zone when the DNS domain is set up. (If a reverse DNS zone is already configured, then that existing reverse DNS zone is used.) If this option is not used, then the default value is true, which assumes that reverse DNS should be configured by the installation script.</td>
</tr>
</tbody>
</table>
| --setup-dns                        | Tells the installation script to set up a DNS service within the FreeIPA domain. Using an integrated DNS service is optional, so if this...                                                                 
## 2.3.2. Setting up a FreeIPA Server: Basic Interactive Installation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

   ```bash
   IPA admin password:
   Password (confirm):
   ```
Chapter 2. Installing a FreeIPA Server

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

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

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

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

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

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

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

   Configuring the web interface: Estimated time 1 minute
   
   [1/12]: disabling mod_ssl in httpd
   ...
   done configuring httpd.

   Setting the certificate subject base
   restarting certificate server
   Applying LDAP updates
   Restarting the directory server
   Restarting the KDC
   Restarting the web server
   Sample zone file for bind has been created in /tmp/sample.zone.ygzij5.db
   ==========================================================================
   Setup complete

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

   [root@server ~]# service sshd restart

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

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

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

    [root@server ~]# ipa user-find admin
Examples of Creating the FreeIPA Server

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

2.3.3.1. Non-Interactive Basic Installation

As shown in Section 2.3.2, "Setting up a FreeIPA Server: Basic Interactive Installation", only a few pieces of information are required to configure a FreeIPA server. While the setup script can prompt for this information in interactive mode, this information can also be passed with the setup command to allow automated and unattended configuration:

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

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

Example 2.1. Basic Installation without Interaction

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

The script then prints the submitted values:

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

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

The server name must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures. Additionally, the hostname must all be lower-case. No capital letters are allowed.
Then the script runs through the configuration progress for each FreeIPA service, as in Section 2.3.2, “Setting up a FreeIPA Server: Basic Interactive Installation”.

### 2.3.3.2. Using Different CA Configurations

FreeIPA uses an integrated certificate authority (CA) to create the certificates and keytabs used by users and hosts within the domain. There are three different ways that FreeIPA incorporates the CA into the FreeIPA server:

- The installation script installs a root Dogtag Certificate System CA. The Dogtag Certificate System CA provides the full range of certificate services, based on policies that are defined in the Dogtag Certificate System configuration.

  This is the default configuration.

- Alternatively, the installation script can set up a Dogtag Certificate System CA that is subordinate to an external CA. A subordinate CA is chained to another CA, and it uses the policies and restrictions defined by that external CA. The root CA can be an external CA like Verisign or a corporate CA.

  A Dogtag Certificate System CA is still installed and configured as part of the FreeIPA server installation, but its CA certificates are issued by the external CA rather than by itself.

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

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

#### Example 2.2. Using an External CA

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

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

2. The script sets up the NTP and Directory Server services as normal.

3. The script completes the CA setup and returns information about where the certificate signing request (CSR) is located, `/root/ipa.csr`. This request must be submitted to the external CA.

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

4. Submit the request to the CA. The process differs for every service.

5. Retrieve the issued certificate and the CA certificate chain for the issuing CA. Again, the process differs for every certificate service, but there is usually a download link on a web page or in the notification email that allows administrators to download all the required certificates. Be sure to get the full certificate chain for the CA, not just the CA certificate.
6. Rerun `ipa-server-install`, specifying the locations and names of the certificate and CA chain files. For example:

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

7. Complete the setup process and verify that everything is working as expected, as in [Section 2.3.2, “Setting up a FreeIPA Server: Basic Interactive Installation”](#).

### 2.3.3.3. Using DNS

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

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

**Example 2.3. Interactive DNS Setup**

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

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

2. The script configures the hostname and domain name as normal.

3. The script then prompts for DNS forwarders. If forwarders will be used, enter yes, and then supply the list of DNS servers. If FreeIPA will manage its own DNS service, then enter no.

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

4. The script sets up the NTP, Directory Server, Certificate System, Kerberos, and Apache services.

5. Before completing the configuration, the script prompts to ask whether it should configure reverse DNS services. If you select yes, then it configures the named service.

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

6. Verify that everything is working as expected, as in [Section 2.3.2, “Setting up a FreeIPA Server: Basic Interactive Installation”](#).
Chapter 2. Installing a FreeIPA Server

If DNS is used with FreeIPA, then two pieces of information are required: any DNS forwarders that will be used and using (or not) reverse DNS. To perform a non-interactive setup, this information can be passed using the `--forwarder` or `--no-forwarders` option and `--no-reverse` option.

**Example 2.4. Setting up DNS Non-Interactively**

To use DNS always requires the `--setup-dns`. To user forwarders, use the `--forwarder` option; for multiple forwarders, use multiple invocations of `--forwarder`.

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

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

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

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

### 2.3.4. Troubleshooting Installation Problems

The server installation log is located in `/var/log/ipaserver-install.log`. The FreeIPA logs, both for the server and for FreeIPA-associated services, are covered in *Section 18.1.3, “Checking FreeIPA Server Logs”*. 

**GSS Failures When Running IPA Commands**

Immediately after installation, there can be Kerberos problems when trying to run an `ipa-*` command. For example:

```bash
ipa: ERROR: Kerberos error: (‘Unspecified GSS failure. Minor code may provide more information’, 851968)/ (‘Decrypt integrity check failed’, -1765328353)
```

There are two potential causes for this:

- DNS is not properly configured.
- Active Directory is in the same domain as the FreeIPA server.

**named Daemon Fails to Start**

If a FreeIPA server is configured to manage DNS and is set up successfully, but the named service fails to start, this can indicate that there is a package conflict. Check the `/var/log/messages` file for error messages related to the `named` service and the `ldap.so` library:

```bash
ipaserver named[6886]: failed to dynamically load driver 'ldap.so': libldap-2.4.so.2: cannot open shared object file: No such file or directory
```

This usually means that the `bind-chroot` package is installed and is preventing the named service from starting. To resolve this issue, remove the `bind-chroot` package and then restart the FreeIPA server.

```bash
[root@server ~]# yum remove bind-chroot
```
2.4. Setting up FreeIPA Replicas

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

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

A replica is a clone of a specific FreeIPA server. The server and replica share the same internal information about users, machines, certificates, and configured policies. These data are copied from the server to the replica in a process called replication. The two Directory Server instances used by an FreeIPA server — the Directory Server instance used by the FreeIPA server as a data store and the Directory Server instance used by the Dogtag Certificate System to store certificate information — are replicated over to corresponding consumer Directory Server instances used by the FreeIPA replica.

**TIP**

If you are using a Dogtag Certificate System instance as the CA for the FreeIPA domain, then it is possible to make a replica of a replica.

2.4.1. Prepping and Installing the Replica Server

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

- Make sure that the machine meets all of the prerequisites listed in Section 2.1, “Preparing to Install the FreeIPA Server”.

- The replica must be the same version as the original master server. If the master server is running on Red Hat Enterprise Linux 6.3, FreeIPA version 2.2.x, then the replica must also run on Red Hat Enterprise Linux 6.3 and use the FreeIPA 2.2.x packages.

**IMPORTANT**

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

- Install the server packages as in Section 2.2, “Installing the FreeIPA Server Packages”. For example:

  [root@server ~]# yum install freeipa-server bind bind-dyndb-ldap
Chapter 2. Installing a FreeIPA Server

The replica and the master server must be running the same version of FreeIPA.

- If there is an existing Dogtag Certificate System or Red Hat Certificate System instance on the replica machine, make sure that port 7389 is free. This port is used by the master FreeIPA server to communicate with the replica.

- Make sure the appropriate ports are open on both the server and the replica machine during and after the replica configuration. Servers and replicas connect to each other over ports 9443, 9444, 9445, and 7389 during the replica configuration. Once the replica is set up, the server and replica communicate over port 7389.

2.4.2. Creating the Replica

1. On the master server, create a replica information file. This contains realm and configuration information taken from the master server which will be used to configure the replica server.

Run the `ipa-replica-prepare` command on the master FreeIPA server. The command requires the fully-qualified domain name of the replica machine. Using the `--ip-address` option automatically creates DNS entries for the replica, including the A and PTR records for the replica to the DNS.

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

**IMPORTANT**

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

Additionally, the hostname must all be lower-case. No capital letters are allowed.

For more options with `ipa-replica-prepare`, see [Section C.5.3, “ipa-replica-prepare”](#).

Each replica information file is created in the `/var/lib/ipa/` directory as a GPG-encrypted file. Each file is named specifically for the replica server for which it is intended, such as `replica-info-ipareplica.example.com.gpg`. 

---

**IMPORTANT**

Do not run the `ipa-server-install` script.
A replica information file cannot be used to create multiple replicas. It can only be used for the specific replica and machine for which it was created.

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

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

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

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

For example:

```
[root@ipareplica ~]# ipa-replica-install --setup-ca --setup-dns /var/lib/ipa/replica-
    info-ipareplica.example.com.gpg
Directory Manager (existing master) password:
Warning: Hostname (ipareplica.example.com) not found in DNS
Run connection check to master
Check connection from replica to remote master 'ipareplica. example.com':
    Directory Service: Unsecure port (389): OK
    Directory Service: Secure port (636): OK
    Kerberos KDC: TCP (88): OK
    Kerberos Kpasswd: TCP (464): OK
    HTTP Server: Unsecure port (80): OK
    HTTP Server: Secure port (443): OK
The following list of ports use UDP protocol and would need to be checked manually:
    Kerberos KDC: UDP (88): SKIPPED
    Kerberos Kpasswd: UDP (464): SKIPPED
Connection from replica to master is OK.
Start listening on required ports for remote master check
Get credentials to log in to remote master
admin@EXAMPLE.COM password:
Execute check on remote master
admin@example.com's password:
Check connection from master to remote replica 'ipareplica. example.com':
    Directory Service: Unsecure port (389): OK
```
Chapter 2. Installing a FreeIPA Server

Directory Service: Secure port (636): OK
Kerberos KDC: TCP (88): OK
Kerberos KDC: UDP (88): OK
Kerberos Kpasswd: TCP (464): OK
Kerberos Kpasswd: UDP (464): OK
HTTP Server: Unsecure port (80): OK
HTTP Server: Secure port (443): OK
Connection from master to replica is OK.
Connection check OK

Additional options for the replica installation script are listed in Section C.5.2, “ipa-replica-install”.

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

4. Enter the Directory Manager password when prompted. The script then configures a Directory Server instance based on information in the replica information file and initiates a replication process to copy over data from the master server to the replica, a process called initialization.

5. Verify that the proper DNS entries were created so that FreeIPA clients can discover the new server. DNS entries are required for required domain services:

- _ldap._tcp
- _kerberos._tcp
- _kerberos._udp
- _kerberos-master._tcp
- _kerberos-master._udp
- _kpasswd._tcp
- _kpasswd._udp
- _ntp._udp

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

```
[root@ipareplica ~]# DOMAIN=example.com
[root@ipareplica ~]# NAMESERVER=ipareplica
[root@ipareplica ~]# for i in _ldap._tcp _kerberos._tcp _kerberos._udp _kerberos-master._tcp _kerberos-master._udp _kpasswd._tcp _kpasswd._udp _ntp._udp; do echo $i; dig @$NAMESERVER $i.${DOMAIN} srv +nocmd +noquestion +nocomments +nostaats +noa
```

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

```
  _ldap._tcp.example.com. 86400 IN SRV 0 100 389 ipaserver1.example.com.
  _ldap._tcp.example.com. 86400 IN SRV 0 100 389 ipaserver2.example.com.
  _kerberos._tcp.example.com. 86400 IN SRV 0 100 88 ipaserver1.example.com.
  ...8<...
```
6. Optional. Set up DNS services for the replica. These are not configured by the setup script, even if the master server uses DNS.

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

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

**IMPORTANT**

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

---

### 2.4.3. Troubleshooting Replica Installation

#### Certificate System setup failed.

If the replica installation fails on step 3 ([3/11]: configuring certificate server instance), that usually means that the required port is not available. This can be verified by checking the debug logs for the CA, `/var/log/pki-ca/debug`, which may show error messages about being unable to find certain entries. For example:

```
[04/Feb/2012:22:29:03][http-9445-Processor25]: DatabasePanel compareAndWaitEntries ou=people,o=ipaca not found, let's wait
```

The only resolution is to uninstall the replica:

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

After uninstalling the replica, ensure that port 7389 on the replica is available, and retry the replica installation.

#### There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.

When the replica starts, there can be a series of SASL bind errors recorded in the 389 Directory Server logs stating that the GSS-API connection failed because it could not find a credentials cache:

```
slapd_ldap_sasl_interactive_bind - Error: could not perform interactive bind for id [] mech [GSSAPI]: error -2 (Local error) (SASL(-1): generic failure: GSSAPI Error: Unspecified GSS failure. Minor code may provide more information (Credentials cache file '/tmp/krb5cc_496' not found)) ...
```
The replica is looking for a credentials cache in /tmp/krb5cc_496 (where 496 is the 389 Directory Server user ID) and cannot find it.

There may also be messages that the server could not obtain Kerberos credentials for the host principal:

```
set_krb5_creds - Could not get initial credentials for principal [ldap/ replica1.example.com] in keytab [WRFILE:/etc/dirsrv/ds.keytab]: -1765328324 (Generic error)
```

These errors are both related to how and when the 389 Directory Server instance loads its Kerberos credentials cache.

While 389 Directory Server itself supports multiple different authentication mechanisms, FreeIPA only uses GSS-API for Kerberos connections. The 389 Directory Server instance for FreeIPA keeps its Kerberos credentials cache in memory. When the 389 Directory Server process ends — like when the FreeIPA replica is stopped — the credentials cache is destroyed.

Also, the 389 Directory Server is used as the backend storage for the principal information for the KDC.

When the replica then restarts, the 389 Directory Server instance starts first, since it supplies information for the KDC, and then the KDC server starts. This start order is what causes the GSS-API and Kerberos connection errors.

The 389 Directory Server attempts to open a GSS-API connection, but since there is no credentials cache yet and the KDC is not started, the GSS connection fails. Likewise, any attempt to obtain the host credentials also fails.

These errors are transient. The 389 Directory Server re-attempts the GSS-API connection after the KDC starts and it has a credentials cache. The 389 Directory Server logs then record a `bind resumed` message.

These startup GSS-API connection failures can be ignored as long as that connection is successfully established.

### 2.5. Uninstalling FreeIPA Servers and Replicas

To uninstall both a FreeIPA server and a FreeIPA replica, pass the `-uninstall` option to the `ipa-server-install` command:

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

### 2.6. Upgrading from FreeIPA 2.1 to 2.2

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

#### 2.6.1. Upgrading Packages

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

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

This is the easiest way to upgrade the server because it automatically pulls in updates for related services, like SSSD, which provide FreeIPA functionality.
To upgrade the FreeIPA server packages specifically, run `yum` on the master server:

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

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

The update process automatically updates all schema and LDAP configuration, Apache configuration, and other services configuration, and restarts all FreeIPA-associated services.

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

**NOTE**

Schema changes are replicated between servers. So once one master server is updated, all servers and replicas will have the updated schema, even if their packages are not yet updated. This ensures that any new entries which use the new schema can still be replicated among all the servers in the FreeIPA domain.

**NOTE**

Clients do not need to have new packages installed. The client packages used to configure a Fedora system do not impact the enrollment of the client within the domain.

Updating client packages could bring in updated packages for other dependencies, such as `certmonger` which contain bug fixes, but this is not required to maintain client functionality or behavior within the FreeIPA domain.

### 2.6.2. Removing Browser Configuration for Ticket Delegation

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

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

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

FreeIPA 2.2 uses the Kerberos Services for User to Proxy (S4U2Proxy), so this additional delegation step is no longer required.

**Updating Existing Configured Browsers**

For browsers which have already been configured to use the FreeIPA web UI, the `delegation-uris` setting can be cleared after upgrading to `ipa-server-2.2.0` or `ipa-client-2.2.0`.
Chapter 2. Installing a FreeIPA Server

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

### Updating configure.jar for New Browser Configuration

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

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

**NOTE**

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

1. Update the packages on the initial FreeIPA master server (the first instance). This will bring in the 2.2 UI packages, including the `configure.jar` file.

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

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

3. Create a temporary working directory.

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

4. Copy the updated `preference.html` file to the working directory.

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

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

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

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

6. Copy the regenerated `configure.jar` file to all other FreeIPA servers and replicas.

### 2.6.3. Testing Before Upgrading the FreeIPA Server

It can be beneficial, and safer, to test newer versions of FreeIPA before upgrading production systems. There is a relatively simple way to do this by creating a sacrificial replica and testing on that system.
1. Set up a replica based on one of the production servers, with the same version of FreeIPA as is running in production, as described in Section 2.4, "Setting up FreeIPA Replicas". For this example, this is called Test Replica. Make sure that Test Replica can successfully connect to the production server and domain.

2. After verifying that Test Replica has been successfully added to the production domain, disconnect Test Replica from the network.

3. Remove the replication agreements for Test Replica from the original FreeIPA server and from Test Replica.

4. Reconnect Test Replica to the network.

5. Upgrade the packages on Test Replica using `yum` or whatever package update tool is appropriate for your system. For example:

   [root@ipareplica ~]# yum update ipa

6. Test common things on Test Replica, like getting Kerberos credentials, opening the server UI, and running commands.
Setting up Systems as FreeIPA Clients

A client is any system which is a member of the FreeIPA domain. While this is frequently a Fedora system (and FreeIPA has special tools to make configuring Fedora clients very simple), machines with other operating systems can also be added to the FreeIPA domain.

One important aspect of a FreeIPA 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**

FreeIPA 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 project page.

This chapter explains how to configure a system to join a FreeIPA domain.

**NOTE**

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

3.1. What Happens in Client Setup

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

- Retrieve the CA certificate for the FreeIPA CA.

- Create a separate Kerberos configuration to test the provided credentials. This enables a Kerberos connection to the FreeIPA XML-RPC server, necessary to join the FreeIPA client to the FreeIPA 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 Fedora systems:

```
[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = false
rdns = false
```

1 https://fedorahosted.org/sssd/
Chapter 3. Setting up Systems as FreeIPA Clients

```plaintext
forwardable = yes
ticket_lifetime = 24h

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

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

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

### 3.2. Supported Platforms for FreeIPA Clients

These platforms can be configured to be FreeIPA clients:

- Fedora 14, 15, 16, and 17
- Windows XP and later
- Solaris 9 and 10
- HP-UX 11i
- AIX 5.3 and 6.1

### 3.3. System Ports

FreeIPA uses a number of ports to communicate with its services. These ports, listed in Table 3.1, "FreeIPA Ports", must be open and available for FreeIPA to work. They cannot be in use by another service or blocked by a firewall. To make sure that these ports are available, try `iptables` to list the available ports or `nc`, `telnet`, or `nmap` to connect to a port or run a port scan.

To open a port:

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

The `iptables` man page has more information on opening and closing ports on a system.
### Table 3.1. FreeIPA Ports

<table>
<thead>
<tr>
<th>Service</th>
<th>Ports</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP/HTTPS</td>
<td>80</td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>443</td>
<td></td>
</tr>
<tr>
<td>LDAP/LDAPS</td>
<td>389</td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>636</td>
<td></td>
</tr>
<tr>
<td>Kerberos</td>
<td>88</td>
<td>TCP and UDP</td>
</tr>
<tr>
<td></td>
<td>464</td>
<td></td>
</tr>
<tr>
<td>DNS</td>
<td>53</td>
<td>TCP and UDP</td>
</tr>
<tr>
<td>NTP</td>
<td>123</td>
<td>UDP</td>
</tr>
</tbody>
</table>

### 3.4. Configuring a Fedora System as a FreeIPA Client

There are two elements to prepare before beginning the client setup process for the Fedora 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 FreeIPA server address. The `ipa-client-install` script retrieves the Active Directory DNS records instead of any records that were added for FreeIPA.

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

To configure the client:

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

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

   ```
   # yum install freeipa-client
   ```

   An administrator machine requires the `freeipa-admintools` package, as well:

   ```
   # yum install freeipa-client freeipa-admintools
   ```

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

   **TIP**

   If every machine in the domain will be a FreeIPA client, then add the FreeIPA server address to the DHCP configuration.
3. Run the client setup command.

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

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

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

**IMPORTANT**

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

Other options for `ipa-client-install` are listed in Section C.6.1, “ipa-client-install”.

**NOTE**

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

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

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

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

```
DNS discovery failed to find the IPA Server
Please provide your IPA server name (ex: ipa.example.com): ipaserver.example.com
```
6. The client script then prompts for a Kerberos identity to use to contact and then join the Kerberos realm. When these credentials are supplied, then the client is able to join the FreeIPA Kerberos domain and then complete the configuration:

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

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

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

8. Set up NFS to work with Kerberos.

**TIP**

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

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

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

```bash
# ipa service-add nfs/ipaclient.example.com@EXAMPLE
```
b. On the FreeIPA server, obtain a keytab for the NFS service principal.

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

**NOTE**

Some versions of the Linux NFS implementation have limited encryption type support. If the NFS server is hosted on a version older than Fedora 16, use the `-e des-cbc-crc` option to the `ipa-getkeytab` command for any nfs/<FQDN> service keytabs to set up, both on the server and on all clients. This instructs the KDC to generate only DES keys.

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

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

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

d. Configure the `/etc/exports` file on the NFS server.

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

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

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

### 3.5. 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 FreeIPA client entries and the services can be configured manually.
1. Install SSSD 1.5.x or later, if it is not already installed.

2. Optional. Install the FreeIPA tools so that administrative tasks can be performed from the host.

   ```
   # yum install freeipa-admintools
   ```

3. On a FreeIPA server. Create a host entry for the client.

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

   Creating hosts manually is covered in Section 6.2, “Adding Host Entries”.

4. On a FreeIPA server. Create keytabs for the client.

   a. Log in as FreeIPA administrator.

   ```
   $ kinit admin
   ```

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

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

   c. Generate the keytab for the client.

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

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

   **Tip**

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

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

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

7. Configure SSSD by editing the `/etc/sssd/sssd.conf` file to point to the FreeIPA domain.

   ```
   [sssd]
   config_file_version = 2
   services = nss, pam
   domains = example.com
   [nss]
   [pam]
   ```
Chapter 3. Setting up Systems as FreeIPA Clients

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

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

vim /etc/nsswitch.conf
...
passwd:     files sss
shadow:     files sss
group:      files sss
...
netgroup:   files sss
...

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

[logging]
default = FILE:/var/log/krb5libs.log
dc = FILE:/var/log/krb5kdc.log
dadmin_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
domain.com = EXAMPLE.COM

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

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

... account     [default=bad success=ok user_unknown=ignore] pam_sss.so
... session     optional       pam_sss.so
• For `/etc/pam.d/system-auth`:

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

• For `/etc/pam.d/password-auth`:

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

• For `/etc/pam.d/smartcard-auth`:

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

11. Install the FreeIPA server’s CA certificate.
   
a. Obtain the certificate from the server.

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

   b. Install the certificate in the system’s NSS database.

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

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

   Use the `ipa-getcert` command, which creates and manages the certificate through `certmonger`. The options are described more in Section B.1, "Requesting a Certificate with certmonger".

   ```bash
   $ 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 a FreeIPA server, copied over to the host, and installed using `certutil`.

13. Set up NFS to work with Kerberos.
TIP

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

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

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

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

NOTE

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

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

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

NOTE

Some versions of the Linux NFS implementation have limited encryption type support. If the NFS server is hosted on a version older than Fedora 16, use the `-e des-cbc-crc` option to the `ipa-getkeytab` command for any nfs/<FQDN> service keytabs to set up, both on the server and on all clients. This instructs the KDC to generate only DES keys.

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

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

```
# scp /tmp/krb5.keytab root@nfs.example.com:/etc/krb5.keytab
```
d. Copy the keytab from the FreeIPA server to the FreeIPA client. For example:

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

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

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

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

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

### 3.6. Setting up a Linux Client Through Kickstart

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

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

1. Create the host entry on the FreeIPA 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 FreeIPA domain. However, when the script is run automatically, the system has to have some way to access the FreeIPA domain without using an existing FreeIPA 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 FreeIPA domain.

   For example:

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

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

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

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

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

   ```
   %post --log=/root/ks-post.log
   # Get the hostname to set as the host principal
   ```
4. Run the kickstart script.

### 3.7. Configuring a Microsoft Windows System to Join the FreeIPA Realm

1. Download the MIT Kerberos 3.x package for Windows.

   http://web.mit.edu/kerberos/dist/index.html

2. Run the `kfw-3.x-exe` file to launch the MIT Kerberos Installation Wizard.

3. Read and accept the license agreement.

4. Install the KfW client. All other components are optional.

5. Accept the default destination path.

6. Select **Download from web path**, and enter the URL to the FreeIPA server. For example:

   http://ipaserver.example.com/ipa/config/

   Include the trailing backslash, or the configuration will fail.

7. Select **Autostart the Network Identity Manager each time you login to Windows**.

8. Click **Install** to begin the installation. When the installation is complete, click **Finish** to exit the Wizard.

9. Edit the hosts file and add the FreeIPA server. For example:

   1.2.3.4     ipaserver.example.com     ipaserver

   Depending on the version of Windows, the HOSTS file could be located in different directories. For Windows XP and later systems, this is in `C:\WINDOWS\system32\drivers\etc\`.

#### NOTE

One potential problem is that a ticket is not generated by Kerberos on Windows. Windows can use multiple ticket caches with MIT Kerberos. This can create odd scenarios, where it is possible to authenticate against FreeIPA's domain in the command line, but not to open the web UI.

MIT Kerberos for Windows provides some debugging tools which can be used to troubleshoot Windows Kerberos problems, available at [http://web.mit.edu/Kerberos/dist/index.html#kfw-3.2](http://web/mit.edu/Kerberos/dist/index.html#kfw-3.2)
3.8. Configuring a Solaris System as a FreeIPA Client

3.8.1. Configuring Solaris 10

1. FreeIPA provides an example profile for configuring Solaris 10 as a FreeIPA client. This can be loaded using `ldapclient` and the `init` command:

```
[root@solaris ~]# ldapclient init ipa.example.com
```

The `ldapclient` can also be run to enter the information for the FreeIPA domain manually:

```
[root@solaris ~]# ldapclient manual
-a credentialLevel=proxy
-a authenticationMethod=tls:simple
-a defaultSearchBase=dc=example,dc=com
-a domainName=example.com
-a defaultServerList=192.168.0.1
-a proxyDN=cn=proxyagent,ou=profile,dc=example,dc=com
-a proxyPassword={NS1}fbc123a92116812
-a objectClassMap=group:posixGroup=posixgroup
-a objectClassMap=passwd:posixAccount=posixaccount
-a objectClassMap=shadow:shadowAccount=posixAccount
-a serviceSearchDescriptor=passwd:cn=users,cn=accounts,dc=example,dc=com
-a serviceSearchDescriptor=group:cn=groups,cn=accounts,dc=example,dc=com
-a serviceSearchDescriptor=shadow:cn=sysaccounts,cn=etc,dc=example,dc=com
-a serviceSearchDescriptor=sudoers:cn=sysaccounts,cn=etc,dc=example,dc=com
```

2. Create a Solaris profile in the FreeIPA Directory Server instance for the Solaris domain clients to use. The LDAP entry should reflect the configuration that was passed to the Solaris machine in the `ldapclient` command:

```
[root@ipaserver ~]# ldapadd -h 192.168.0.1 -p 389 -D "cn=directory manager" -w secret
dn: cn=solaris,ou=profile,dc=example,dc=com
objectClass: top
objectClass: DUAConfigProfile
cn: solaris
credentialLevel: proxy
authenticationMethod: tls:simple
defaultSearchBase: dc=example,dc=com
defaultServerList: 192.168.0.1
objectClassMap: group:posixGroup=posixgroup
objectClassMap: passwd:posixAccount=posixAccount
objectClassMap: shadow:shadowAccount=posixAccount
serviceSearchDescriptor: passwd:cn=users,cn=accounts,dc=example,dc=com
serviceSearchDescriptor: group:cn=groups,dc=example,dc=com
serviceSearchDescriptor: shadow:cn=sysaccounts,cn=etc,dc=example,dc=com
serviceSearchDescriptor: netgroup:cn=sysaccounts,cn=etc,dc=example,dc=com
serviceSearchDescriptor: sudoers:cn=sysaccounts,cn=etc,dc=example,dc=com
bindTimeLimit: 10
profileTTL: 43200
searchTimeLimit: 30
defaultSearchScope: one
```
3. Create the `cn=proxyagent` account in the FreeIPA Directory Server instance.

```bash
[root@ipaserver ~]# ldapadd -h 192.168.0.1 -p 389 -D "cn=directory manager" -w secret
dn: cn=proxyagent,ou=profile,dc=example,dc=com
objectClass: top
objectClass: person
sn: proxyagent
cn: proxyagent
userPassword:: e1NTSEF9Mm53KytGeU81Z1dka1FLNUZ1aDdXOHJkJK93TEppY2NjRmt6Whnc9PQ=
```

4. On the FreeIPA server, use the `certutil` command to create `cert8.db` and `key3.db` databases.

```bash
[root@ipaserver ~]# certutil -N -d .
Then, copy the database over to the Solaris machine in the `/var/ldap` directory. For example:

```bash
[root@ipaserver ~]# scp cert8.db solaris.example.com:/var/ldap
[root@ipaserver ~]# scp key3.db solaris.example.com:/var/ldap
```

5. Remove the `ldap` option from all entries in `/etc/nsswitch.conf` except for the `passwd`, `group`, `shadow`, `netgroup`, and `sudoers` entries.

6. Configure and enable NTP and synchronize the time between the client and the FreeIPA server.

```bash
[root@solaris ~]# ntpdate ipaserver.example.com
```

7. Configure the Kerberos client. The Kerberos configuration includes specifying the realm and domain details and default ticket attributes.

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

[libdefaults]
default_realm = EXAMPLE.COM
verify_ap_req_nofail = false

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

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

[logging]
default = FILE:/var/krb5/kdc.log
kdc = FILE:/var/krb5/kdc.log

[appdefaults]
kinit = {
renewable = true
forwardable= true
}
```
The default file created by `ldapclient` configures forwardable tickets by default, which makes it possible to connect to the UI from any system and provides a way to audit administration operations.

8. Configure PAM to use Kerberos authentication. For example:

```
[root@solarsis ~]# vim /etc/pam.conf

# login service (explicit because of pam_dial_auth)
#
login auth requisite pam_authtok_get.so.1
login auth required pam_dhkeys.so.1
login auth sufficient pam_krb5.so.1 try_first_pass
login auth required pam_unix_auth.so.1
login auth required pam_dial_auth.so.1

# Default definitions for Authentication management
# Used when service name is not explicitly mentioned for authentication
#
other auth requisite pam_authtok_get.so.1
other auth required pam_dhkeys.so.1
other auth required pam_unix_cred.so.1
other auth sufficient pam_krb5.so.1
other auth required pam_unix_auth.so.1

# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
#
other account requisite pam_roles.so.1
other account required pam_unix_account.so.1
other account required pam_krb5.so.1

# Password construction requirements apply to all users.
# Remove force_check to have the traditional authorized administrator
# bypass of construction requirements.
other password requisite pam_authtok_check.so.1 force_check
other password sufficient pam_krb5.so.1
other password required pam_authtok_store.so.1
```

9. Configure NFS to work with the Kerberos domain.

a. Add an NFS service principal for the client.

```
[root@ipaserver ~]# ipa service-add nfs/client.example.com
```

b. Create the NFS keytab file.

```
[root@ipaserver ~]# ipa-getkeytab -s ipaserver.example.com -p nfs/client.example.com
-k /tmp/krb5.keytab -e des-cbc-crc
```

c. Copy the keytab from the server to the client.

```
[root@ipaserver ~]# scp /tmp/krb5.keytab root@client.example.com:/tmp/krb5.keytab
```

d. On the FreeIPA client, use the `ktutil` command to import the contents into the main host keytab.

```
# ktutil
ktutil: read_kt /tmp/krb5.keytab
ktutil: write_kt /etc/krb5/krb5.keytab
```
ktutil: q

e. Verify that the NFS service keytab was created:

```
[root@solaris ~]# klist -ket /etc/krb5/krb5.keytab
```

f. Verify that the NFS server is accessible:

```
[root@solaris ~]# showmount -e ipaserver.example.com
```

g. Make sure that this line is uncommented in the `/etc/nfssec.conf` file.

```
krb5 390003 kerberos_v5 default - # RPCSEC_GSS
```

h. Mount the NFS share.

```
[root@solaris ~]# mount -t nfs4 ipaserver.example.com:/ /mnt/ -o sec=krb5
```

10. Configure `sudo` on the Solaris machine to work with the FreeIPA server.

a. If necessary, install the required packages for SASL, OpenSSL, sudo and LDAP, and BerkeleyDB:

- CSWbdb4 (BerkeleyDB 4)
- CSWcommon
- CSWlibnet
- CSWoldaprt
- CSWossl
- CSWossldevel
- CSWosslrt
- CSWosslutils
- CSWsasl
- CSWsudo-common
- CSWsudoldap

These are available from Blastwave^2.

b. Edit the OpenLDAP `ldap.conf` file to use the secure URL for the FreeIPA Directory Server instance and to use the FreeIPA CA certificate.

```
[root@solaris ~]# vim /opt/csw/etc/openldap/ldap.conf
```

base dc=example,dc=com
timelimit 120
bind_timelimit 120
idle_timelimit 3600
uri ldaps://ipaserver.example.com
ssl start_tls
sudoers_base ou=SUDOers,dc=example,dc=com
ssl on
TLS_REQCERT allow
TLS_CACERT /etc/openldap/cacerts/ca.crt
TLS_CACERTFILE /etc/openldap/cacerts/ca.crt
TLS_CACERTDIR /etc/openldap/cacerts ...

3.8.2. Configuring Solaris 9
1. Perform steps 1 through 8 in Section 3.8.1, “Configuring Solaris 10” to set up the Solaris 9 client.
2. Configure the NFS client.
   a. Configure the /etc/exports file on the NFS server.
      
```bash
/nfs client.example.com(sec=krb5p,rw,sync,fsid=0,no_subtree_check)
```
   b. Add an NFS service principal for the client.
      
```
[root@ipaserver ~]# ipa service-add nfs/client.example.com
```
   c. Create the NFS keytab file.
      
```
[root@ipaserver ~]# ipa-getkeytab -s ipaserver.example.com -p nfs/client.example.com -k /tmp/krb5.keytab -e des-cbc-crc
```
   d. Copy the keytab from the server to the client.
      
```
[root@ipaserver ~]# scp /tmp/krb5.keytab root@client.example.com:/tmp/krb5.keytab
```
   e. Make sure that this line is uncommented in the /etc/nfssec.conf file.
      
```
krb5 390005 kerberos_v5 default - # RPCSEC_GSS
```
   f. Obtain a ticket for the NFS client.
      
```
[root@solaris ~]# kinit -k nfs/client.example.com
```
   g. Mount the NFS share.
      
```
[root@solaris ~]# mount -F nfs -o sec=krb5p ipaserver.example.com:/nfs /mnt/
```
   h. On the FreeIPA client, use the ktutil command to import the contents into the main host keytab.
      
```
...
3.9. Configuring an HP-UX System as a FreeIPA Client

3.9.1. Configuring NTP
Configure and enable NTP and make sure that time is synchronized between the client and the FreeIPA server.

3.9.2. Configuring LDAP Authentication
1. Install the `ldapux` client.

   ```
   [root@hp-server ~]# swinstall -s /path/to/J4269AA_B.04.15.01_HP-UX_B.11.23_IA_PA.depot
   ```

2. Change to the configuration directory, and run the setup script.

   ```
   [root@hp-server ~]# cd /opt/ldapux/config
   # ./setup
   ```

   **NOTE**
   Running the setup script is only necessary for the first HP-UX client. Every subsequent HP-UX client only needs to know where the LDAP profile is stored. All clients will then use the same configuration.

   For more information on this, see the HP-UX documentation at [http://docs.hp.com/en/J4269-90075/ch02s07.html](http://docs.hp.com/en/J4269-90075/ch02s07.html).

   The setup script prompts for information about the FreeIPA LDAP service, such as its port and host, Directory Manager credentials, and schema and directory suffixes.

   Would you like to continue with the setup? [Yes]
   Select which Directory Server you want to connect to? [RedHat Directory]
   Directory server host? [ipaserver.example.com]
   Directory Server port number [389]
   Would you like to extend the printer schema in this directory server? [No]
   Would you like to install PublicKey schema in this directory server? [No]
   Would you like to install the new automount schema? [No]
3. Ensure that the LDAP client daemon is running.

```bash
# ps -ef | grep ldapclientd
```

If necessary, start the daemon:

```bash
# /opt/ldapux/bin/ldapclientd
```

4. Check that the user and group entries in the LDAP client are correct and available:

```bash
# nsquery passwd admin
# nsquery group admins
```

5. Create a new group on the FreeIPA server.

```bash
# ipa group-add testgroup
```

6. Add a test user to the new group.

```bash
# ipa group-add-member -a testuser testgroup
```

Validate the new user and group:

```bash
# nsquery passwd testuser
# nsquery group testgroup
```

7. To ensure that the LDAP client daemon starts when the system boots, add the following lines to the `/etc/opt/ldapux/ldapclientd.conf` file:

```
[StartOnBoot]
enable=yes
```
3.9.3. Configuring Kerberos

Edit the `/etc/krb5.conf` file to reflect the Kerberos domain used by the FreeIPA server. 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:

```
[lidbdefaults]
default_realm = EXAMPLE.COM
default_keytab_name = FILE:/etc/krb5.keytab
default_tkt_enctypes = des3-cbc-sha1 arcfour-hmac aes256-cts des-cbc-md5 des-cbc-crc
default_tgs_enctypes = des3-cbc-sha1 arcfour-hmac aes256-cts des-cbc-md5 des-cbc-crc
cache_type = 2

[realms]
EXAMPLE.COM = {
    kpasswd_server = ipaserver.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

[appdefaults]
kinit = {
    forwardable = true
}
```

3.9.4. Configuring PAM

The PAM configuration differs slightly between different versions of HP-UX.

3.9.4.1. HP-UX 11i v2

Edit the `/etc/pam.conf` file so that all of the required modules are loaded for authentication. For example:

```
# # PAM configuration
# # This pam.conf file is intended as an example only.
# see pam.conf(4) for more details
#
# Authentication management
#
login auth required libpam_hpsec.so.1
login auth sufficient libpam_krb5.so.1
login auth required libpam_unix.so.1 try_first_pass
su auth required libpam_hpsec.so.1
su auth sufficient libpam_krb5.so.1
su auth required libpam_unix.so.1 try_first_pass
dtlogin auth required libpam_hpsec.so.1
dtlogin auth sufficient libpam_krb5.so.1
dtlogin auth required libpam_unix.so.1 try_first_pass
dtaction auth required libpam_hpsec.so.1
dtaction auth sufficient libpam_krb5.so.1
dtaction auth required libpam_unix.so.1 try_first_pass
ftp auth required libpam_hpsec.so.1
ftp auth required libpam_krb5.so.1
```
ftp auth required libpam_unix.so.1 try_first_pass
sshd auth required libpam_hpsec.so.1
sshd auth sufficient libpam_krb5.so.1
sshd auth required libpam_unix.so.1 try_first_pass
OTHER auth required libpam_unix.so.1
#

# Account management
#
login account required libpam_hpsec.so.1
login account sufficient libpam_krb5.so.1
login account required libpam_unix.so.1
su account required libpam_hpsec.so.1
su account sufficient libpam_krb5.so.1
su account required libpam_unix.so.1
dtlogin account required libpam_hpsec.so.1
dtlogin account sufficient libpam_krb5.so.1
dtlogin account required libpam_unix.so.1
dtaction account required libpam_hpsec.so.1
dtaction account sufficient libpam_krb5.so.1
dtaction account required libpam_unix.so.1
ftp account required libpam_hpsec.so.1
ftp account sufficient libpam_krb5.so.1
ftp account required libpam_unix.so.1
sshd account required libpam_hpsec.so.1
sshd account sufficient libpam_krb5.so.1
sshd account required libpam_unix.so.1
OTHER account required libpam_unix.so.1
#

# Session management
#
login session required libpam_hpsec.so.1
login session sufficient libpam_krb5.so.1
login session required libpam_unix.so.1
dtlogin session required libpam_hpsec.so.1
dtlogin session sufficient libpam_krb5.so.1
dtlogin session required libpam_unix.so.1
dtaction session required libpam_hpsec.so.1
dtaction session sufficient libpam_krb5.so.1
dtaction session required libpam_unix.so.1
sshd session required libpam_hpsec.so.1
sshd session sufficient libpam_krb5.so.1
sshd session required libpam_unix.so.1
OTHER session required libpam_unix.so.1
#

# Password management
#
login password required libpam_hpsec.so.1
login password sufficient libpam_krb5.so.1
login password required libpam_unix.so.1
passwd password required libpam_hpsec.so.1
passwd password sufficient libpam_krb5.so.1
passwd password required libpam_unix.so.1
dtlogin password required libpam_hpsec.so.1
dtlogin password sufficient libpam_krb5.so.1
dtlogin password required libpam_unix.so.1
dtaction password required libpam_hpsec.so.1
dtaction password sufficient libpam_krb5.so.1
dtaction password required libpam_unix.so.1
OTHER password required libpam_unix.so.1

3.9.4.2. HP-UX 11i v1
Edit the /etc/pam.conf file to reflect the following example:
### PAM configuration

# This pam.conf file is intended as an example only.
# see pam.conf(4) for more details
#

# Authentication management
#
login auth sufficient /usr/lib/security/libpam_krb5.1
login auth required /usr/lib/security/libpam_unix.1 try_first_pass
su auth sufficient /usr/lib/security/libpam_krb5.1
su auth required /usr/lib/security/libpam_unix.1 try_first_pass
dlogin auth required /usr/lib/security/libpam_krb5.1
dlogin auth required /usr/lib/security/libpam_unix.1 try_first_pass
dtaction auth sufficient /usr/lib/security/libpam_krb5.1
dtaction auth required /usr/lib/security/libpam_unix.1 try_first_pass
ftp auth sufficient /usr/lib/security/libpam_krb5.1
ftp auth required /usr/lib/security/libpam_unix.1 try_first_pass
OTHER auth required /usr/lib/security/libpam_unix.1
#

# Account management
#
login account sufficient /usr/lib/security/libpam_krb5.1
login account required /usr/lib/security/libpam_unix.1
su account sufficient /usr/lib/security/libpam_krb5.1
su account required /usr/lib/security/libpam_unix.1
dlogin account sufficient /usr/lib/security/libpam_krb5.1
dlogin account required /usr/lib/security/libpam_unix.1
dtaction account sufficient /usr/lib/security/libpam_krb5.1
dtaction account required /usr/lib/security/libpam_unix.1
ftp account sufficient /usr/lib/security/libpam_krb5.1
ftp account required /usr/lib/security/libpam_unix.1
OTHER account required /usr/lib/security/libpam_unix.1
#

# Session management
#
login session sufficient /usr/lib/security/libpam_krb5.1
login session required /usr/lib/security/libpam_unix.1
dlogin session sufficient /usr/lib/security/libpam_krb5.1
dlogin session required /usr/lib/security/libpam_unix.1
dtaction session sufficient /usr/lib/security/libpam_krb5.1
dtaction session required /usr/lib/security/libpam_unix.1
OTHER session required /usr/lib/security/libpam_unix.1
#

# Password management
#
login password sufficient /usr/lib/security/libpam_krb5.1
login password required /usr/lib/security/libpam_unix.1
passwd password sufficient /usr/lib/security/libpam_krb5.1
passwd password required /usr/lib/security/libpam_unix.1
dlogin password sufficient /usr/lib/security/libpam_krb5.1
dlogin password required /usr/lib/security/libpam_unix.1
dtaction password sufficient /usr/lib/security/libpam_krb5.1
dtaction password required /usr/lib/security/libpam_unix.1
OTHER password required /usr/lib/security/libpam_unix.1
Configuring SSH

3.9.5. Configuring SSH
1. Ensure that you have version A.05.10.007 or later of ssh installed. A current package can be downloaded from the HP website at http://software.hp.com/portal/swdepot/displayProductInfo.do?productNumber=T1471AA.

2. Edit the /etc/opt/ssh/ssh_config file:
   • Remove any PreferredAuthentications entries.
   • Add the following lines:

   ```
   Host *
   GSSAPIAuthentication yes
   GSSAPITrustDNS no
   PreferredAuthentications "gssapi-with-mic,publickey,password"
   ```

   **IMPORTANT**
   Include the tab character before the GSSAPIAuthentication, GSSAPITrustDNS, and PreferredAuthentications lines, and include the double quotes around the PreferredAuthentications value.

3. Remove the /etc/krb5.keytab file.

4. Set up the NFS/Kerberos mapping for the Solaris client on the FreeIPA server.
   a. Add a host service principal for the HP-UX client.

   ```
   # ipa service-add host/hpuxipaclient.example.com
   ```

   b. Create the host keytab file.

   ```
   # ipa-getkeytab -s ipaserver.example.com -p host/hpuxipaclient.example.com -k /tmp/krb5.keytab -e des-cbc-crc
   ```

   c. Copy this keytab to the HP-UX machine, and save it as /etc/krb5/krb5.keytab.

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

3.9.6. Configuring Access Control
HP-UX systems provide the pam_authz PAM module, which can be used to control login access to the system based on a user's group membership. For details on how to configure access control with this module, see the HP documentation at http://h20000.www2.hp.com/bc/docs/support/SupportManual/c02261530/c02261530.pdf.

Example 3.1. pam_authz.policy File: Allow User Access, Deny Admin Access
This configuration in /etc/opt/ldapux/pam_authz.policy prevents the admin user from logging in while still allowing regular users to log in.
Chapter 3. Setting up Systems as FreeIPA Clients

### 3.9.7. Testing the Configuration

**NOTE**

By default, the admin user is given `/bin/bash` as the shell to use and `/home/admin` as the home directory. It may be necessary to install bash to be able to log in.

There are two quick ways to check the Kerberos and PAM configuration for the HP client:

- Authenticate as an administrator on a Linux box that is a client in the FreeIPA domain, and then attempt to SSH into the HP machine. The admin user should be able to log in using SSH without being asked for a password.

  ```bash
  # kinit admin
  # ssh admin@hpuxipaclient.example.com
  ```

- Log into the FreeIPA web UI using the administrator credentials on the HP machine.
3.10. Configuring an AIX System as a FreeIPA Client

3.10.1. Prerequisites
Make sure that all of these packages are installed on the AIX machine before beginning the client configuration:

- v5.3 OS
- v5.3 Updates
- krb5 client packages
- openssh
- wget
- bash
- krb5 server
- ldap.client
- openssl
- modcrypt.base (for gssd)

Configure and enable NTP and make sure that time is synchronized between the client and the FreeIPA server.

3.10.2. Configuring the AIX Client
Setting up an AIX client requires setting up the client to work in the FreeIPA Kerberos domain and, optionally, to enable SSH authentication to the AIX client using FreeIPA credentials.

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:

1. Configure the krb5 client settings to use the FreeIPA Kerberos domain:

   ```
   # mkkrb5clnt -r EXAMPLE.COM -d example.com -c ipaclient.example.com -s ipaserver.example.com
   ```

2. Get a Kerberos ticket:

   ```
   # kinit admin
   ```

3. On the FreeIPA server, add a user that is only used for authentication. (This can be substituted with krb5 authentication if that works from the LDAP client). Otherwise go to the FreeIPA server and use `ldapmodify`, bind as Directory Manager and create this user. The user should be assigned a shared password.

   ```
   # ldapmodify -D "cn=directory manager" -w secret -p 389 -h ipaserver.example.com -x -a
   ```
Chapter 3. Setting up Systems as FreeIPA Clients

4. On the AIX system, configure the LDAP client settings to use the FreeIPA directory services:

   ```bash
   # mksecldap -c -h ipaserver.example.com -d cn=accounts,dc=example,dc=com -a
   uid=nss,cn=sysaccounts,cn=etc,dc=example,dc=com -p secret
   ```

5. In the `/etc/security/ldap` directory, create user and group map files:

   - For example, for the `FreeIPAuser.map` file:

     ```bash
     #FreeIPAUser.map file
     keyobjectclass SEC_CHAR posixaccount s
     
     # The following attributes are required by AIX to be functional
     username SEC_CHAR uid s
     id SEC_INT uidnumber s
     pgrp SEC_CHAR gidnumber s
     home SEC_CHAR homedirectory s
     shell SEC_CHAR loginshell s
     gecos SEC_CHAR gecos s
     spassword SEC_CHAR userpassword s
     lastupdate SEC_INT shadowlastchange s
     ```

   - For example, for the `FreeIPAGroup.map` file:

     ```bash
     #FreeIPAGroup.map file
     groupname SEC_CHAR cn s
     id SEC_INT gidNumber s
     users SEC_LIST member m
     ```

6. Modify the `/etc/security/ldap/ldap.cfg` file to set the REALM and base DN values for the FreeIPA domain.

   ```bash
   userbasedn:cn=users,cn=accounts,dc=example,dc=com
   groupbasedn:cn=groups,cn=accounts,dc=example,dc=com
   userattrmappath:/etc/security/ldap/FreeIPAUser.map
groupattrmappath:/etc/security/ldap/FreeIPAGroup.map
   userclasses:posixaccount
   ```

7. Start the LDAP client daemon:

   ```bash
   # start-secdapclntd
   ```

8. Test the LDAP client connection to the FreeIPA server:

   ```bash
   # lsladap -a passwd
   ```
9. Add the following sections to the `/usr/lib/security/methods.cfg` file to configure the system login to use Kerberos and LDAP:

<table>
<thead>
<tr>
<th>Name</th>
<th>Program</th>
<th>Program 64</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>KRB5A</td>
<td>/usr/lib/security/KRB5A</td>
<td>/usr/lib/security/KRB5A_64</td>
<td>authonly</td>
</tr>
<tr>
<td>LDAP</td>
<td>/usr/lib/security/LDAP</td>
<td>/usr/lib/security/LDAP64</td>
<td></td>
</tr>
<tr>
<td>KRB5ALDAP</td>
<td></td>
<td></td>
<td>auth=KRB5A, db=LDAP</td>
</tr>
</tbody>
</table>

10. Edit the `/etc/security/user` file, and modify the `default` section to use the Kerberos/LDAP system and the LDAP user registry.

```bash
SYSTEM = "KRB5ALDAP"
registry = LDAP
```

11. To test the Kerberos configuration, log in as a FreeIPA user and verify that the user and group information is correct:

```bash
$ id
```

12. Optionally, configure the FreeIPA client to accept incoming SSH requests and authenticate with the user's Kerberos credentials.
   
   a. Set the SSH syslog configuration:

   ```bash
   auth.info /var/log/sshd.log
   auth.info /var/log/sshd.log
   auth.crit /var/log/sshd.log
   auth.warn /var/log/sshd.log
   auth.notice /var/log/sshd.log
   auth.err /var/log/sshd.log
   ```

   b. Set the SSH logging configuration:

   ```bash
   SyslogFacility AUTH
   LogLevel INFO
   ```

   c. Configure `sshd` to use GSS-API, including disabling DNS for GSS-API:

   ```bash
   vim /etc/ssh/sshd_config
   # GSSAPI options
   GSSAPIAuthentication yes
   #GSSAPICleanupCredentials yes
   GSSAPITrustDNS no
   ```

   d. Restart the `sshd` daemon:

   ```bash
   # stopsrc -s sshd
   # startsrc -s sshd
   ```
Chapter 3. Setting up Systems as FreeIPA Clients

e. Restart the `syslogd` daemon:

```
# stopsrc -s syslogd
# startsrc -s syslogd
```

f. Add the client to the FreeIPA server's Kerberos configuration.

   i. Add a host service principal for the client.

```
# ipa service-add host/ipaclient.example.com
```

   ii. Retrieve the host keytab.

```
# ipa-getkeytab -s ipaserver -p host/ipaclient.example.com -k /tmp/krb5.keytab -e des-cbc-crc
```

   iii. Copy the keytab from the server to the client.

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

g. On the FreeIPA client, use the `ktutil` command to import the contents into the main host keytab.

```
# ktutil
ktutil: read_kt /tmp/krb5.keytab
ktutil: write_kt /etc/krb5/krb5.keytab
ktutil: q
```

h. On the FreeIPA server, get a ticket for the admin user.

```
# kinit admin
```

i. To test the SSH configuration, try to log in as the admin user using SSH without providing a password.

```
# ssh admin@ipaclient.example.com
```

**NOTE**

By default, the admin user is given `/bin/bash` as the shell to use and `/home/admin` as the home directory. It may be necessary to install bash to be able to log in.

### 3.11. Troubleshooting Client Installations

For clients configured using `ipa-client-install`, the client installation log is located in `/var/log/ipaclient-install.log`. The FreeIPA logs, both for the server and client and for FreeIPA-associated services, are covered in `Section 18.1.3, “Checking FreeIPA Server Logs”`. 
The client can't resolve reverse hostnames when using an external DNS.

These are some issues and workarounds for client installation problems.

### 3.11.1. The client can't resolve reverse hostnames when using an external DNS.

While FreeIPA can host its own DNS server as part of the domain services, it can also use external DNS name server. However, because of some of the limitations of reverse DNS, there can be problems with resolving reverse lookups if the external DNS is listed in the client's `/etc/resolv.conf` file or if there are other resources on the network with SRV records, like Active Directory.

The problem is that the external DNS name server returns the wrong hostname for the FreeIPA server.

One way this exhibits is errors with finding the FreeIPA server in the Kerberos database:

```
Jun 30 11:11:49 server1 krb5kdc[1279](info): TGS_REQ (4 etypes {18 17 16 23}) 192.168.60.135: UNKNOWN_SERVER: authtime 0, admin EXAMPLE.COM for HTTP/server1.wrong.example.com@EXAMPLE.COM, Server not found in Kerberos database
```

There are several ways to work around this issue:

- Edit the `/etc/resolv.conf` file to remove the external DNS name server references.
- Add reverse lookup records for each FreeIPA server.
- Give the FreeIPA client or domain a subnet and forward all requests for that subnet.

### 3.11.2. The client is not added to the DNS zone.

If a client is in a subnet not controlled by a FreeIPA DNS server, then the `nsupdate` command may fail to add the client to the DNS zone when `ipa-client-install` runs.

If FreeIPA is managing the DNS domain, then add a zone entry for the client manually, as described in Section 9.5, “Managing DNS Record Entries”. For example:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa dnsrecord-add ipaclient.example.com www --a-rec 1.2.3.4
```

If the DNS domain is managed outside of FreeIPA, the resource record can be added manually to the zone configuration. For information on DNS in Fedora, see the DNS chapter in the Deployment Guide.

### 3.12. Uninstalling a FreeIPA Client

---

Chapter 3. Setting up Systems as FreeIPA Clients

For Fedora clients, the `ipa-client-install` utility can be used to uninstall the client and remove it from the FreeIPA domain. To remove the client, use the `--uninstall` option.

```bash
# ipa-client-install --uninstall
```

**NOTE**

There is an uninstall option with the `ipa-join` command. This is called by `ipa-client-install --uninstall` as part of the uninstallation process. However, while the `ipa-join` option removes the client from the domain, it does not actually uninstall the client or properly remove all of the FreeIPA-related configuration. Do not run `ipa-join -u` to attempt to uninstall the FreeIPA client. The only way to uninstall a client completely is to use `ipa-client-install --uninstall`. 
Basic Usage

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

4.1. About the FreeIPA Client Tools

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

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

4.1.1. About the FreeIPA Command-Line Tools

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

The command-line scripts offer a number of benefits:

• The scripts allow management tasks to be automated and performed repeatedly in a consistent way without manual intervention.

• Entries can be added with all possible attributes configured (or a desired subset of attributes) in a single step. The web UI frequently requires two steps to fully configure an entry: the first to create the entry and the next to add optional attributes.

• The command-line scripts support adding additional attributes which may not be available in the UI or even custom attributes to entries, if the schema is configured.

4.1.1.1. The Structure of the `ipa` Command

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

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

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

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

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

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

```bash
ipa help topic
```

![TIP]

**TIP**

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

```bash
ipa help
```

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

### 4.1.1.1.1. Adding, Editing, and Deleting Entries with `ipa`

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

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

For `add` operations, commands usually prompt for any required configuration attributes, which can be passed as command-line options or using `--set/addattr` options (Section 4.1.1.3, "Managing Entry Attributes with `--setattr`, `--addattr`, and `--delattr`").

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

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

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

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

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

### 4.1.1.1.2. Finding and Displaying Entries with `ipa`

Entries for an entire type are searched for using the `*-find` command and an optional search criterion. The criterion is a string which can either be an exact match or use an asterisk (*) as a wildcard.

```bash
ipa user-find *smith
```

With no search criterion, every entry of that type is displayed.
Searches (any **find** command) have certain limits imposed as part of the server configuration, specifically how many entries are returned (size limits) and how long a search will run (time limits). This is covered in Section 4.4.2, "Setting FreeIPA Search Limits". Part of the server configuration is setting global defaults for size and time limits on searches. While these limits are always enforced in the web UI, they can be overridden with any **find** command with the **--sizelimit** and **--timelimit** options. For example, if the default time limit is 60 seconds and a search is going to take longer, the time limit can be increased to 120 seconds:

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

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

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

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

### 4.1.1.1.3. Adding Members to Groups and Containers with ipa

Group members are added and removed with separate commands, apart from simply modifying an entry. Member commands essentially create a relationship between different FreeIPA entries. While this is obvious in traditional group-member roles, it is also true for some policy entries (like SELinux and sudo policies) where entries are associated with another entry.

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

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

### 4.1.1.2. Positional Elements in ipa Commands

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

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

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

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

```
ipa command parentEntryName chid1EntryName --childOptions=childValues
```

### 4.1.1.3. Managing Entry Attributes with **setattr**, **addattr**, and **delattr**

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

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

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

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

Both options have this format:

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

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

The **--addattr** option adds a new value for an attribute; for a multi-valued attribute, it adds the new value while preserving any existing values.

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

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

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

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

**NOTE**

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

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

### 4.1.1.4. Using Special Characters with FreeIPA Tools

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

4.1.1.5. Logging into the FreeIPA Domain Before Running

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

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

Different login options are described in Section 4.2, “Logging into FreeIPA”.

4.1.2. Looking at the FreeIPA UI

The FreeIPA web UI is designed for simplicity. This was the primary design goal, and this means that the web UI offers benefits that make using FreeIPA simpler and clearer:

- It shows instant, visual relationships between entries (such as a user and all the groups, sudo rules, netgroups, and policies which are associated with that user).
- All entries are listed immediately without having to run a search. This makes it possible to browse entries. The UI also has a simple search box which quickly filters the list of entries.
- The interface is intuitive to use, without having to learn the command-line tools.
- The web UI can be accessed from machines outside the FreeIPA domain, so the domain can be managed from anywhere.

Using the web UI requires a valid Kerberos ticket for the FreeIPA domain (by default), meaning that it can only be accessed from a machine within the FreeIPA domain. Alternatively, the web UI can be configured to allow password authentication along with Kerberos authentication, or a machine outside the FreeIPA can be configured to work with Kerberos (Section 4.3.4, “Using a Browser on Another System”).

4.1.2.1. The UI Layout

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

<table>
<thead>
<tr>
<th>Main Menu Tab</th>
<th>Configuration Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>• User entries</td>
</tr>
<tr>
<td></td>
<td>• User groups entries</td>
</tr>
<tr>
<td></td>
<td>• Host/client entries</td>
</tr>
<tr>
<td></td>
<td>• Host group entries</td>
</tr>
<tr>
<td></td>
<td>• Netgroups entries</td>
</tr>
<tr>
<td></td>
<td>• Domain services entries</td>
</tr>
<tr>
<td></td>
<td>• DNS (if configured)</td>
</tr>
</tbody>
</table>
The main menu at the top of every page has three tabs which correspond to the functional areas listed in Table 4.1, "Configuration Areas Per Tab". When a tab is selected, there is a submenu of the different configuration areas. Some configuration areas may have multiple possible entries; for example, role-based access controls define user roles/groups, the areas that access can be granted or denied (privileges), and then the permissions granted to those areas. Each separate configuration entry has its own task area beneath the primary configuration area.

---

<table>
<thead>
<tr>
<th>Main Menu Tab</th>
<th>Configuration Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
<td>• Host-based access control</td>
</tr>
<tr>
<td></td>
<td>• Sudo rules</td>
</tr>
<tr>
<td></td>
<td>• Automount</td>
</tr>
<tr>
<td></td>
<td>• User password policies</td>
</tr>
<tr>
<td></td>
<td>• Kerberos ticket policy</td>
</tr>
<tr>
<td>FreeIPA Server (access controls within FreeIPA)</td>
<td>• Role-based access control (permissions based on group membership)</td>
</tr>
<tr>
<td></td>
<td>• Self permissions</td>
</tr>
<tr>
<td></td>
<td>• Delegations (user access control over other users)</td>
</tr>
</tbody>
</table>

---

All entries for a configuration area are listed together on the main page for that area. This page provides direct links to individual entry pages, as well as basic information (the attributes) about the entry. (This is usually just the description, but user entries show a lot more information.)

The page also has some tasks that can be performed on it. For a list page that shows entries, this can be creating or deleting an entry. For a list page for groups, the tasks are for establishing relationships between entities, either by adding (enrolling) or removing an entity from that group. Both individual entries and groups can be searched for through the list page.
Looking at the FreeIPA UI

Each entry page is a form which allows that entry to be edited. This is done by editing text fields or by selecting items from drop-down menus.
4.1.2.2. Page Elements

The web UI uses common elements on all pages.

The most basic is that all blue text is a link to an entry or to an action.

When a task like adding an entry or saving a change is possible, the task link is blue. When it is not possible (such as no items have been selected to be deleted) then the task is grayed out.

All list pages display direct links to entry pages. However, some entries are essentially nested. For example, in automount configuration, the primary entry is the location, and then keys, mount points, and maps are associated with that location as children entries. This hierarchy is reflected in breadcrumb navigation near the top of the page, so it is easy to identify where you are in the UI and how this entry relates to any other related entries.
Most entries have a variety of different configuration areas. A simple user entry has account activity settings, personal information, address information, organizational information, and other contact information. Related attributes are grouped together logically in the UI. These entry form areas can be collapsed or expanded using the arrows to control the amount of information displayed on the page.

When entries are created, they are added with only the required attributes. Additional attributes can be added manually. Some attributes have default values added to the entry and simply need to be edited; other attributes may not exist at all in the new entry and need to be added.
Chapter 4. Basic Usage

Figure 4.7. Add an Attribute

Any changes to any attribute can be undone. A single attribute change can be undone by clicking the dynamic undo button; all changes can be undone by clicking the Reset link at the top of the entry details page.

Figure 4.8. Undo Edits

4.1.2.3. Showing and Changing Group Members

Members can be added to a group through the group configuration. There are tabs for all the member types which can belong to the group, and an administrator picks all of the marching entries and adds them as members.

However, it is also possible for an entity to be added to a group through its own configuration. Each entry has a list of tabs that displays group types that the entry can join. The list of all groups of that type are displayed, and the entity can be added to multiple groups at the same time.
4.2. Logging into FreeIPA

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

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

### 4.2.1. Logging into FreeIPA

Logging into FreeIPA requires running **kinit** on a client within the FreeIPA domain.

```
$ kinit
```

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

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

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

**NOTE**

If SSSD or pam_krb5 is configured on the FreeIPA client machine, then when a user logs into the machine, a ticket is created which can be used for machine services which require authentication, such as **sudo**.
4.2.2. Logging in When an FreeIPA User Is Different Than the System User

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

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

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

```
$ kinit admin
```

**NOTE**

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

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

4.2.3. Checking the Current Logged in User

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

```
$ klist
Ticket cache: FILE:/tmp/krb5cc_500
Default principal: ipaUser@EXAMPLE.COM
Valid starting     Expires            Service principal
11/10/08 15:35:45  11/11/08 15:35:45  krbtgt/EXAMPLE.COM@EXAMPLE.COM
Kerberos 4 ticket cache: /tmp/tkt500
klist: You have no tickets cached
```

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

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

4.2.4. Caching User Kerberos Tickets

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

For example, if you authenticated as `admin`, added a new user, set the password, and then tried to authenticate as that user, the administrator's ticket is lost.
To keep separate credential caches in different shells, a special environment variable, KRB5CCNAME, can be used.

4.3. Using the FreeIPA Web UI

In order to use the web UI, the user must be authenticated with the FreeIPA Kerberos domain and have an active Kerberos ticket (Section 4.2, “Logging into FreeIPA”). Generally, the web UI can only be accessed from a FreeIPA server or client machine and the user must be locally authenticated. There are a couple of ways to work around this, either by configuring Kerberos on a non-domain machine to connect to the Kerberos domain (Section 4.3.4, “Using a Browser on Another System”) or by password authentication to the UI.

4.3.1. Supported Web Browsers

These browsers are supported for connecting to the web UI:

- Firefox 10.x
- Firefox 3.6

4.3.2. Opening the FreeIPA Web UI

The browser must be properly configured, as described in Section 4.3.3, “Configuring the Browser”, to support Kerberos authentication so that the user can connect to the UI.

To open the web UI:

1. Get a valid Kerberos ticket using kinit, as in Section 4.2, “Logging into FreeIPA”.

2. Open the FreeIPA URL. The full URL is https://IPAserver-FQDN/ipa/ui, but this service is also accessed simply by opening https://IPAserver-FQDN. For example:

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

4.3.3. Configuring the Browser

Firefox can use Kerberos credentials to authenticate to the FreeIPA UI, but Kerberos negotiation needs to be configured to use the FreeIPA domain. At the first log-in attempt, if Firefox has not been configured to support Kerberos authentication, then an error message appears.
If you see that error, then the FreeIPA web UI can perform the required configuration:

1. Click the **follow these directions** link.
2. Click the link to import the CA certificate for the FreeIPA server.
3. Set the web site and software developer (first and last) trust bits for the CA certificate.
4. Click the **Configure Firefox** button. This automatically fills out all the **negotiate** settings in the Firefox configuration to use the FreeIPA domain settings.

When the process is complete, a success box pops up saying that Firefox has been configured for single sign-on. For there, you are redirected to the FreeIPA web UI.
This can also be done manually:

1. Open Firefox.
2. Type `about:config` in the address bar.
3. In the `Search` field, type `negotiate` to filter out the Kerberos-related parameters.
4. On Fedora, enter the domain name for the URI parameters, including the preceding period (.) and set the `gsslib` parameter to true:

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

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

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

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

5. Open the web UI by going to the fully-qualified domain name of the FreeIPA server such as `http://ipaserver.example.com`. Make sure that you can open the web UI and that there are no Kerberos authentication errors.

6. Next, download the FreeIPA server's CA certificate from `http://ipa.example.com/ipa/config/ca.crt`.

7. Select the first (`Trust this CA to identify web sites`) and third (`Trust this CA to identify software developers`) check boxes.

### 4.3.4. Using a Browser on Another System

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

This is especially useful if there are multiple realms or overlapping domains across your infrastructure.
1. Copy the `/etc/krb5.conf` file from the FreeIPA server.

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

**WARNING**

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

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

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

3. Configure Firefox on the external machine as in Section 4.3.3, "Configuring the Browser".

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

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

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

The second part of the message offers the alternative of using simple authentication. The **form-based authentication** link opens a login page.
Then simply supply the UID and password for a configured FreeIPA user.
4.3.6. Using the UI with Proxy Servers

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

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

4.3.7. Troubleshooting UI Connection Problems

If negotiate authentication is not working, turn on verbose logging for the authentication process to help diagnose the issue:

1. Close all browser windows.
2. In a terminal, set the new log levels for Firefox:

```
export NSPR_LOG_MODULES=negotiateauth:5
export NSPR_LOG_FILE=/tmp/moz.log
```
This enables verbose logging and logs all information to `/tmp/moz.log`.

3. Restart the browser from the same terminal window and attempt to.

Some of the common error messages and workarounds are in Table 4.2, "UI Error Log Messages".

<table>
<thead>
<tr>
<th>Error Log Message</th>
<th>Description and Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1208550944[90039d0]: entering nsNegotiateAuth::GetNextToken() -1208550944[90039d0]: gss_init_sec_context() failed: Miscellaneous failure No credentials cache found</td>
<td>There are no Kerberos tickets. Run <code>kinit</code>.</td>
</tr>
<tr>
<td>-1208994096[8d683d8]: entering nsAuthGSSAPI::GetNextToken() -1208994096[8d683d8]: gss_init_sec_context() failed: Miscellaneous failure Server not found in Kerberos database</td>
<td>This can occur when you have successfully obtained Kerberos tickets but are still unable to authenticate to the UI. This indicates that there is a problem with the Kerberos configuration. The first place to check is the <code>[domain_realm]</code> section in the <code>/etc/krb5.conf</code> file. Make sure that the FreeIPA Kerberos domain entry is correct and matches the configuration in the Firefox negotiation parameters. For example: <code>.example.com = EXAMPLE.COM example.com = EXAMPLE.COM</code></td>
</tr>
<tr>
<td>Nothing is in the log file.</td>
<td>It is possible that you are behind a proxy which is removing the HTTP headers required for negotiate authentication. Try to connect to the server using HTTPS instead, which allows the request to pass through unmodified. Then check the log file again.</td>
</tr>
</tbody>
</table>

### 4.4. Understanding Search Limits and Settings

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.

#### 4.4.1. Types of Search Limits and Where They Apply

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 FreeIPA server has several different limits imposed on searches:

- **The search limit configuration for the FreeIPA server.** This is a setting for the FreeIPA server itself, which is applied to all requests sent to the server from all FreeIPA clients, the FreeIPA CLI tools, and the FreeIPA web UI for normal page display.

  By default, this limit is 100 entries.
• The time limit configuration for the FreeIPA server. Much like the search size limit, the time limit sets a maximum amount of time that the FreeIPA server, itself, waits for searches to run. Once it reaches that limit, the server stops the search and returns whatever entries were returned in that time.

   By default, this limit is two seconds.

• The page size limit. Although not strictly a search limit, the page size limit does limit how many entries are returned per page. The server returns the set of entries, up to the search limit, and then randomly selects up to 20 entries per page for display. Paging results makes the results more understandable and more viewable.

   This is hard-coded to 20 for all searches.

• The LDAP search limit (--pkey option). All searches performed in the UI, and CLI searches which use the -pkey option, override the search limit set in the FreeIPA 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.

4.4.2. Setting FreeIPA 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 FreeIPA server performance.

4.4.2.1. With the Web UI

1. Open the IPA Server tab.

2. Select the Configuration subtab.

3. Scroll to the Search Options area.
4. Change the search limit settings.

   • *Search size limit*, the maximum number of records to return in a search.

   • *Search time limit*, the maximum amount of time, in seconds, to spend on a search before the server returns results.

   **TIP**

   Setting the time limit or size limit value to -1 means that there are no limits on searches.

5. When the changes are complete, click the *Update* link at the top of the *Configuration* page.
4.4.2.2. With the Command Line

The search limits can be changed using the `config-mod` command.

```
$ ipa config-mod --searchtimelimit=5 --searchrecordslimit=500
```

Max. username length: 32
Home directory base: /home
Default shell: /bin/sh
Default users group: ipausers
Default e-mail domain for new users: rhts.eng.bos.redhat.com
Search time limit: 5
Search size limit: 50
User search fields: uid,givenname,sn,telephonenumber,ou,title
Group search fields: cn,description
Enable migration mode: FALSE
Certificate Subject base: O=EXAMPLE.COM
Password Expiration Notification (days): 4

TIP

Setting the time limit or size limit value to -1 means that there are no limits on searches.

4.4.3. 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 *sen --timelimit=120
```

4.4.4. 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`

4.4.4.1. Setting User Search Attributes

4.4.4.1.1. From the Web UI
1. Open the IPA Server tab.
2. Select the Configuration subtab.
3. Scroll to the User Options area.

4. Add any additional search attributes, in a comma-separated list, in the User search fields field.
5. When the changes are complete, click the Update link at the top of the Configuration page.

4.4.4.1.2. From the Web UI
To change the search attributes, use the --usersearch option to set the attributes for user searches.
4.4.4.2. Setting 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`.

4.4.4.2.1. From the Web UI

1. Open the **IPA Server** tab.
2. Select the **Configuration** subtab.
3. Scroll to the **Group Options** area.

---

$ ipa config-mod --usersearch=uid,givenname,sn,telephonenumber,ou,title

**NOTE**

Always give the complete list of search attributes. Whatever values are passed with the configuration argument overwrite the previous settings.

4. Add any additional search attributes, in a comma-separated list, in the Group search fields field.

5. When the changes are complete, click the Update link at the top of the Configuration page.
4.4.4.2. From the Command Line
To change the search attributes, use the `--groupsearch` options to set the attributes for group searches.

```
$ ipa config-mod --groupsearch=cn,description
```

**NOTE**

Always give the complete list of search attributes. Whatever values are passed with the configuration argument overwrite the previous settings.

4.4.5. 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.
Chapter 5.

Identity: Managing Users and User Groups

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

5.1. Setting up User Home Directories

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

5.1.1. About Home Directories

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

- The default prefix for users' home directories is /home.
- FreeIPA does not automatically create home directories when users log in. Automatically creating home directories requires either the pam_oddjob_mkhomedir module or the pam_mkhomedir module. This module can be configured as part of client installation or after installation, as described in Section 5.1.2, “Enabling the PAM Home Directory Module”.

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

**NOTE**

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

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

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

- Use automount to mount only the user's home directory and only when the user logs in, rather than loading the entire /home tree.
Chapter 5. Identity: Managing Users and User Groups

- Use a remote user who has limited permissions to create home directories and mount the share on the FreeIPA server as that user. Since the FreeIPA server runs as an httpd process, it is possible to use sudo or a similar program to grant limited access to the FreeIPA server to create home directories on the NFS server.

- Use a mechanism, such as the pam_oddjob_mkhomedir module, to create the home directory as that user.

Using automounts for home directories is described in Section 5.1.3, "Manually Mounting Home Directories".

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

5.1.2. Enabling the PAM Home Directory Module

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

**NOTE**

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

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

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

- The --mkhomedir option can be used with the ipa-client-install command. While this is possible for clients, this option is not available to servers when they are set up.

- The pam_oddjob_mkhomedir module can be enabled using the system's authconfig command. For example:

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

This option can be used for both server and client machines post-installation.
5.1.3. Manually Mounting Home Directories

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

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

   $ ipa automountlocation-add userdirs
   Location: userdirs

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

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

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

Using automounts with FreeIPA is described in detail in Chapter 10, Policy: Using Automount.

5.2. Managing User Entries

5.2.1. About Username Formats

The default length for usernames is 32 characters.

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

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

TIP

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

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

5.2.2.1. From the Web UI
1. Open the Identity tab, and select the Users subtab.

2. Click the Add link at the top of the users list.

3. Fill in the user’s first and last names. The user login (UID) is automatically generated based on the user’s full name, but this can be set manually by clicking the Optional field link.
4. Click the Add and Edit button to go directly to the expanded entry page and fill in more attribute information, as in Section 5.2.3.1, “From the Web UI”. The user entry is created with some basic information already filled in, based on the given user information and the user entry template.
Chapter 5. Identity: Managing Users and User Groups

USER: jsmith

jsmith is a member of:

IDENTITY SETTINGS

Job Title: 

First name:  * John

Last name:  * Smith

Full name:  * John Smith

Display name:  John Smith

Initials:  JS

ACCOUNT SETTINGS

Status:  Enabled: Click to Disable

User login:  jsmith

Password:  Reset Password

UID:  * 172200003

GID:  

Refresh  Reset  Update
5.2.2.2. From the Command Line

New user entries are added with the `user-add` command. Attributes (listed in Table 5.2, “Default FreeIPA User Attributes”) can be added to the entry with specific values or the command can be run with no arguments.

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

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

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

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

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

**IMPORTANT**

When a user is created without specifying a UID or GID number, then the user account is automatically assigned an ID number that is next available in the server or replica range. (Number ranges are described more in Section 5.7, “Managing Unique UID and GID Number Assignments”) This means that a user always has a unique number for its UID number and, if configured, for its private group.

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

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

5.2.3. Editing Users

5.2.3.1. From the Web UI

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

3. There are a number of different types of attributes that can be edited for the user. All of the default attributes are listed in Table 5.2, “Default FreeIPA User Attributes”. Most of the attributes in the Identity Settings and Account Settings areas have default values filled in for them, based on the user information or on the user entry template.
4. Edit the fields or, if necessary, click the **Add** link by an attribute to create the attribute on the entry.
Chapter 5. Identity: Managing Users and User Groups

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

5.2.3.2. From the Command Line

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

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

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

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

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

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

**Example 5.1. Multiple Mail Attributes**

A user is created first using his work email account.

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

Then, his personal email account is added.

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

Both email addresses are listed for the user.

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

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

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

5.2.4. Activating and Deactivating User Accounts

User accounts can be deactivated. A deactivated user cannot log into FreeIPA or its related services (like Kerberos) and he cannot perform any tasks. However, the user account still exists within FreeIPA and all of the associated information remains unchanged.
Any existing connections remain valid until the Kerberos TGT and other tickets expire. Once the ticket expires, the user cannot renew the ticket.

5.2.4.1. From the Web UI

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

2. Click the name of the user for whom to deactivate or activate.

3. Scroll to the **Account Settings** area.

4. Click the **Deactivate** link.
5. Click the **Update** link at the top of the page.

5.2.4.2. From the Command Line

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

```bash
$ ipa user-disable jsmith
```

5.2.5. Deleting Users

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

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

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

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

5.2.5.1. With the Web UI

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

2. Select the checkboxes by the names of the users to delete.

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

4. When prompted, confirm the delete action.
5.2.5.2. From the Command Line

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

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

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

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

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

5.3. Managing Public SSH Keys for Users

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

There are a couple of problems with this system:

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

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

User SSH keys are added to user entries in FreeIPA, either when the user is created or by modifying the entry later. In a key file, such as a user's `id_rsa.pub` file, a key entry is identified by its type and then the key itself. For example, for an RSA key:

```
ssh-rsa ABCD1234==
```

Only the second part, the base 64-encoded key itself, is uploaded to the user entry.

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

```
[jsmith@server ~]$ ipa user-mod jsmith --sshpubkey="12345abcde="
```

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

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

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

After uploading the user keys, configure SSSD to use FreeIPA as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing user keys.

### 5.4. Changing Passwords

Password policies ([Chapter 11, Policy: Defining Password Policies](#)) and minimal access restrictions can be applied to a password change operation:

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

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

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

- Changing a password as the LDAP Directory Manager user, using LDAP tools, overrides any FreeIPA password policies.

### 5.4.1. From the Web UI

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

2. Click the name of the user for whom to reset the password. All users can change their own password; only administrators or users with delegated permissions can change other user's passwords.
3. Scroll to the **Account Settings** area.

4. Click the **Reset Password** link.
5. In the pop-up box, enter and confirm the new password.

5.4.2. From the Command Line

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

```
[bjensen@ipaserver ~]$ kinit admin
```
5.5. Unlocking User Accounts After Password Failures

If a user attempts to log in and uses the wrong password a certain number of times, then that user account is locked. The exact number of failed attempts that locks and account and the duration of the lockout is defined as part of the password policy (Section 11.6, “Setting Account Lockout Policies”).

A password policy can implicitly define a reset period, where the account unlocks naturally after a certain amount of time lapses. However, if the duration is fairly long or if the deployment requires stronger security checks before unlocking an account, then an administrator can unlock an account manually.

An account is unlocked using the `user-unlock` command. For example:

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

5.6. Managing User Private Groups

On Fedora 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 FreeIPA domain, it is also created with a corresponding private group, following the Fedora 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 assigned to NIS groups or other system groups.

5.6.1. Disabling Private Groups for a Specific User

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

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

5.6.2. Disabling Private Groups Globally

User private groups are managed through the Managed Entries Plug-in in 389 Directory Server. This plug-in can be disabled, which effectively disables private group creation for all new users.

This is done using the `ipa-managed-entries` command.

1. Use the `ipa-managed-entries` command to list possible Managed Entries Plug-in definitions. By default, there are two, one for new users (UPG) and one for netgroups (NGP).

```
[root@ipaserver ~]# ipa-managed-entries --list -p DMpassword
Available Managed Entry Definitions:
  UPG Definition
  NGP Definition
```

---

1 See Section 5.7, “Managing Unique UID and GID Number Assignments” for information on changing GID/UID assignment ranges.
2. Disable the desired Managed Entries Plug-in instance. For example:

```
[root@ipaserver ~]# ipa-managed-entries -e "UPG Definition" -p DMpassword disable
Disabling Plugin
```

3. Restart the 389 Directory Server to load the new plug-in configuration.

```
[root@ipaserver ~]# service dirsrv restart
```

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

## 5.7. Managing Unique UID and GID Number Assignments

A FreeIPA server must generate random UID and GID values and simultaneously ensure that replicas never generate the same UID or GID value. The need for unique UID and GID numbers might even cross FreeIPA domains, if a single organization has multiple disparate domains.

The UID and GID numbers are divided into **ranges**. By keeping separate numeric ranges for individual servers and replicas, the chances are minimal that any numbers issued by one server or replica will duplicate those from another. Ranges are updated and shared intelligently between servers and replicas through the Dynamic Numeric Assignment (DNA) Plug-in, as part of the backend 389 Directory Server instance for the domain. The same range is used for user IDs (`uidNumber`) and group IDs (`gidNumber`). A user and a group may have the same ID, but since the ID is set in different attributes, there is no conflict. Using the same ID number for both a user and a group also allows an administrator to configure user private groups, where a unique system group is created for each user and the ID number is the same for both the user and the group.

**IMPORTANT**

When a user is created interactively or without specifying a UID or GID number, then the user account is created with an ID number that is next available in the server or replica range. This means that a user always has a unique number for its UID number and, if configured, for its private group.

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

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

### 5.7.1. About ID Range Assignments During Installation

The FreeIPA administrator can initially define a range during server installation, using the `--idstart` and `--idmax` options with `ipa-server-install`. These options are not required, so the setup script can assign random ranges during installation.

If no range is set manually when the first FreeIPA server is installed, a range of 200,000 IDs is randomly selected. There are 10,000 possible ranges. Selecting a random range from that number
provides a high probability of non-conflicting IDs if two separate FreeIPA domains are ever merged in the future.

With a single FreeIPA server, IDs are assigned to entries in order through the range. With replicas, the initial server ID range is split and distributed.

When a replica is installed, it is configured with an invalid range. It also has a directory entry (that is shared among replicas) that instructs the replica where it can request a valid range. When the replica starts, or as its current range is depleted so that less than 100 IDs are available, it can contact one of the available servers for a new range allotment. A special extended operation splits the range in two, so that the original server and the replica each have half of the available range.

5.7.2. Adding New Ranges
If the range for the entire domain is close to depletion, a new range can be manually selected and assigned to one of the master servers. All replicas then request ID ranges from the master as necessary.

The changes to the range are done by editing the 389 Directory Server configuration to change the DNA Plug-in instance. The range is defined in the dnaNextRange parameter. For example:

```
ldapmodify -x -D "cn=Directory Manager" -W -h server.example.com -p 389
Enter LDAP Password: *******
dn: cn=Posix IDs,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
changetype: modify
add: dnaNextRange
dnaNextRange: 123400000-123500000
```

**NOTE**
This command only adds the specified range of values; it does not check that the values in that range are actually available. This check is performed when an attempt is made to allocate those values. If a range is added that contains mostly values that were already allocated, the system will cycle through the entire range searching for unallocated values, and then the operation ultimately fails if none are available.

5.8. Managing User and Group Schema
When a user entry is created, it is automatically assigned certain LDAP object classes which, in turn, make available certain attributes. LDAP attributes are the way that information is stored in the directory. (This is discussed in detail in the Directory Server Deployment Guide and the Directory Server Schema Reference.)

<table>
<thead>
<tr>
<th>Description</th>
<th>Object Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeIPA object classes</td>
<td>ipaobject</td>
</tr>
<tr>
<td>Person object classes</td>
<td>person</td>
</tr>
<tr>
<td></td>
<td>organizationalperson</td>
</tr>
<tr>
<td></td>
<td>inetorgperson</td>
</tr>
<tr>
<td></td>
<td>inetuser</td>
</tr>
<tr>
<td></td>
<td>posixaccount</td>
</tr>
<tr>
<td>Kerberos object classes</td>
<td>krbprincipalaux</td>
</tr>
</tbody>
</table>
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 5.1, “Default FreeIPA 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 5.11, “Specifying Default User and Group Settings”.

<table>
<thead>
<tr>
<th>UI Field</th>
<th>Command-Line Option</th>
<th>Required, Optional, or Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>User login</td>
<td>username</td>
<td>Required</td>
</tr>
<tr>
<td>First name</td>
<td>--first</td>
<td>Required</td>
</tr>
<tr>
<td>Last name</td>
<td>--last</td>
<td>Required</td>
</tr>
<tr>
<td>Full name</td>
<td>--cn</td>
<td>Optional</td>
</tr>
<tr>
<td>Display name</td>
<td>--displayname</td>
<td>Optional</td>
</tr>
<tr>
<td>Initials</td>
<td>--initials</td>
<td>Default</td>
</tr>
<tr>
<td>Home directory</td>
<td>--homedir</td>
<td>Default</td>
</tr>
<tr>
<td>GECOS field</td>
<td>--gecos</td>
<td>Default</td>
</tr>
<tr>
<td>Shell</td>
<td>--shell</td>
<td>Default</td>
</tr>
<tr>
<td>Kerberos principal</td>
<td>--principal</td>
<td>Default</td>
</tr>
<tr>
<td>Email address</td>
<td>--email</td>
<td>Optional</td>
</tr>
<tr>
<td>Password</td>
<td>--password</td>
<td>Optional. <em>Unlike the other options, this accepts no value. The script prompts for the new password.</em></td>
</tr>
<tr>
<td>User ID number²</td>
<td>--uid</td>
<td>Default</td>
</tr>
<tr>
<td>Group ID number²</td>
<td>--gidnumber</td>
<td>Default</td>
</tr>
<tr>
<td>Street address</td>
<td>--street</td>
<td>Optional</td>
</tr>
<tr>
<td>City</td>
<td>--city</td>
<td>Optional</td>
</tr>
<tr>
<td>State/Province</td>
<td>--state</td>
<td>Optional</td>
</tr>
<tr>
<td>Zip code</td>
<td>--postalcode</td>
<td>Optional</td>
</tr>
<tr>
<td>Telephone number</td>
<td>--phone</td>
<td>Optional</td>
</tr>
<tr>
<td>Mobile telephone number</td>
<td>--mobile</td>
<td>Optional</td>
</tr>
<tr>
<td>Pager number</td>
<td>--pager</td>
<td>Optional</td>
</tr>
<tr>
<td>Fax number</td>
<td>--fax</td>
<td>Optional</td>
</tr>
<tr>
<td>Organizational unit</td>
<td>--orgunit</td>
<td>Optional</td>
</tr>
<tr>
<td>Job title</td>
<td>--title</td>
<td>Optional</td>
</tr>
<tr>
<td>Manager</td>
<td>--manager</td>
<td>Optional</td>
</tr>
<tr>
<td>Car license</td>
<td>--carlicense</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>--noprivate</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Chapter 5. Identity: Managing Users and User Groups

<table>
<thead>
<tr>
<th>UI Field</th>
<th>Command-Line Option</th>
<th>Required, Optional, or Default¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH Keys</td>
<td>--sshpubkey</td>
<td>Optional</td>
</tr>
<tr>
<td>Additional attributes</td>
<td>--addattr</td>
<td>Optional</td>
</tr>
</tbody>
</table>

¹ 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.

² When a user is created without specifying a UID number, then the user account is automatically assigned an ID number that is next available in the server or replica range. (Number ranges are described more in Section 5.7, “Managing Unique UID and GID Number Assignments”.) This means that a user always has a unique number for its UID number and, if configured, for its private group.

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

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

5.8.1. About Changing the Default User and Group Schema

It is possible to add or change the object classes and attributes used for user and group entries (Section 5.8, “Managing User and Group Schema”).

The FreeIPA 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 FreeIPA schema validation, however. Most important, the FreeIPA server does not check that the defined user or group object classes contain all of the required object classes for FreeIPA entries. For example, all FreeIPA 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.

5.8.2. 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 FreeIPA-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.

5.8.2.1. From the Web UI

1. Add all of the custom schema elements to the 389 Directory Server instance used by FreeIPA. Adding schema elements is described in the `schema chapter of the Directory Server Administrator's Guide`².
Applying Custom Object Classes to New User Entries

2. Open the **IPA Server** tab.

3. Select the **Configuration** subtab.

4. Scroll to the **User Options** area.

   ![Configuration Form](image)

5. At the bottom of the users area, click the **Add** link to add a new field for another object class.

   **IMPORTANT**

   Always include the *existing* default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by FreeIPA are not included, then subsequent attempts to add an entry will fail with object class violations.
6. When the changes are complete, click the Update link at the top of the Configuration page.

5.8.2.2. From the Command Line

1. Add all of the custom schema elements to the 389 Directory Server instance used by FreeIPA. Adding schema elements is described in the schema chapter of the Directory Server Administrator’s Guide.

2. Add the new object class to the list of object classes added to entries. The option for user object classes is --userobjectclasses.

**IMPORTANT**

Always include the existing default object classes when the configuration is updated. Otherwise, the current settings will be overwritten. If any object classes required by FreeIPA are not included, then subsequent attempts to add an entry will fail with object class violations.

For example:

```
$ ipa config-mod --
userobjectclasses=top, person, organizationalperson, inetorgperson, inetuser, posixaccount, krbprincipalaux, krbticketpolicyaux, ipaobject, employeeinfo
```
5.8.3. 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 FreeIPA server configuration.

5.8.3.1. From the Web UI
1. Add all of the custom schema elements to the 389 Directory Server instance used by FreeIPA. 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.
5. Click the **Add** link to add a new field for another object class.
5.8.3.2. From the Command Line

1. Add all of the custom schema elements to the 389 Directory Server instance used by FreeIPA. 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`.

For example:

```bash
$ ipa config-mod --
  groupobjectclasses=top,groupofnames,nestedgroup,ipausergroup,ipaobject,employeegroup
```

5.9. Managing User Groups

User groups are a way of centralizing control over important management tasks, particularly access control and password policies. Three groups are created during the installation, specifically for use by FreeIPA operations:

- **ipausers**, which contains all users.
- **admins**, which contains administrative users. The initial `admin` user belongs to this group.
- **editors**, which is a special group for users working through the web UI. This group allows users to **edit** other users' entries, though without all of the rights of the admin user.

All groups in FreeIPA are essentially static groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, FreeIPA 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 FreeIPA or infrastructure usage guidelines for access controls.

**NOTE**

Some operating systems limit the number of groups that can be assigned to system users. For example, Solaris and AIX systems both limit users to 16 groups per user. This can be an issue when using nested groups, when a user may be automatically added to multiple groups.

When a group entry is created, it is automatically assigned certain LDAP object classes. (LDAP object classes and attributes are discussed in detail in the *Directory Server Deployment Guide* and the *Directory Server Schema Reference.*) For groups, only two attributes truly matter: the name and the description.

**Table 5.3. Default FreeIPA Group Object Classes**

<table>
<thead>
<tr>
<th>Description</th>
<th>Object Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeIPA object classes</td>
<td>ipaobject</td>
</tr>
<tr>
<td></td>
<td>ipausergroup</td>
</tr>
<tr>
<td></td>
<td>nestedgroup</td>
</tr>
<tr>
<td>Group object classes</td>
<td>groupofnames</td>
</tr>
<tr>
<td></td>
<td>posixgroup</td>
</tr>
</tbody>
</table>

**5.9.1. Creating User Groups**

**5.9.1.1. With the Web UI**

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Click the **Add** link at the top of the groups list.
3. Enter all of the information for the group.
Chapter 5. Identity: Managing Users and User Groups

- A unique name. This is the identifier used for the group in the FreeIPA 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 FreeIPA 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, FreeIPA 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 5.9.2.1, “With the Web UI (Group Page)”**.

### 5.9.1.2. With the Command Line

New groups are created using the `group-add` command. (This adds only the group; members are added separately.)

Two attributes are always required: the group name and the group description. If those attributes are not given as arguments, then the script prompts for them.

```
$ ipa group-add groupName --desc="description" [--nonposix]
```

Additionally, there is one other configuration option, **--nonposix**. (By default, all groups are created as POSIX groups.) To enable interoperability with Windows users and groups and programs like Samba, it is possible to create non-POSIX groups by using the **--nonposix** option. This option tells the script not to add the `posixGroup` object class to the entry.

For example:

```
$ ipa group-add examplegroup --desc="for examples" --nonposix
```

When no arguments are used, the command prompts for the required group account information:

```
$ ipa group-add
Group name: engineering
Description: for engineers
----------
Added group "engineering"
----------
Group name: engineering
Description: for engineers
GID: 387115842
```
Adding Group Members

121

IMPORTANT

When a group is created without specifying a GID number, then the group entry is assigned the ID number that is next available in the server or replica range. (Number ranges are described more in Section 5.7, “Managing Unique UID and GID Number Assignments”.) This means that a group always has a unique number for its GID number.

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

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

NOTE

You cannot edit the group name. The group name is the primary key, so changing it is the equivalent of deleting the group and creating a new one.

5.9.2. Adding Group Members

5.9.2.1. With the Web UI (Group Page)

NOTE

This procedure adds a user to a group. User groups can contain other user groups as their members. These are nested groups.

It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

When creating nested groups, be careful not to create recursive groups. For example, if GroupA is a member of GroupB, do not add GroupB as a member of GroupA. Recursive groups are not supported and can cause unpredictable behavior.

1. Open the Identity tab, and select the User Groups subtab.
2. Click the name of the group to which to add members.
3. Click the **Add** link at the top of the task area.

4. Click the checkbox by the names of the users to add, and click the right arrows button, >>, to move the names to the selection box.
5. Click the **Add** button.

Group members can be users or other user groups. It can take up to several minutes for the members of the child group to show up as members of the parent group. This is especially true on virtual machines where the nested groups have more than 500 members.

**5.9.2.2. With the Web UI (User's Page)**

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

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

2. Click the name of the user to edit.
3. Open the **User Groups** tab on the user entry page.

4. Click the **Add** link at the top of the task area.
5. Click the checkbox by the names of the groups for the user to join, and click the right arrows button, >>, to move the groups to the selection box.
6. Click the **Add** button.

### 5.9.2.3. With the Command Line
Members are added to a group using the `group-add-member` command. This command can add both users as group members and other groups as group members.

The syntax of the `group-add-member` command requires only the group name and a comma-separated list of users to add:

```
$ ipa group-add-member groupName [--users=list] [--groups=list]
```

For example, this adds three users to the `engineering` group:

```
$ ipa group-add-member engineering --users=jsmith,bjensen,mreynolds
Group name: engineering
Description: for engineers
GID: 387115642
Member users: jsmith,bjensen,mreynolds
Number of members added 3
```
Likewise, other groups can be added as members, which creates nested groups:

```
$ ipa group-add-member engineering --groups=dev,qe1,dev2
Group name: engineering
   Description: for engineers
   GID: 387115842
   Member groups: dev,qe1,dev2
   Number of members added 3
```

When displaying nested groups, members are listed as members and the members of any member groups are listed as indirect members. For example:

```
$ ipa group-show examplegroup
Group name: examplegroup
   Description: for examples
   GID: 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.

```
$ ipa group-remove-member engineering --users=jsmith
Group name: engineering
   Description: for engineers
   GID: 855800009
   Member users: bjensen,mreynolds
   Number of members removed 1
```

### 5.9.2.4. 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 FreeIPA 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.

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

**5.9.3. 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.

---

**5.9.3.1. With the Web UI**

1. Open the **Identity** tab, and select the **User Groups** subtab.
2. Select the checkbox by the name of the group to delete.
3. Click the **Delete** link at the top of the task area.

4. When prompted, confirm the delete action.

### 5.9.3.2. With the Command Line

The `group-del` command to deletes the specified group. For example:
5.10. Searching for Users and Groups

The user searches in FreeIPA 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 FreeIPA configuration, as in Section 5.11, “Specifying Default User and Group Settings”.

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 5.4, “Default Search Attributes”. All search attributes are searched in a user/group search.

Table 5.4. Default Search Attributes

<table>
<thead>
<tr>
<th>User Search Attributes</th>
<th>Group Search Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>First name</td>
<td>Name</td>
</tr>
<tr>
<td>Last name</td>
<td>Description</td>
</tr>
<tr>
<td>Login ID</td>
<td>Description</td>
</tr>
<tr>
<td>Job title</td>
<td></td>
</tr>
<tr>
<td>Organizational unit</td>
<td></td>
</tr>
<tr>
<td>Phone number</td>
<td></td>
</tr>
</tbody>
</table>

The attributes which are searched in user and group searches can be changed, as described in Section 4.4.4, “Setting Search Attributes” and Section 4.4.4.2, “Setting Group Search Attributes”.

5.10.1. With the UI

Both user and group main pages have a search bar in the upper right corner of the task area. This search box searches against all of the fields listed in Table 5.4, “Default Search Attributes”. Type in any string, even a single letter, and click the magnifying glass icon. The UI filters the displayed list to match the search string.

Figure 5.2. User Search Box

5.10.2. With the Command Line

Searches are simple:
There are a few general rules with searches:

- If there is no string, then the search returns every entry in FreeIPA, up to the search limit.
- With the command-line tools, only a single search string can be used for user and group searches. With the UI, multiple strings can be used.
- Searches are case insensitive.
- Search results are displayed alphabetically, with exact matches listed first, followed by partial matches.
- Wildcards cannot be used in searches. The search string must include at least one character that appears in one of the indexed search fields.

**Example 5.2. User Search for John**
The basic search looks for the string *john*, which can appear in any of the search indexes.

```
$ ipa user-find john
--------------
2 users matched
--------------
User login: jpeterson
First name: john
Last name: peterson
Home directory: /home/jpeterson
Login shell: /bin/sh
UID: 855800007
GID: 855800007
Account disabled: False
User login: jsmith
First name: john
Last name: smith
Home directory: /home/jsmith
Login shell: /bin/sh
UID: 855800004
GID: 855800004
Account disabled: False
--------------
Number of entries returned 2
--------------
```

A search can also accept options like `--raw`. `--raw` prints the LDAP attributes for the user account rather than the reading-friendly field names.

```
$ ipa user-find john --raw
--------------
2 users matched
--------------
uid: jpeterson
givenname: john
sn: peterson
homedirectory: /home/jpeterson
loginshell: /bin/sh
uidnumber: 855800007
gidnumber: 855800007
```
Chapter 5. Identity: Managing Users and User Groups

5.11. Specifying Default User and Group Settings

FreeIPA uses a template when it creates new entries.

For users, the template is very specific. FreeIPA uses default values for several core attributes for FreeIPA 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 FreeIPA server, `cn=ipaconfig,cn=etc,dc=example,dc=com`.

The configuration can be changed using the `ipa config-mod` command.

Table 5.5. Default User Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Command-Line Option</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum username length</td>
<td>--maxusername</td>
<td>Sets the maximum number of characters for usernames. The default value is eight.</td>
</tr>
<tr>
<td>Root for home directories</td>
<td>--homedirectory</td>
<td>Sets the default directory to use for user home directories. The default value is <code>/home</code>.</td>
</tr>
<tr>
<td>Default shell</td>
<td>--defaultshell</td>
<td>Sets the default shell to use for users. The default value is <code>/bin/sh</code>.</td>
</tr>
<tr>
<td>Default user group</td>
<td>--defaultgroup</td>
<td>Sets the default group to which all newly created accounts are added. The default value is <code>ipausers</code>, which is automatically created during</td>
</tr>
<tr>
<td>Field</td>
<td>Command-Line Option</td>
<td>Descriptions</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default e-mail domain</td>
<td>--emaildomain</td>
<td>Sets the email domain to use to create email addressed based on the new accounts. The default is the FreeIPA server domain.</td>
</tr>
<tr>
<td>Search time limit</td>
<td>--searchtimelimit</td>
<td>Sets the maximum amount of time, in seconds, to spend on a search before the server returns results.</td>
</tr>
<tr>
<td>Search size limit</td>
<td>--searchrecordslimit</td>
<td>Sets the maximum number of records to return in a search.</td>
</tr>
<tr>
<td>User search fields</td>
<td>--usersearch</td>
<td>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.</td>
</tr>
<tr>
<td>Group search fields</td>
<td>--groupsearch</td>
<td>Sets the fields in a group entry that can be used as a search string.</td>
</tr>
<tr>
<td>Certificate subject base</td>
<td></td>
<td>Sets the base DN to use when creating subject DNs for client certificates. This is configured when the server is set up.</td>
</tr>
<tr>
<td>Default user object classes</td>
<td>--userobjectclasses</td>
<td>Sets a list of object classes that are used to create FreeIPA user accounts.</td>
</tr>
<tr>
<td>Default group object classes</td>
<td>--groupobjectclasses</td>
<td>Sets a list of object classes that are used to create FreeIPA group accounts.</td>
</tr>
<tr>
<td>Password expiration notification</td>
<td>--pwdexpnotify</td>
<td>Sets how long, in days, before a password expires for the server to send a notification.</td>
</tr>
<tr>
<td>Password plug-in features</td>
<td></td>
<td>Sets the format of passwords that are allowed for users.</td>
</tr>
</tbody>
</table>

5.11.1. Viewing Settings from the Web UI

1. Open the IPA Server tab.

2. Select the Configuration subtab.

3. The complete configuration entry is shown in three sections, one for all search limits, one for user templates, and one for group templates.
5.11.2. Viewing Settings from the Command Line

The `config-show` command shows the current configuration which applies to all new user accounts. By default, only the most common attributes are displayed; use the `-all` option to show the complete configuration.

```
# ipa config-show --all

dn: cn=ipaconfig,cn=etc,dc=example,dc=com
Max. username length: 32
Home directory base: /home
Default shell: /bin/sh
Default users group: ipausers
Default e-mail domain for new users: example.com
Search time limit: 2
Search size limit: 100
User search fields: uid,givenname,sn,telephonenumber,ou,title
```
<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group search fields</td>
<td>cn, description</td>
</tr>
<tr>
<td>Enable migration mode</td>
<td>FALSE</td>
</tr>
<tr>
<td>Certificate Subject base</td>
<td>O=EXAMPLE.COM</td>
</tr>
<tr>
<td>Default group objectclasses</td>
<td>top, groupofnames, nestedgroup, ipausergroup, ipaobject</td>
</tr>
<tr>
<td>Default user objectclasses</td>
<td>top, person, organizationalperson, inetorgperson, inetuser, posixaccount, krbprincipalaux, krbticketpolicyaux, ipaobject</td>
</tr>
<tr>
<td>Password Expiration Notification (days)</td>
<td>4</td>
</tr>
<tr>
<td>Password plugin features</td>
<td>AllowNHash</td>
</tr>
<tr>
<td>cn</td>
<td>ipaConfig</td>
</tr>
<tr>
<td>objectclass</td>
<td>nsContainer, top, ipaGuiConfig, ipaConfigObject</td>
</tr>
</tbody>
</table>
Identity: Managing Hosts and Services

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 FreeIPA domain and allow it to identify the FreeIPA server it will connect with. After the initial configuration, FreeIPA has tools to manage both of these services in response to changes in the domain services, changes to the IT environment, or changes on the machines themselves which affect Kerberos, certificate, and DNS services, like changing the client hostname.

This chapter describes how to manage identity services that relate directly to the client machine:

- DNS entries and settings
- Machine authentication
- Hostname changes (which affect domain services)

6.1. 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 FreeIPA 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 FreeIPA:

- 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 FreeIPA domain. Any service that can store a Kerberos principal or an SSL certificate (or both) can be configured as an FreeIPA service. Adding a service to the FreeIPA 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.)

A FreeIPA 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. A FreeIPA domain provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

Machines are treated as another identity that is managed by FreeIPA. Clients use DNS to identify FreeIPA servers, services, and domain members — which, like user identities are stored in the 389 Directory Server instance for the FreeIPA 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 FreeIPA includes machines as well as users. Machine authentication is required for the FreeIPA server to trust the machine and to accept FreeIPA connections from the client software installed on that machine. After authenticating the client, the FreeIPA server can respond to its requests. FreeIPA 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 FreeIPA as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in FreeIPA. This is described in Section 6.8, “Managing Public SSH Keys for Hosts” and the Fedora Deployment Guide.

- Key tables (or keytabs, a symmetric key resembling to some extent a user password) and machine certificates. Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. Initially granting a Kerberos ticket, renewing the Kerberos credentials, and even destroying the Kerberos session are all handled by the FreeIPA services. Managing Kerberos is covered in Chapter 12, Policy: Managing the Kerberos Domain.

- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the FreeIPA server’s certificate authority and then stored in FreeIPA’s Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called certmonger, which is described in Appendix B, Working with certmonger.

### 6.2. Adding Host Entries

A host entry is always created when a client is configured. On Fedora systems, this is done automatically with the *ipa-client-install* script. On other platforms — and in alternative enrollment scenarios, as in Section 6.3, “Enrolling Clients Manually” — the host entry is created manually.

#### 6.2.1. Adding Host Entries from the Web UI

1. Open the Identity tab, and select the Hosts subtab.
2. Click the Add link at the top of the hosts list.
3. Fill in the machine name and select the domain from the configured zones in the drop-down list. If the host has already been assigned a static IP address, then include that with the host entry so that the DNS entry is fully created.
DNS zones can be created in FreeIPA, which is described in Section 9.4.1, “Adding DNS Zones.” If the FreeIPA server does not manage the DNS server, the zone can be entered manually in the menu area, like a regular text field.

**NOTE**

Select the **Force** checkbox to add the host DNS record, even if the hostname cannot be resolved.

This is useful for hosts which use DHCP and do not have a static IP address. This essentially creates a placeholder entry in the FreeIPA 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.
Adding Host Entries from the Command Line

Host entries are created using the `host-add` command. This commands adds the host entry to the FreeIPA Directory Server. The full list of options with `host-add` are listed in Section C.4, "ipa Host Commands". At its most basic, an add operation only requires the client hostname to add the client to the Kerberos realm and to create an entry in the FreeIPA LDAP server:

```
$ ipa host-add client1.example.com
```

If the FreeIPA 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 6.1. Creating Host Entries with Static IP Addresses**

```
$ ipa host-add --force --ip-address=192.168.166.31 client1.example.com
```
Commonly, hosts may not have a static IP address or the IP address may not be known at the time the client is configured. For example, laptops may be preconfigured as FreeIPA 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 FreeIPA 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 6.2. Creating Host Entries with DHCP
$ ipa host-add --force client1.example.com
```

Host records are deleted using the `host-del` command. If the FreeIPA 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
```

### 6.3. Enrolling Clients Manually

Enrolling machines as clients in the FreeIPA 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 FreeIPA 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 FreeIPA generate a random one.

### 6.3.1. Performing a Split Enrollment

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 6.2, “Adding Host Entries”.
2. The second administrator installs the FreeIPA client packages on the machine, as in Section 3.4, “Configuring a Fedora System as a FreeIPA 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:
Manually Unconfiguring Client Machines

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 FreeIPA domain. The keytab is saved with root:root ownership and 0600 permissions.

6.4. Manually Unconfiguring Client Machines

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

ipa-client-install --uninstall

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

**WARNING**

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

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

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

   To remove specific principals:

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

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

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

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

   $ ipa host-del server.example.com

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

   $ ipa-join
### 6.5. Managing Services

#### 6.5.1. 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 FreeIPA 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, FreeIPA 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 FreeIPA 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.

#### 6.5.1.1. Adding Services and Keytabs from the Web UI

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

2. Click the **Add** link at the top of the services list.
3. Select the service type from the drop-down menu, and give it a name.
4. Select the hostname of the FreeIPA host on which the service is running. The hostname is used to construct the full service principal name.
5. Click the **Add** button to save the new service principal.

6. Use the **ipa-getkeytab** command to generate and assign the new keytab for the service principal.

   ```bash
   # ipa-getkeytab -s server.example.com -p HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e des-cbc-crc
   ```

   • The realm name is optional. The FreeIPA server automatically appends the Kerberos realm for which it is configured. You cannot specify a different realm.

   • The hostname must resolve to a DNS A record for it to work with Kerberos. You can use the **--force** flag to force the creation of a principal should this prove necessary.

   • The **-e** argument can include a comma-separated list of encryption types to include in the keytab. This supersedes any default encryption type.

   **WARNING**

   Creating a new key resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

### 6.5.1.2. 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:

```bash
$ 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 FreeIPA domain. (Actually, it can be run on any FreeIPA 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 FreeIPA 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:

```bash
# ipa-getkeytab -s server.example.com -p HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e des-cbc-crc
```

- The realm name is optional. The FreeIPA server automatically appends the Kerberos realm for which it is configured. You cannot specify a different realm.
- The hostname must resolve to a DNS A record for it to work with Kerberos. You can use the **--force** flag to force the creation of a principal should this prove necessary.
- The **-e** argument can include a comma-separated list of encryption types to include in the keytab. This supersedes any default encryption type.

**WARNING**

The **ipa-getkeytab** command resets the secret for the specified principal. This means that all other keytabs for that principal are rendered invalid.

### 6.5.2. Adding Services and Certificates for Services

While services can use keytabs, some services require certificates for access. In that case, a service can be added (or modified) to include a certificate with its service entry.

#### 6.5.2.1. Adding Services and Certificates from the Web UI

1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the **Add** link at the top of the services list.
3. Select the service type from the drop-down menu, and give it a name.
4. Select the hostname of the FreeIPA host on which the service is running. The hostname is used to construct the full service principal name.
5. Click the **Add and Edit** button to go directly to the service entry page.

6. Scroll to the bottom of the page, to the **Service Certificate** section.

7. Click the **New Certificate** button to create the service certificate.

### 6.5.2.2. Adding Services and Certificates from the Command Line

1. Create the service principal. The service is recognized through a name like `service/FQDN`:

   ```bash
   [jsmith@ipaserver ~]$ kinit admin
   [jsmith@ipaserver ~]$ ipa service-add serviceName/hostname
   ```

   For example:

   ```bash
   [jsmith@ipaserver ~]$ ipa service-add HTTP/server.example.com
   -------------------------------------------------------
   Added service "HTTP/server.example.com@EXAMPLE.COM"
   ```
2. Create a certificate for the service. Be sure to copy the keytab to the appropriate directory for the service.

For example:

$ ipa cert-request --principal=HTTP/web.example.com example.csr

TIP

Use the --add option to create the service automatically when requesting the certificate.

Alternatively, use the getcert command, which creates and manages the certificate through certmonger. The options are described more in Section B.1, “Requesting a Certificate with certmonger”.

$ ipa-getcert request -d /etc/httpd/alias -n Server-Cert -K HTTP/client1.example.com -N 'CN=client1.example.com,O=EXAMPLE.COM'

6.5.3. Storing Certificates in NSS Databases

When services use certificates, the certificates and keys can be stored in NSS databases (which may also be used by the services themselves, as well as FreeIPA).

1. Create the NSS databases.

$ certutil -N -d /path/to/database/dir

2. Request the certificate using certutil, an NSS tool.

$ certutil -R -s "CN=client1.example.com,O=EXAMPLE.COM" -d /path/to/database/dir -a > example.csr

If the FreeIPA domain is using Certificate System for its CA, only the CN of the subject name is used. With a self-signed CA, the subject must match the configured certificate subject base. The FreeIPA server rejects requests with a subject base that differs from this value.

6.5.4. Configuring Clustered Services

The FreeIPA server is not cluster aware. However, it is possible to configure a clustered service to be part of FreeIPA 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 FreeIPA domain.

2. Create any service principals and generate the required keytabs.
3. Collect any keytabs that have been set up for services on the host, including the host keytab at /etc/krb5.keytab.

4. Use the ktutil command to produce a single keytab file that contains the contents of all of the keytab files.
   a. For each file, use the rkt command to read the keys from that file.
   b. Use the wkt command to write all of the keys which have been read to a new keytab file.

5. Replace the keytab files on each host with the newly-created combined keytab file.

6. At this point, each host in this cluster can now impersonate any other host.

7. Some services require additional configuration to accommodate cluster members which do not reset hostnames when taking over a failed service.
   - For sshd, set GSSAPIStrictAcceptorCheck no in /etc/ssh/sshd_config.
   - For mod_auth_kerb, set KrbServiceName Any in /etc/httpd/conf.d/auth_kerb.conf.

   **NOTE**

   For SSL servers, the subject name or a subject alternative name for the server’s certificate must appear correct when a client connects to the clustered host. If possible, share the private key among all of the hosts. If each cluster member contains a subject alternative name which includes the names of all the other cluster members will satisfy any client connection requirements.

6.5.5. Using the Same Service Principal for Multiple Services

Within a cluster, the same service principal can be used for multiple services, spread across different machines.

1. Retrieve a service principal using the ipa-getkeytab command.

   ```
   # ipa-getkeytab -s kdc.example.com -p HTTP/server.example.com -k /etc/httpd/conf/krb5.keytab -e des-cbc-crc
   ```

2. Either direct multiple servers or services to use the same file, or copy the file to individual servers as required.

6.6. Disabling and Re-enabling Host and Service Entries

Active services and 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 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.
6.6.1. Disabling Host and Service Entries
Disabling a host or service prevents domain users from access it without permanently removing it from the domain. This can be done by using the `host-disable` and `service-disable` commands.

For example, for a host:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa host-disable server.example.com
```

For a service, specify the principal rather than the hostname:

```
$ 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.

6.6.2. Re-enabling Hosts and Services
Disabling a service or host essentially kills its current, active keytabs. Removing the keytabs effectively removes the host or service from the FreeIPA domain without otherwise touching its configuration entry.

To re-enable a host or service, simply use the `ipa-getkeytab` command. The `-s` option sets which FreeIPA 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 FreeIPA client or server, then it can be run without any LDAP credentials (`-D` and `-w`). The FreeIPA 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 FreeIPA server. The credentials should correspond to the host or service which is being re-enabled.

6.7. Extending Access Permissions over Other Hosts and Services
As discussed in Section 1.3, “Relationships Between Servers and Clients”, within the FreeIPA domain, manage means being able to retrieve a keytab and certificates for another host or service. Every host and service has a `managedby` entry which lists what hosts or services can manage it. By default, a host can manage itself and all of its services. It is also possible to allow a host to manage other hosts, or services on other hosts, by updating the appropriate delegations or providing a suitable `managedby` entry.
Delegating Service Management

A FreeIPA service can be managed from any FreeIPA 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 6.1. 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.

### 6.7.1. Delegating Service Management

A host is delegated control over a service using the `service-add-host` command. There are two parts to delegating the service: specifying the principal and identifying the hosts (in a comma-separated list) with control:

```
# ipa service-add-host principal --hosts=hostnames
```

For example:

```
# ipa service-add-host http/web.example.com --hosts=client1.example.com
```

Once the host is delegated authority, the host principal can be used to manage the service:

```
# kinit -kt /etc/krb5.keytab host/`hostname`
# ipa-getkeytab -s `hostname` -k /tmp/test.keytab -p http/web.example.com
```

```
Keytab successfully retrieved and stored in: /tmp/test.keytab
```

To create a ticket for this service, create a certificate request on the host with the delegated authority and use the `cert-request` command to create a service entry and load the certification information:

```
# ipa cert-request --add --principal=http/web.example.com web.csr
```

Certificate: MIICETCCAXqgA...[snip]
Subject: CN=web.example.com,O=EXAMPLE.COM
Issuer: CN=EXAMPLE.COM Certificate Authority
Not Before: Tue Feb 08 18:51:51 2011 UTC
6.7.2. Delegating Host Management

Hosts are delegated authority over other hosts through the `host-add-managedby` command. This creates a managedby entry. Once the managedby entry is created, then the host can retrieve a keytab for the host it has delegated authority over.

1. Log in as the admin user.

```
# kinit admin
```

2. Add the managedby entry. For example, this delegates authority over client2 to client1.

```
# ipa host-add-managedby client2.example.com --hosts=client1.example.com
```

3. Obtain a ticket as the host client1 and then retrieve a keytab for client2:

```
# kinit -kt /etc/krb5.keytab host/`hostname`
# ipa-getkeytab -s `hostname` -k /tmp/client2.keytab -p host/client2.example.com
Keytab successfully retrieved and stored in: /tmp/client2.keytab
```

6.7.3. Delegating Host or Service Management in the Web UI

Each host and service entry has a configuration tab that indicates what hosts have been delegated management control over that host or service.

1. Open the Identity tab, and select the Hosts or Services subtab.

2. Click the name of the host or service that you are going to grant delegated management to.

3. Click the Hosts subtab on the far right of the host/service entry. This is the tab which lists hosts which can manage the selected host/service.
4. Click the **Add** link at the top of the list.

5. Click the checkbox by the names of the hosts to which to delegate management for the host/service. Click the right arrows button, **>>**, to move the hosts to the selection box.
Chapter 6. Identity: Managing Hosts and Services

6. Click the **Add** button to close the selection box and to save the delegation settings.

### 6.7.4. Accessing Delegated Services

For both services and hosts, if a client has delegated authority, it can obtain a keytab for that principal on the local machine. For services, this has the format `service/hostname@REALM`. For hosts, the `service` is host.

With **kinit**, use the `-k` option to load a keytab and the `-t` option to specify the keytab.

For example, to access a host:

```
# kinit -kt /etc/krb5.keytab host/ipa.example.com@EXAMPLE.COM
```

To access a service:

```
# kinit -kt /etc/httpd/conf/krb5.keytab http/ipa.example.com@EXAMPLE.COM
```
6.8. Managing Public SSH Keys for Hosts

OpenSSH uses public-private key pairs to authenticate hosts. One machine attempts to access another machine and presents its key pair. The first time the host authenticates, the administrator on the target machine has to approve the request manually. The machine then stores the host's public key in a known_hosts file. Any time that the remote machine attempts to access the target machine again, the target machine simply checks its known_hosts file and then grants access automatically to approved hosts.

There are a few problems with this system:

- The known_hosts file stores host entries in a triplet of the host IP address, hostname, and key. This file can rapidly become out of date if the IP address changes (which is common in virtual environments and data centers) or if the key is updated.

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

- Administrators have to approve host keys to add them to the configuration, but it is difficult to verify either the host or key issuer properly, which can create security problems.

On Fedora, 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 FreeIPA as one of its identity information providers, FreeIPA provides a universal and centralized repository of keys. Administrators do not need to worry about distributing, updating, or verifying host SSH keys.

6.8.1. About FreeIPA Clients and OpenSSH

The ipa-client-install script, by default, configures an OpenSSH server and client on the FreeIPA 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 FreeIPA for key caching and retrieval.

**NOTE**

Even if the machine is added as an FreeIPA client using ipa-client-install, the client is not created with any SSH keys. These keys need to be created separately and added to the host account, as described in Section 6.8.2, “Adding Host Keys”.

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 FreeIPA DNS records, where the host keys are stored.

Alternatively, it is also 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-sshf, prevents the host from creating DNS SSHFP records with its own DNS entries. This can be used with or without the --no-sshd option.
6.8.2. Adding Host Keys

Host SSH keys are added to host entries in FreeIPA, either when the host is created using `host-add` or by modifying the entry later.

In a key file, such as a user's `id_rsa.pub` file, a key entry is identified by its type and then the key itself. For example, for an RSA key:

```
ssh-rsa ABCD1234==
```

Only the second part, the base 64-encoded key itself, is uploaded to the user entry.

**NOTE**

Host keys are not created by the `ipa-client-install` command.

1. Run the `host-mod` command with the `--sshpubkey` option to upload the 64 bit-encoded public key to the host entry.

   Adding a host key also changes the DNS SSHFP entry for the host, so also use the `--updatedns` option to update the host's DNS entry.

   For example:

   ```
   [jsmith@server ~]$ ipa host-mod --sshpubkey="12345abcde=" --updatedns host1.example.com
   ```

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

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

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

   **TIP**

   A host can have multiple public keys.

2. After uploading the user keys, configure SSSD to use FreeIPA as one of its identity domains and set up OpenSSH to use the SSSD tooling for managing user keys. This is covered in the *Fedora Deployment Guide*.

6.8.3. Removing Host Keys

Host keys can be removed once they expire or are no longer valid.

Use the `--sshpubkey=` with 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:
6.9. Renaming Machines and Reconfiguring FreeIPA Client Configuration

The hostname of a system is critical for the correct operation of Kerberos and SSL. Both of these security mechanisms rely on the hostname to ensure that communication is occurring between the specified hosts. Infrastructures which use virtual machines or clustered servers will commonly have hosts which are renamed because systems are copied, moved, or renamed.

Fedora does not provide a simple rename command to facilitate the renaming of a FreeIPA host. Renaming a host in a FreeIPA domain involves deleting the entry in FreeIPA, uninstalling the client software, changing the hostname, and re-enrolling using the new name. Additionally, part of renaming hosts requires regenerating service principals.

To reconfigure the client:

1. Identify which services are running on the machine. These need to be re-created when the machine is re-enrolled.

   ```bash
   # 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.

   ```bash
   # ipa hostgroup-find server.example.com
   ```

3. For any service principals (in addition to the host principal), determine the location of the corresponding keytabs on `server.example.com`. The keytab location is different for each service, and FreeIPA does not store this information.

   Each service on the client system has a Kerberos principal in the form `service name/hostname@REALM`, such as `ldap/server.example.com@EXAMPLE.COM`.

4. Unenroll the client machine from the FreeIPA domain:

   ```bash
   # ipa-client-install --uninstall
   ```

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

   ```bash
   # ipa-rmkeytab -k /path/to/keytab -r EXAMPLE.COM
   ```
Chapter 6. Identity: Managing Hosts and Services

6. On another FreeIPA machine, as a FreeIPA administrator, remove the host entry. This removes all services and revokes all certificates issued for that host:

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

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

7. Rename the machine.

8. Re-enroll the system with FreeIPA:

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

This generates a host principal for with the new hostname in `/etc/krb5.keytab`.

9. For every service that needs a new keytab, run the following command:

```
# ipa service-add serviceName/new-hostname
```

10. To generate certificates for services, use either `certmonger` or the FreeIPA administration tools.

11. Re-add the host to any applicable host groups.

### 6.10. Managing Host Groups

Host groups are a way of centralizing control over important management tasks, particularly access control.

All groups in FreeIPA are essentially static groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, FreeIPA 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 FreeIPA or infrastructure usage guidelines for access controls.

### 6.10.1. Creating Host Groups

#### 6.10.1.1. Creating Host Groups from the Web UI

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

2. Click the **Add** link at the top of the groups list.
3. Enter the name and a description for the group.

4. Click the **Add and Edit** button to go immediately to the member selection page.

5. Select the members, as described in *Section 6.10.2.1, “Adding Group Members from the Web UI”*. 
6.10.1.2. 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.

```bash
$ ipa hostgroup-add groupName --desc="description"
```

6.10.2. Adding Group Members

6.10.2.1. Adding Group Members from the Web UI
1. Open the Identity tab, and select the Host Groups subtab.
2. Click the name of the group to which to add members.
3. Click the Add link at the top of the task area.
4. Click the checkbox by the names of the hosts to add, and click the right arrows button, >>, to move the hosts to the selection box.
5. Click the Add button.

6.10.2.2. Adding Group Members from the Command Line

Members are added to a host group using the `hostgroup-add-member` command. This command can add both hosts as group members and other groups as group members.

The syntax of the `hostgroup-add-member` command requires only the group name and a comma-separated list of hosts to add:

```
$ ipa hostgroup-add-member groupName [--hosts=list] [--hostgroups=list]
```

For example, this adds three hosts to the `caligroup` group:

```
$ ipa hostgroup-add-member caligroup --
hosts=ipaserver.example.com,client1.example.com,client2.example.com
Group name: caligroup
Description: for machines in california
GID: 387115842
Member hosts: ipaserver.example.com,client1.example.com,client2.example.com
Number of members added 3
```
Likewise, other groups can be added as members, which creates nested groups:

```
$ ipa hostgroup-add-member caligroup --groups=mountainview,sandiego
Group name: caligroup
Description: for machines in california
GID: 387115842
Member groups: mountainview,sandiego
-------------------------
Number of members added 2
-------------------------
```

6.11. Troubleshooting Host Problems

6.11.1. Certificate Not Found/Serial Number Not Found Errors

The FreeIPA information is stored in a separate LDAP directory than the certificate information, and these two LDAP databases are replicated separately. It is possible for a replication agreement to be broken for one directory and working for another, which can cause problems with managing clients.

Specifically, if the replication agreement between the two CA databases is broken, then a server may not be able to find certificate information about a valid FreeIPA client, causing certificate errors:

```
Certificate operation cannot be completed: EXCEPTION (Certificate serial number 0x2d not found)
```

For example, a FreeIPA server and replica have a function replication agreement between their FreeIPA databases, but the replication agreement between their CA databases is broken. If a host is created on the server, the host entry is replicated over to the replica — but the certificate for that host is not replicated. The replica is aware of the client, but any management operations for that client will fail because the replica doesn't have a copy of its certificate.

6.11.2. Debugging Client Connection Problems

Client connection problems are apparent immediately. This can mean that users cannot log into a machine or attempts to access user and group information fails (for example, `getent passwd admin`).

Authentication in FreeIPA is managed with the SSSD daemon, which is described in the Red Hat Enterprise Linux Deployment Guide. If there are problems with client authentication, then check the SSSD information.

First, check the SSSD logs in `/var/log/sssd/`. There is a specific log file for the DNS domain, such as `sssd_example.com.log`. If there is not enough information in the logs at the default logging level, then increase the log level.

To increase the log level:

1. Open the `sssd.conf` file.

   ```
   vim /etc/sssd/sssd.conf
   ``

2. In the `[domain/example.com]` section, set debug_level.
debug_level = 9

3. Restart the sssd daemon.

   service sssd restart

4. Check the `/var/log/sssd/sssd_example.com.log` file for the debug messages.
Identity: Integrating with NIS Domains and Netgroups

Network information service (NIS) is one of the most common ways to manage identities and authentication on Unix networks. It is simple and easy to use, but it also has inherent security risks and a lack of flexibility that can make administering NIS domains problematic.

FreeIPA supplies a way to integrate netgroups and other NIS data into the FreeIPA domain, which incorporates the stronger security structure of FreeIPA over the NIS configuration. Alternatively, administrators can simply migrate user and host identities from a NIS domain into the FreeIPA domain.

7.1. About NIS and FreeIPA

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 FreeIPA, NIS objects are integrated into FreeIPA using the underlying LDAP directory. LDAP services offer support for NIS objects (as defined in RFC 2307), which FreeIPA 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. FreeIPA can use a NIS listener to map passwords, groups, and netgroups to FreeIPA entries.

In FreeIPA 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.

---

Chapter 7. Identity: Integrating with NIS Domains and Netgroups

In FreeIPA, these netgroup entries are handled using the **netgroup-*** commands, which show the basic LDAP entry:

```plaintext
# 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 FreeIPA translates the LDAP entry into a traditional NIS map and sends it to a client over the NIS protocol (using a NIS plug-in) or it translates it into an LDAP format that is compliant with RFC 2307 or RFC 2307bis.

For more information on NIS, see the Berkeley lab manpages at [http://compute.cnr.berkeley.edu/cgi-bin/man-cgi?netgroup+4](http://compute.cnr.berkeley.edu/cgi-bin/man-cgi?netgroup+4).

### 7.2. Setting the NIS Port for FreeIPA

The FreeIPA 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 FreeIPA 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 FreeIPA’s internal Directory Server instance. To specify the port:

1. Edit the plug-in configuration and add the port number as an argument. For example, to set the port to 514:

   ```bash
   [root@ipaserver ~]# ldapmodify -x -D 'cn=directory manager' -w secret
   ```
Creating Netgroups

```
 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"
```

2. Restart the Directory Server to load the new plug-in configuration.

   ```
   [root@ipaserver ~]# service dirsrv restart
   ```

7.3. Creating Netgroups

All netgroups in FreeIPA are essentially static groups, meaning that the members of the group are manually and explicitly added to the group. Tangentially, FreeIPA 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.

7.3.1. Adding Netgroups

7.3.1.1. With the Web UI

1. Open the Identity tab, and select the Netgroups subtab.

2. Click the Add link at the top of the netgroups list.

3. Enter both a unique name and a description for the netgroup. Both the name and description are required.
The group name is the identifier used for the netgroup in the FreeIPA 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 FreeIPA domain, but it can be changed.
   a. Click the **Settings** tab.
   b. Enter the name of the alternate NIS domain in the **NIS domain name** field.
The **NIS domain name** field sets the domain that appears in the netgroup triple. It does *not* affect which NIS domain the FreeIPA listener responds to.

6. Add members, as described in Section 7.3.2.1, “With the Web UI”.

### 7.3.1.2. 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 FreeIPA domain, but it can be set to something different, depending on the network configuration.

```
$ ipa netgroup-add --desc="description" [--nisdomain=domainName] groupName
```

For example:

```bash
# ipa netgroup-add --desc="my new netgroup" example-netgroup
# ipa netgroup-add-member --hosts=ipa.example.com example-netgroup
# ypcat -d example.com -h ipa.example.com netgroup
(ipa.example.com,-,example.com)
```
NOTE

The `--nisdomain` option sets the domain that appears in the netgroup triple. It does not affect which NIS domain the FreeIPA listener responds to.

7.3.2. 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.

7.3.2.1. With the Web UI

1. Open the Identity tab, and select the Netgroups subtab.

2. Click the name of the netgroup to which to add members.
3. Select the tab for the type of netgroup member to add. Netgroups can have users, user groups, hosts, host groups, and other netgroups as members.

4. Click the Add link at the top of the task area.

5. Click the checkbox by the names of the users to add, and click the right arrows button, >>, to move the names to the selection box.
6. Click the Add button.

7.3.2.2. With the Command Line

Once the group is configured, begin adding netgroup members with the `netgroup-add-member` command. Users, groups, hosts, host groups, and other netgroups can all be added to the netgroup entry. The entry name of the NIS group being edited usually comes at the end of the command:

```
# ipa netgroup-add-member --users=users --groups=groups --hosts=hosts --hostgroups=hostGroups --netgroups=netgroups groupName
```

To set more than one member, use a comma-separated list with the option. For example, this sets two users and two hosts with the other configuration:

```
# ipa netgroup-add-member --users=jsmith,bjensen --groups=ITadmin --hosts=host1.example.com,host2.example.com --hostgroups=EngDev --netgroups=nisgroup2 example-group
```
7.4. Exposing Automount Maps to NIS Clients

When the NIS service is enabled on a system, the FreeIPA server is automatically configured to set the NIS domain to the FreeIPA domain's name, and to include FreeIPA 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 FreeIPA for them to be exposed to NIS clients. The NIS server is managed by a special plug-in entry in the FreeIPA LDAP directory; this is a container entry, and each NIS domain and map used by the NIS server is configured as a child entry beneath that container. The NIS domain entry in the must have the name of the NIS domain, the name of the NIS map, how to find the directory entries to use as the NIS map's contents, and which attributes to use as the NIS map's key and value. Most of these settings will be the same for every map.

The FreeIPA server stores the automount maps, grouped by automount location, in the cn=automount branch of the FreeIPA directory tree.

The NIS domain and map is added using LDAP tools, like `ldapadd`, and editing the directory directly. For example, this adds an automount map that is named `auto.example` in a location named `default` and for a server named `nisserver`:

```
ldapadd -h nisserver.example.com -x -D "cn=Directory Manager" -w secret
dn: nis-domain=example.com+nis-map=auto.example,cn=NIS Server,cn=plugins,cn=config
objectClass: extensibleObject
nis-domain: example.com
nis-map: auto.example
nis-filter: (objectclass=automount)
nis-key-format: %{automountKey}
nis-value-format: %{automountInformation}
nis-base: automountmapname=auto.example,cn=default,cn=automount,dc=example,dc=com
```

A similar add operation needs to be run for every map that is configured.

7.5. Migrating from NIS to FreeIPA

There is no direct migration path from NIS to FreeIPA. This is a manual process with three major steps: setting up netgroup entries in FreeIPA, exporting the existing data from NIS, and importing that data into FreeIPA. There are several options for how to set up the FreeIPA environment and how to export data; the best option depends on the type of data and the overall network environment that you have.

7.5.1. Preparing Netgroup Entries in FreeIPA

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 FreeIPA and delete the netgroups entirely. Otherwise, create the user entries in FreeIPA and then create a FreeIPA-managed netgroup and add those users as members. This is described in Section 7.3, "Creating Netgroups".
Chapter 7. Identity: Integrating with NIS Domains and Netgroups

For host entries
Whenever a host group is created in FreeIPA, 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 FreeIPA. In that case, each entry can be created using the original NIS names:

1. Create an entry for every user referenced in a netgroup.
2. Create an entry for every host referenced in a netgroup.
3. Create a netgroup with the same name as the original netgroup.
4. Add the users and hosts as direct members of the netgroup. Alternatively, put the users and hosts into FreeIPA groups or other netgroups, and then add those groups as members to the netgroup.

7.5.2. Enabling the NIS Listener in FreeIPA
The FreeIPA 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. FreeIPA uses three NIS maps:

- `passwd`
- `group`
- `netgroup`

Using FreeIPA 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 FreeIPA:

1. Obtain new Kerberos credentials as a FreeIPA admin user.
   
   ```
   [root@ipaserver ~]# kinit admin
   ```

2. Enable the NIS listener and compatibility plug-ins:
   
   ```
   [root@ipaserver ~]# ipa-nis-manage enable
   [root@ipaserver ~]# ipa-compat-manage enable
   ```

3. Restart the DNS and Directory Server service:
   
   ```
   [root@server ~]# service restart rpcbind
   [root@server ~]# service restart dirsrv
   ```

7.5.3. Setting Weak Password Encryption for NIS User Authentication to FreeIPA
A NIS server can handle CRYPT password hashes. Once an existing NIS server is migrated to FreeIPA (and its underlying LDAP database), it may still be necessary to preserve the NIS-supported
CRYPT passwords. However, the LDAP server does not use CRYPT hashes by default. It uses SSHA or SSHA-256. If the 389 Directory Server password hash is not changed, then NIS users cannot authenticate to the FreeIPA domain, and `kinit` fails with password failures.

To set the underlying 389 Directory Server to use CRYPT as the password hash, change the `passwordStorageScheme` attribute using `ldapmodify`:

```
[root@server ~]# ldapmodify -D "cn=directory server" -w secret -p 389 -h ipaserver.example.com
dn: cn=config
changetype: modify
replace: passwordStorageScheme
passwordStorageScheme: crypt
```

**NOTE**

Changing the password storage scheme only applies the scheme to new passwords; it does not retroactively change the encryption method used for existing passwords.

If weak crypto is required for password hashes, it is better to change the setting as early as possible so that more user passwords use the weaker password hash.
Chapter 8.

Identity: Integrating with Microsoft Active Directory

FreeIPA uses active synchronization to integrate user data stored in an Active Directory domain and the user data stored in the FreeIPA domain. Critical user attributes, including passwords, are synchronized between the services.

The capability to sync Active Directory and FreeIPA domains is inherent when a FreeIPA server is first installed. The synchronization process is configured by creating agreements between the FreeIPA server and the Active Directory domain controller.

This chapter describes how to configure synchronization, how to configure Active Directory for integration with FreeIPA, and how to configure Windows systems within the Active Directory domain to be aware of the FreeIPA domain.

8.1. About Active Directory and FreeIPA

Within the FreeIPA domain, information is shared among servers and replicas by copying that information, reliably and predictably, between the data masters (servers) and other data masters. This process is replication.

A similar process can be used to share data between the FreeIPA domain and a Microsoft Active Directory domain. This is synchronization.

Synchronization is the process of copying data back and forth between Active Directory and FreeIPA.

Synchronization is defined in an agreement between a FreeIPA server and an Active Directory domain controller. The sync agreement defines all of the information required to identify sync-able user entries (like the subtree to synchronize and requisite object classes in the user entries) as well as defining how account attributes are handled. The sync agreements are created with default values which can be tweaked to meet the needs of a specific domain. When two servers are involved in synchronization, they are called peers.

Synchronization is most commonly bi-directional. Information is sent back and forth between the FreeIPA domain and the Windows domain in a process that is very similar to how FreeIPA servers and replicas share information among themselves. It is possible to configure synchronization — or certain data areas — to only sync one way. That is uni-directional synchronization.

To prevent the risk of data conflicts, synchronization is configured between one FreeIPA server and one Active Directory domain controller. The FreeIPA server propagates changes back to the FreeIPA domain, while the domain controller propagates changes back to the Windows domain.

There are some key features to FreeIPA synchronization:

- A synchronization operation runs every five minutes.
- Synchronization can only be configured with one Active Directory domain. Multiple domains are not supported.
- Synchronization can only be configured with one Active Directory domain controller. However, it is possible to have a list of failover Active Directory domain controllers. Likewise, there can be a list of failover FreeIPA servers to keep synchronization uninterrupted.
- Only user information is synchronized.
• Both user attributes and passwords can be synchronized.

• While modifications are bi-directional (going both from Active Directory to FreeIPA and from FreeIPA to Active Directory), creating or adding accounts are only uni-directional, from Active Directory to FreeIPA. New accounts created in Active Directory are synchronized over to FreeIPA automatically. However, user accounts created in FreeIPA must also be created in Active Directory before they will be synchronized.

• Account lock information is synchronized by default, so a user account which is disabled in one domain is disabled in the other.

• Password synchronization changes take effect immediately.

When Active Directory users are synchronized over to FreeIPA, certain attributes (including Kerberos and POSIX attributes) will have IPA attributes are automatically added to the user entries. These attributes are used by FreeIPA within its domain. They are not synchronized back over the corresponding Active Directory user entry.

Some of the data in synchronization can be modified as part of the synchronization process. For examples, certain attributes can be automatically added to Active Directory user accounts when they are synced over to the FreeIPA domain. These attribute changes are defined as part of the synchronization agreement and are described in Section 8.4.3, “Changing the Behavior for Syncing User Account Attributes”.

8.2. About Synchronized Attributes

8.2.1. User Attribute Synchronization
FreeIPA synchronizes a subset of user attributes between FreeIPA and Active Directory user entries. Any other attributes present in the entry, either in FreeIPA or in Active Directory, are ignored by synchronization.

NOTE

Most POSIX attributes are not synchronized.

Although there are significant schema differences between the Active Directory LDAP schema and the 389 Directory Server LDAP schema used by FreeIPA, there are many attributes that are the same. These attributes are simply synchronized between the Active Directory and FreeIPA user entries, with no changes to the attribute name or value format.

User Schema That Are the Same in FreeIPA and Windows Servers
• cn
• physicalDeliveryOfficeName
• description
• postOfficeBox
• destinationIndicator
• postalAddress
• facsimileTelephoneNumber
• postalCode
• givenname
• registeredAddress
• homePhone
• sn
• homePostalAddress
• st
• initials
• street
• l
• telephoneNumber
• mail
• teletexTerminalIdentifier
• mobile
• telexNumber
• o
• title
• ou
• usercertificate
• pager
• x121Address

Some attributes have different names but still have direct parity between FreeIPA (which uses 389 Directory Server) and Active Directory. These attributes are *mapped* by the synchronization process.

<table>
<thead>
<tr>
<th>User Attribute Synchronization</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn</td>
</tr>
<tr>
<td>nsAccountLock</td>
</tr>
<tr>
<td>ntUserDomainId</td>
</tr>
<tr>
<td>ntUserHomeDir</td>
</tr>
<tr>
<td>ntUserScriptPath</td>
</tr>
<tr>
<td>ntUserLastLogon</td>
</tr>
</tbody>
</table>

Table 8.1. User Schema Mapped between FreeIPA and Active Directory
### Chapter 8. Identity: Integrating with Microsoft Active Directory

FreeIPA | Active Directory
---|---
ntUserLastLogoff | lastLogoff
ntUserAcctExpires | accountExpires
ntUserCodePage | codePage
ntUserLogonHours | logonHours
ntUserMaxStorage | maxStorage
ntUserProfile | profilePath
ntUserParms | userParameters
ntUserWorkstations | userWorkstations

The \textit{cn} is mapped directly (\textit{cn} to \textit{cn}) when syncing from FreeIPA to Active Directory. When syncing from Active Directory \textit{cn} is mapped from the \textit{name} attribute in Active Directory to the \textit{cn} attribute in FreeIPA.

#### 8.2.1.1. User Schema Differences between FreeIPA and Active Directory

Even though attributes may be successfully synced between Active Directory and FreeIPA, there may still be differences in how Active Directory and FreeIPA define the underlying X.500 object classes. This could lead to differences in how the data are handled in the different LDAP services.

This section describes the differences in how Active Directory and FreeIPA handle some of the attributes which can be synchronized between the two domains.

##### 8.2.1.1.1. Values for \textit{cn} Attributes

In 389 Directory Server, the \textit{cn} attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the FreeIPA \textit{cn} attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that, potentially, if a \textit{cn} value is added to an Active Directory entry and that value is not one of the values for \textit{cn} in FreeIPA, then all of the FreeIPA \textit{cn} values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the \textit{cn} attribute as its naming attribute, where FreeIPA uses \textit{uid}. This means that there is the potential to rename the entry entirely (and accidentally) if the \textit{cn} attribute is edited in the FreeIPA. If that \textit{cn} change is written over to the Active Directory entry, then the entry is renamed, and the new named entry is written back over to FreeIPA.

##### 8.2.1.1.2. Values for \textit{street} and \textit{streetAddress}

Active Directory uses the attribute \textit{streetAddress} for a user or group’s postal address; this is the way that 389 Directory Server uses the \textit{street} attribute. There are two important differences in the way that Active Directory and FreeIPA use the \textit{streetAddress} and \textit{street} attributes, respectively:

- In 389 Directory Server, \textit{streetAddress} is an alias for \textit{street}. Active Directory also has the \textit{street} attribute, but it is a separate attribute that can hold an independent value, not an alias for \textit{streetAddress}.
- Active Directory defines both \textit{streetAddress} and \textit{street} as single-valued attributes, while 389 Directory Server defines \textit{street} as a multi-valued attribute, as specified in RFC 4519.
Because of the different ways that 389 Directory Server and Active Directory handle `streetAddress` and `street` attributes, there are two rules to follow when setting address attributes in Active Directory and FreeIPA:

- The synchronization process maps `streetAddress` in the Active Directory entry to `street` in FreeIPA. To avoid conflicts, the `street` attribute should not be used in Active Directory.

- Only one FreeIPA `street` attribute value is synced to Active Directory. If the `streetAddress` attribute is changed in Active Directory and the new value does not already exist in FreeIPA, then all `street` attribute values in FreeIPA are replaced with the new, single Active Directory value.

### 8.2.1.1.3. Constraints on the initials Attribute

For the `initials` attribute, Active Directory imposes a maximum length constraint of six characters, but 389 Directory Server does not have a length limit. If an `initials` attribute longer than six characters is added to FreeIPA, the value is trimmed when it is synchronized with the Active Directory entry.

### 8.2.1.1.4. Requiring the surname (sn) Attribute

Active Directory allows `person` entries to be created without a surname attribute. However, RFC 4519 defines the `person` object class as requiring a surname attribute, and this is the definition used in Directory Server.

If an Active Directory `person` entry is created without a surname attribute, that entry will not be synced over to FreeIPA since it fails with an object class violation.

### 8.2.1.2. Active Directory Entries and RFC 2307 Attributes

Windows uses unique, random security IDs (SIDs) to identify users. These SIDs are assigned in blocks or ranges, identifying different system user types within the Windows domain. When users are synchronized between FreeIPA and Active Directory, Windows SIDs for users are mapped to the Unix UIDs used by the FreeIPA entry. Another way of saying this is that the Windows SID is the only ID within the Windows entry which is used as an identifier in the corresponding Unix entry, and then it is used in a mapping.

When Active Directory domains interact with Unix-style applications or domains, then the Active Directory domain may use Services for Unix or IdM for Unix to enable Unix-style `uidNumber` and `gidNumber` attributes. This allows Windows user entries to follow the specifications for those attributes in RFC 2307\(^2\).

However, the `uidNumber` and `gidNumber` attributes are not actually used as the `uidNumber` and `gidNumber` attributes for the FreeIPA entry. The FreeIPA `uidNumber` and `gidNumber` attributes are generated when the Windows user is synced over.

---

8.2.2. Group Attribute Synchronization

8.2.2.1. About Windows Group Types
In Active Directory, there are two major types of groups: security and distribution. Security groups are most similar to groups in FreeIPA, since security groups can have policies configured for access controls, resource restrictions, and other permissions. Distribution groups are for mailing distribution. These are further broken down into global and local groups. The FreeIPA `ntGroupType` supports all four group types:

- `-21483646` for global/security (the default)
- `-21483644` for domain local/security
- `2` for global/distribution
- `4` for domain local/distribution

8.2.2.2. Group Attributes Synchronized between FreeIPA and Active Directory
Only a subset of FreeIPA and Active Directory attributes are synchronized. These attributes are hard-coded and are defined regardless of which way the entry is being synchronized. Any other attributes present in the entry, either in FreeIPA or in Active Directory, remain unaffected by synchronization.

Some attributes used in FreeIPA and Active Directory group entries are identical. These are usually attributes defined in an LDAP standard, which are common among all LDAP services. These attributes are synchronized to one another exactly. **Table 8.3, “Group Entry Attributes That Are the Same between FreeIPA and Active Directory”** shows attributes that are the same between the FreeIPA and Windows servers.

Some attributes define the same information, but the names of the attributes or their schema definitions are different. These attributes are mapped between Active Directory and FreeIPA, so that attribute A in one server is treated as attribute B in the other. For synchronization, many of these attributes relate to Windows-specific information. **Table 8.2, “Group Entry Attribute Mapping between FreeIPA and Active Directory”** shows the attributes that are mapped between the FreeIPA and Windows servers.

For more information on the differences in ways that FreeIPA and Active Directory handle some schema elements, see **Section 8.2.2.3, “Group Schema Differences between FreeIPA and Active Directory”**.

**Table 8.2. Group Entry Attribute Mapping between FreeIPA and Active Directory**

<table>
<thead>
<tr>
<th>FreeIPA</th>
<th>Active Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cn</code></td>
<td><code>name</code></td>
</tr>
</tbody>
</table>
Setting up Active Directory for Synchronization

<table>
<thead>
<tr>
<th>FreelIPA</th>
<th>Active Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntUserDomainID</td>
<td>name</td>
</tr>
<tr>
<td>ntGroupType</td>
<td>groupType</td>
</tr>
<tr>
<td>uniqueMember</td>
<td>Member¹</td>
</tr>
<tr>
<td>member</td>
<td>Member</td>
</tr>
</tbody>
</table>

¹ The Member attribute in Active Directory is synced to the uniqueMember attribute in FreeIPA.

Table 8.3. Group Entry Attributes That Are the Same between FreeIPA and Active Directory

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FreelIPA</th>
<th>Active Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>ou</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>seeAlso</td>
<td></td>
</tr>
<tr>
<td>mail</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.2.2.3. Group Schema Differences between FreeIPA and Active Directory

Although Active Directory supports the same basic X.500 object classes as FreeIPA, there are a few incompatibilities of which administrators should be aware.

Nested groups (where a group contains another group as a member) are supported and for WinSync are synchronized. However, Active Directory imposes certain constraints as to the composition of nested groups. For example, a global group contains a domain local group as a member. FreeIPA has no concept of local and global groups, and, therefore, it is possible to create entries in FreeIPA that violate Active Directory's constraints when synchronized.

8.3. Setting up Active Directory for Synchronization

Synchronizing user accounts alone is enabled within FreeIPA, so all that is necessary is to set up a sync agreement (Section 8.4.2, "Creating Synchronization Agreements"). On the Windows server, it is necessary to create the user that the FreeIPA server will use to connect to the Active Directory domain.

The process for creating a user in Active Directory is covered in the Windows server documentation at http://technet.microsoft.com/en-us/library/cc732336.aspx. The new user account must have the proper permissions:

- Grant the sync user account Replicating directory changes rights to the synchronized Active Directory subtree. Replicator rights are required for the sync user to perform synchronization operations.

  Replicator rights are described in http://support.microsoft.com/kb/303972.

- Add the sync user as a member of the Account Operator and Enterprise Read-Only Domain controller groups. It is not necessary for the user to belong to the full Domain Admin group.

8.4. Managing Synchronization Agreements

8.4.1. Trusting the Active Directory and FreeIPA CA Certificates

Both Active Directory and FreeIPA use certificates for server authentication. For the Active Directory and FreeIPA SSL server certificates to be trusted by each other, both servers need to trust the
Chapter 8. Identity: Integrating with Microsoft Active Directory

CA certificate for the CA which issued those certificates. This means that the Active Directory CA certificate needs to be imported into the FreeIPA database, and the FreeIPA CA certificate needs to be imported into the Active Directory database.

1. On the Active Directory server, download the FreeIPA server's CA certificate from http://ipa.example.com/ipa/config/ca.crt.

2. Install the FreeIPA CA certificate in the Active Directory certificate database. This can be done using the Microsoft Management Console or the `certutil` utility. For example:

   ```
certutil -installcert -v -config "ipaserver.example.com\Example Domain CA" c:\path\to\ca.crt
   ```

   For more details, see the Active Directory documentation.

3. Export the Active Directory CA certificate.
   a. In My Network Places, open the CA distribution point. For example, the location on Windows Server 2003 is `C:\WINDOWS\system32\certsrv\CertEnroll\`
   b. Double-click the security certificate file (.crt file) to display the Certificate dialog box.
   c. On the Details tab, click Copy to File to start the Certificate Export Wizard.
   d. Click Next, and then select Base-64 encoded X.509 (.CER).
e. Specify a suitable directory and file name for the exported file. Click **Next** to export the certificate, and then click **Finish**.

4. Copy the Active Directory certificate over to the FreeIPA server machine.

5. Download the FreeIPA server's CA certificate from `http://ipa.example.com/ipa/config/ca.crt`.

6. Copy both the Active Directory CA certificate and the FreeIPA CA certificate into the `/etc/openldap/cacerts/` directory.

7. Update the hash symlinks for the certificates.

   ```bash
cacertdir_rehash /etc/openldap/cacerts/
```

8. Edit the `/etc/openldap/ldap.conf` file, and add the information to point to and use the certificates in the `/etc/openldap/cacerts/` directory.

   ```bash
TLS_CACERTDIR /etc/openldap/cacerts/
TLS_REQCERT allow
```
8.4.2. Creating Synchronization Agreements

Synchronization agreements are created on the FreeIPA server using the `ipa-replica-manage connect` command because it creates a connection to the Active Directory domain. The options to create the synchronization agreement are listed in Table 8.4, “Synchronization Agreement Options”.

1. Make sure that the Active Directory and FreeIPA servers trust each other’s CA certificates, as in Section 8.4.1, “Trusting the Active Directory and FreeIPA CA Certificates”.

2. Remove any existing Kerberos credentials on the FreeIPA server.

   ```
   $ kdestroy
   ```

3. Use the `ipa-replica-manage` command to create a Windows synchronization agreement. This requires the `--winsync` option. If passwords will be synchronized as well as user accounts, then also use the `--passsync` option and set a password to use for Password Sync.

   The `--binddn` and `--bindpwd` options give the username and password of the system account on the Active Directory server that FreeIPA will use to connect to the Active Directory server.

   ```
   $ ipa-replica-manage connect --winsync
   --binddn cn=administrator,cn=users,dc=example,dc=com
   --bindpwd Windows-secret
   --passsync secretpwd
   --cacert /etc/openldap/cacerts/windows.cer
   adserver.example.com -v
   ```

4. When prompted, enter the Directory Manager password.

5. Optional. Configure Password Synchronization, as in Section 8.5.2, “Setting up Password Synchronization”.

Table 8.4. Synchronization Agreement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--winsync</td>
<td>Identifies this as a synchronization agreement.</td>
</tr>
<tr>
<td>--binddn</td>
<td>Gives the full user DN of the synchronization identity. This is the user DN</td>
</tr>
<tr>
<td></td>
<td>that the FreeIPA LDAP server uses to bind to Active Directory. This user</td>
</tr>
<tr>
<td></td>
<td>must exist in the Active Directory domain and must have replicator, read,</td>
</tr>
<tr>
<td></td>
<td>search, and write permissions on the Active Directory subtree.</td>
</tr>
<tr>
<td>--bindpwd</td>
<td>Gives the password for the sync user.</td>
</tr>
<tr>
<td>--passsync</td>
<td>Gives the password for the Windows user account which is involved in</td>
</tr>
<tr>
<td></td>
<td>synchronization.</td>
</tr>
<tr>
<td>--cacert</td>
<td>Gives the full path and file name of the Active Directory CA certificate.</td>
</tr>
<tr>
<td></td>
<td>This certificate is exported in Section 8.4.1, “Trusting the Active Directory</td>
</tr>
<tr>
<td></td>
<td>and FreeIPA CA Certificates”.</td>
</tr>
<tr>
<td>--win-subtree</td>
<td>Gives the DN of the Windows subtree containing the users to synchronize.</td>
</tr>
<tr>
<td></td>
<td>The default value is cn=Users,$SUFFIX.</td>
</tr>
</tbody>
</table>
Changing the Behavior for Syncing User Account Attributes

When the sync agreement is created, it has certain default behaviors defined for how the synchronization process handled the user account attributes during synchronization. The types of behaviors are things like how to handle lockout attributes or how to handle different DN formats. This behavior can be changed by editing the synchronization agreement. The list of attribute-related parameters are in Table 8.5, “Synced Attribute Settings”.

The sync agreement exists as a special plug-in entry in the LDAP server and each attribute behavior is set through an LDAP attribute. To change the sync behavior, use the `ldapmodify` command to modify the LDAP server entry directly.

For example, account lockout attributes are synchronized between FreeIPA and Active Directory by default, but this can be disabled by editing the `ipaWinSyncAcctDisable` attribute. (Changing this means that if an account is disabled in Active Directory, it is still active in FreeIPA and vice versa.)

```
jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password
do: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
replace: ipaWinSyncAcctDisable
ipaWinSyncAcctDisable: none
modifying entry "cn=ipa-winsync,cn=plugins,cn=config"
```

Table 8.5. Synced Attribute Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General User Account Parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipaWinSyncNewEntryFilter</td>
<td>Sets the search filter to use to find the entry which contains the list of object classes to add to new user entries.</td>
<td>The default is (cn=ipaConfig).</td>
</tr>
<tr>
<td>ipaWinSyncNewUserOCAttr</td>
<td>Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.</td>
<td>The default is ipauserobjectclasses.</td>
</tr>
<tr>
<td>ipaWinSyncHomeDirAttr</td>
<td>Identifies which attribute in the entry contains the default location of the POSIX home directory.</td>
<td>The default is ipaHomesRootDir.</td>
</tr>
<tr>
<td>ipaWinSyncUserAttr</td>
<td>Sets an additional attribute with a specific value to add to Active Directory users when they are synced over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and</td>
<td>ipaWinSyncUserAttr: attributeName attributeValue</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Possible Values</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>the sync process adds all of the values to the entry.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This only sets the attribute value if the entry does not already have that attribute present. If the attribute is present, then the entry’s value is used when the Active Directory entry is synced over.</td>
<td></td>
</tr>
<tr>
<td>ipaWinSyncUserFlatten</td>
<td>Sets whether to normalize the DN of Active Directory entries to conform with the FreeIPA directory structure. In FreeIPA, all users are stored under the <code>cn=users,cn=accounts, $SUFFIX</code> entry, but Active Directory can have more branches in its directory, which can result in DNs like <code>cn=John Smith,ou=Development,ou=Engineering,cn=users,dc=example,dc=com</code>. Flattening the DN discards any additional intervening organizational units in the Active Directory DN and creating a simple DN on the FreeIPA side. Any <code>ou</code> attributes are stored in the FreeIPA user entry.</td>
<td>true</td>
</tr>
<tr>
<td>ipaWinSyncForceSync</td>
<td>Sets whether to check existing FreeIPA users which match an existing Active Directory user should be automatically edited so they can be synchronized. If a FreeIPA user account has a <code>uid</code> parameter which is identical to the <code>samAccountName</code> in an existing Active Directory user, then that account is not synced by default. This attribute tells the sync service to add the <code>ntUser</code> and <code>ntUserDomainId</code> to</td>
<td>true</td>
</tr>
</tbody>
</table>
### Changing the Behavior for Syncing User Account Attributes

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>the FreeIPA user entries automatically, which allows them to be synchronized.</td>
<td></td>
</tr>
</tbody>
</table>

### User Account Lock Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
</table>
| ipaWinSyncAcctDisable | Sets which way to synchronize account lockout attributes. It is possible to control which account lockout settings are in effect. For example, **to_ad** means that when account lockout attribute is set in FreeIPA, its value is synced over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synced from both domains. | • both (default)  
• to_ad  
• to_ds  
• none |

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncInactivatedFilter</td>
<td>Sets the search filter to use to find the DN of the group used to hold inactivated (disabled) users. This does not need to be changed in most deployments.</td>
<td>The default is <code>(&amp;(cn=inactivated) (objectclass=groupOfNames))</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncActivatedFilter</td>
<td>Sets the search filter to use to find the DN of the group used to hold active users. This does not need to be changed in most deployments.</td>
<td>The default is <code>(&amp;(cn=activated) (objectclass=groupOfNames))</code>.</td>
</tr>
</tbody>
</table>

### Group Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncDefaultGroupAttr</td>
<td>Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the <code>gidNumber</code> for the user account.</td>
<td>The default is <code>ipaDefaultPrimaryGroup</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncDefaultGroupFilter</td>
<td>Sets the search filter to map the group name to the POSIX <code>gidNumber</code>.</td>
<td>The default is <code>(&amp;(gidNumber=*) (objectclass=posixGroup) (cn=groupAttr_value))</code>.</td>
</tr>
</tbody>
</table>

### Realm Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncRealmAttr</td>
<td>Sets the attribute which contains the realm name in the realm entry.</td>
<td>The default is <code>cn</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipaWinSyncRealmFilter</td>
<td>Sets the search filter to use to find the entry which contains the FreeIPA realm name.</td>
<td>The default is <code>objectclass=krbRealmContainer</code>).</td>
</tr>
</tbody>
</table>
8.4.4. Changing the Synchronized Windows Subtree

Creating a synchronization agreement automatically sets the two subtrees to use as the synchronized user database. In FreeIPA, the default is `cn=users, cn=accounts, $SUFFIX`, and for Active Directory, the default is `CN=Users, $SUFFIX`.

The value for the Active Directory subtree can be set to a non-default value when the sync agreement is created by using the `--win-subtree` option. After the agreement is created, the Active Directory subtree can be changed by using the `ldapmodify` command to edit the `nsds7WindowsReplicaSubtree` value in the sync agreement entry.

1. Get the name of the sync agreement, using `ldapsearch`. This search returns only the values for the `dn` and `nsds7WindowsReplicaSubtree` attributes instead of the entire entry.

```
[jsmith@ipaserver ~]$ ldapsearch -xLLL -D "cn=directory manager" -w password -p 389 -h ipaserver.example.com -b cn=config objectclass=nsdswindowsreplicationagreement dn nsds7WindowsReplicaSubtree

dn: cn=meToWindowsBox.example.com,cn=replica,cn=dc=example,cn=mapping tree,cn=config
  nsds7WindowsReplicaSubtree: cn=users,dc=example,dc=com
  ... 8< ...
```

2. Modify the sync agreement

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -W -p 389 -h ipaserver.example.com <<EOF
  dn: cn=meToWindowsBox.example.com,cn=replica,cn=dc=example,c2=3Dexample,c3=Dcom,cn=mapping
tree, cn=config
  changetype: modify
  replace: nsds7WindowsReplicaSubtree
  nsds7WindowsReplicaSubtree: cn=alternateusers,dc=example,dc=com
EOF
modifying entry "cn=meToWindowsBox.example.com,cn=replica,cn=dc=example,c2=3Dexample,c3=Dcom,cn=mapping tree, cn=config"
```

The new subtree setting takes effect immediately. If a sync operation is currently running, then it takes effect as soon as the current operation completes.

8.4.5. Configuring Uni-Directional Sync

By default, all modifications and deletions are bi-directional. A change in Active Directory is synced over to FreeIPA, and a change to an entry in FreeIPA is synced over to Active Directory. This is essentially an equitable, multi-master relationship, where both Active Directory and FreeIPA are equal peers in synchronization and are both data masters.

However, there can be some data structure or IT designs where only one domain should be a data master and the other domain should accept updates. This changes the sync relationship from a multi-master relationship (where the peer servers are equal) to a master-consumer relationship.

This is done by setting the `onewaySync` parameter on the sync agreement. The possible values are `fromWindows` (for Active Directory to FreeIPA sync) and `toWindows` (for FreeIPA to Active Directory sync).

For example, to sync changes from Active Directory to FreeIPA:
Enabling uni-directional sync does not automatically prevent changes on the un-synchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync is configured to go from Active Directory to FreeIPA, so Active Directory is (in essence) the data master. If an entry is modified or even deleted on the FreeIPA, then the FreeIPA information is different then the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

### Deleting Synchronization Agreements

Synchronization can be stopped by deleting the sync agreement which disconnects the FreeIPA and Active Directory servers. In the inverse of creating a sync agreement, deleting a sync agreement uses the `ipa-replica-manage disconnect` command and then the hostname of the Active Directory server.

1. Delete the sync agreement.

   ```bash
   # ipa-replica-manage disconnect adserver.example.com
   ```

2. Remove the Active Directory CA certificate from the FreeIPA server database:

   ```bash
   # certutil -D -d /etc/dirsrv/slapd-EXAMPLE.COM/ -n "Imported CA"
   ```

### Winsync Agreement Failures

Creating the sync agreement fails because it cannot connect to the Active Directory server.

One of the most common sync agreement failures is that the FreeIPA server cannot connect to the Active Directory server:

"Update failed! Status: [81 - LDAP error: Can't contact LDAP server]"

This can occur if the wrong Active Directory CA certificate was specified when the agreement was created. This creates duplicate certificates in the FreeIPA LDAP database (in the `/etc/dirsrv/slapd-DOMAIN/` directory) with the name `Imported CA`. This can be checked using `certutil`:

```bash
$ certutil -L -d /etc/dirsrv/slapd-DOMAIN/
```

<table>
<thead>
<tr>
<th>Certificate Nickname</th>
<th>Trust Attributes</th>
</tr>
</thead>
</table>

```
Chapter 8. Identity: Integrating with Microsoft Active Directory

SSL, S/MIME, JAR/XPI

| CA certificate | CTu,u,Cu |
| Imported CA    | CT,,C   |
| Server-Cert   | u,u,u   |
| Imported CA    | CT,,C   |

To resolve this issue, clear the certificate database:

```
# certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"
```

This deletes the CA certificate from the LDAP database.

There are errors saying passwords are not being synced because it says the entry exists

For some entries in the user database, there may be an informational error message that the password is not being reset because the entry already exists:

"Windows PassSync entry exists, not resetting password"

This is not an error. This message occurs when an exempt user, the Password Sync user, is not being changed. The Password Sync user is the operational user which is used by the service to change the passwords in FreeIPA.

8.5. Managing Password Synchronization

Password synchronization is configured separately from Windows Synchronization.

8.5.1. Setting up the Windows Server for Password Synchronization

To synchronize passwords requires that Active Directory be running in SSL and that the Password Sync Service be installed on each Active Directory domain controller. The Password Sync Service records password changes and synchronizes them, over a secure connection, to the FreeIPA entry.

TIP

Install the Microsoft Certificate System in Enterprise Root Mode. Active Directory will then automatically enroll to retrieve its SSL server certificate.

1. Make sure that the Active Directory password complexity policies are enabled so that the Password Sync service will run.
   a. Run `secpol.msc` from the command line.
   b. Select Security Settings.
   c. Open Account Policies, and then open Password Policy.
   d. Enable the Password must meet complexity requirements option and save.
2. If SSL is not already enabled, set up SSL on the Active Directory server. Setting up LDAPS is explained in more detail in the Microsoft knowledgebase at http://support.microsoft.com/kb/321051.

   a. Install a certificate authority in the **Windows Components** section in **Add/Remove Programs**.

   b. Select the **Enterprise Root CA** option.

   c. Reboot the Active Directory server. If IIS web services are running, the CA certificate can be accessed by opening http://servername/certsrv.

   d. Set up the Active Directory server to use the SSL server certificate.

      i. Create a certificate request `.inf`, using the fully-qualified domain name of the Active Directory as the certificate subject. For example:

```
;----------------- request.inf -----------------
[Version]
Signature="$Windows NT$

[NewRequest]
Subject = "CN=ad.server.example.com, O=Engineering, L=Raleigh, S=North Carolina, C=US"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = False
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0

[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1

;----------------- request.inf -----------------
```

For more information on the `.inf` request file, see the Microsoft documentation, such as http://technet.microsoft.com/en-us/library/cc783835.aspx.

   ii. Generate the certificate request.

```
certreq -new request.inf request.req
```

   iii. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```
iv. Accept the certificate request. For example:

```
certreq -accept certnew.cer
```

v. Make sure that the server certificate is present on the Active Directory server.

In the File menu, click Add/Remove, then click Certificates and Personal>Certificates.

vi. Import the CA certificate from Directory Server into Active Directory. Click Trusted Root CA, then Import, and browse for the Directory Server CA certificate.

e. Reboot the domain controller.

### 8.5.2. Setting up Password Synchronization

Install the Password Sync Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the PassSync.msi file from the 389 Directory Server repos^4, and save it to the Active Directory machine.

   **NOTE**

   There are two PassSync packages available, one for 32-bit Windows servers and one for 64-bit. Make sure to select the appropriate packages for your Windows platform.

2. Double-click the PassSync.msi file to install it.

3. The Password Sync Setup window appears. Hit Next to begin installing.

4. Fill in the information to establish the connection to the FreeIPA server.
   - The FreeIPA server connection information, including the hostname and secure port number.
   - The username of the system user which Active Directory uses to connect to the FreeIPA machine. This account is configured automatically when sync is configured on the FreeIPA server. The default account is `uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com`.
   - The password set in the `--passsync` option when the sync agreement was created.
• The search base for the people subtree on the FreeIPA server. The Active Directory server connects to the FreeIPA server similar to an `ldapsearch` or replication operation, so it has to know where in the FreeIPA subtree to look for user accounts. The user subtree is `cn=users,cn=accounts,dc=example,dc=com`.

• The certificate token is not used at this time, so that field should be left blank.

Hit **Next**, then **Finish** to install Password Sync.

5. Import the FreeIPA server's CA certificate into the Active Directory certificate store.
   a. Download the FreeIPA server's CA certificate from `http://ipa.example.com/ipa/config/ca.crt`.
   b. Copy the FreeIPA CA certificate to the Active Directory server.
   c. Install the FreeIPA CA certificate in the Password Sync database. For example:

   ```
   cd "C:\Program Files\389 Directory Server Password Synchronization"
   ```
Chapter 8. Identity: Integrating with Microsoft Active Directory

```bash
certutil.exe -d . -A -n "IPASERVER.EXAMPLE.COM IPA CA" -t CT,, -a -i ipaca.crt
```

6. Reboot the Windows machine to start Password Sync.

**NOTE**

The Windows machine must be rebooted. Without the rebooting, `PasswordHook.dll` is not enabled, and password synchronization will not function.

The first attempt to synchronize passwords, which happened when the Password Sync application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory sync peers. The tools to create the certificate and key databases is installed with the `.msi`.

### 8.5.3. Exempting Active Directory Users from Password Synchronization

The passwords in password change operations are still subject to the password policy settings, such as password expiration times. For example, in FreeIPA every password change requires an immediate password reset. While normal user passwords need to be subject to password policies, administrative passwords should be exempt from any password rules. A list of user DNs can be set in the password synchronization configuration that are exempted from the password policy.

**NOTE**

The Directory Manager password is always exempt from password policy.

Edit the password synchronization entry, `cn=ipa_pwd_extop,cn=plugins,cn=config`, and add the `passSyncManagersDNs` attribute with the name of the user. This attribute is multi-valued. For example:

```bash
$ ldapmodify -x -D "cn=Directory Manager" -w secret -h ldap.example.com -p 389
dn: cn=ipa_pwd_extop,cn=plugins,cn=config
changetype: modify
add: passSyncManagersDNs
passSyncManagersDNs: uid=admin,cn=users,cn=accounts,dc=example,dc=com
```
Identity: Managing DNS

If the FreeIPA server was installed with DNS configured, then all of the DNS entries for the domain — host entries, locations, records — can be managed using the FreeIPA tools.

9.1. About DNS in FreeIPA

DNS is one of the services that can be configured and maintained by the FreeIPA domain. DNS is critical to the performance of the FreeIPA domain; DNS is used for the Kerberos services and SSL connections for all servers and clients and for connections to domain services like LDAP.

While FreeIPA can use an external DNS service, there is a lot more flexibility and control over FreeIPA — DNS interactions when the DNS service is configured within the domain. For example, DNS records and zones can be managed within the domain using FreeIPA tools, and clients can update their own DNS records dynamically. When a host is added to FreeIPA, a DNS record is automatically created in FreeIPA’s DNS service for that host machine.

FreeIPA stores all DNS information as LDAP entries. Every resource record for each machine is stored for the domain. For example, the client1 resource has three IPv4 (A) records and one IPv6 (AAAA) record:

dn: idnsname=client1,idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: client1
arecord: 10.0.0.1
arecord: 10.0.0.2
arecord: 10.0.0.3
aaaarecord: fc00::1
objectclass: top
objectclass: idnsrecord

The schema used to define the DNS entries is in the `/usr/share/ipa/60basev2.ldif` schema file.

The BIND service communicates with the Directory Server using the system `bind-dyndb-ldap` plug-in. When FreeIPA is configured to manage DNS (Section 9.3, “Setting up DNS After FreeIPA Server Installation”), FreeIPA creates a `dynamic-db` configuration section in the `/etc/named.conf` file for the BIND service. This configures the `bind-dyndb-ldap` plug-in for the BIND (named) service.

When this plug-in is properly configured, it delivers the DNS records from the Directory Server to the named service. The configuration can be changed to adapt the behavior of the plug-in and, therefore, the LDAP-BIND interactions.

9.2. The FreeIPA-Generated DNS File

To help create and configure a suitable DNS setup, the FreeIPA installation script creates a sample zone file. During the installation, FreeIPA displays a message similar to the following:

```
Sample zone file for bind has been created in /tmp/sample.zone.F_uMf4.db
```

If a DNS server is already configured in the network, then the configuration in the FreeIPA-generated file can be added to the existing DNS zone file. This allows FreeIPA clients to find LDAP and Kerberos

---

1 Any updated schema files, included updated DNS schema elements, are located in the `/usr/share/ipa/updates` directory.
servers that are required for them to participate in the FreeIPA domain. For example, this DNS zone configuration is created for an FreeIPA server with the KDC and DNS servers all on the same machine in the EXAMPLE.COM realm:

```ini
; ldap servers
ldap._tcp              IN SRV 0 100 389        ipaserver.example.com.

; kerberos realm
_kerberos               IN TXT EXAMPLE.COM

; kerberos servers
_kerberos._tcp          IN SRV 0 100 88         ipaserver.example.com.
_kerberos._udp          IN SRV 0 100 88         ipaserver.example.com.
_kerberos-master._tcp   IN SRV 0 100 88         ipaserver.example.com.
_kerberos-master._udp   IN SRV 0 100 88         ipaserver.example.com.
_kpasswd._tcp           IN SRV 0 100 464        ipaserver.example.com.
_kpasswd._udp           IN SRV 0 100 464        ipaserver.example.com.
```

### 9.3. Setting up DNS After FreeIPA Server Installation

DNS can be configured as part of the FreeIPA server installation, simply by using the `--setup-dns` option. If DNS is not configured then, it can be configured later using the `ipa-dns-install` command. For example:

```bash
ipa-dns-install -p secret --ip-address=1.2.34.56 --no-forwarders [-p password] [-z zone] [-r refresh]
```

- **-p** gives the password for the Directory Manager user in the 389 Directory Server. All of the DNS entries are stored in the LDAP directory, so this directory must be accessed to add the DNS configuration.

- **--ip-address** gives the IP address for the master DNS server.

- **--no-forwarders** means that there are no forwarders used with the DNS service, only root servers. Alternatively, use the `--forwarder` option to define a forward to use; to specify multiple forwarders, use the `--forwarder` option multiple times.

Reverse DNS is configured automatically. It is possible to disable reverse DNS by using the `--no-reverse` option.

If the existing reverse DNS zone is already configured, using the `--no-reverse` option uses the existing reverse zone rather than creating a new reverse zone.

The FreeIPA server can actively check to see when new DNS zones are added and to update its DNS server accordingly. If no value is explicitly given, the zone refresh period is 30 seconds. The refresh interval can be set to another value using the `--zone-refresh` option, which sets the polling interval in seconds.

Similar to refreshing the zones, the FreeIPA server can leave a persistent search open with its Directory Server and capture any new zone changes immediately. This is enabled with the `--zone-notif` option.

If the `--zone-notif` option is used to configure DNS, then the automatic zone refresh is disabled.
9.4. Managing DNS Zone Entries

9.4.1. Adding DNS Zones

9.4.1.1. Adding DNS Zones from the Web UI
1. Open the Identity tab, and select the DNS subtab.
2. Click the Add link at the top of the list of DNS zones.

3. Fill in the information about the new DNS zone. The Zone Name is required; this is the actual domain name. The other information about the administrator email and the authoritative name server are optional.
4. Click the **Add and Add Another** button to go directly to the DNS zone page. In the **Settings** tab, it is possible to reset the default zone configuration to enable dynamic binds (Section 9.8.1, "Enabling Dynamic DNS Updates in the Web UI") or change other default records information (Section 9.4.2.1, "Editing the Zone Configuration in the Web UI"). It is also possible to begin adding new DNS resource records (Section 9.5.1.1, "Adding DNS Resource Records from the Web UI") in the DNS Resource Records tab.
Adding DNS Zones from the Command Line

The `ipa dnszone-add` command adds a new zone to the DNS domain. At a minimum, this requires the name of the new subdomain:

```
$ ipa dnszone-add domainName
```

If the name is not given, the script prompts for it. Other command-line options can also be passed with the `ipa dnszone-add` command.

To add a zone entry:

1. Add the new zone. For example:

```
$ ipa dnszone-add newserver.example.com --admin-email=admin@example.com --minimum=3000 --dynamic-update=TRUE
```

2. Reload the name service.

```
# rndc reload
```

**TIP**

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the named service or configure the BIND service to poll the Directory Server automatically for zone changes. See Section 9.6.2, “Enabling Zone Refreshes and Persistent Searches”.
9.4.2. Modifying DNS Zones

A zone is created with a certain amount of configuration, set to default values.

Example 9.1. Default DNS Zone Entry Settings

```
dn: idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: example.com
idnssoammname: server.example.com.
idnssoarname: root.server.example.com.
idnssoaserial: 2011130701
idnssoarefresh: 3600
idnssoaretry: 900
idnssoaexpire: 1209600
idnssoaminimum: 3600
idnsupdatepolicy: grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self * AAAA;
idnszoneactive: TRUE
idnsallowdynupdate: TRUE
nsrecord: server.example.com.
objectclass: top
objectclass: idnsrecord
objectclass: idnszone
```

All of the possible zone settings are listed in Table 9.1, “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 9.1. Zone Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone name</td>
<td>--name</td>
<td>Sets the name of the zone.</td>
</tr>
<tr>
<td>Authoritative nameserver</td>
<td>--name-server</td>
<td>Sets the fully-qualified domain name of the DNS name server.</td>
</tr>
<tr>
<td>Administrator e-mail address</td>
<td>--admin-email</td>
<td>Sets the email address to use for the zone administrator. This defaults to the root account on the host.</td>
</tr>
<tr>
<td>SOA serial</td>
<td>--serial</td>
<td>Sets a version number for the SOA record file.</td>
</tr>
<tr>
<td>SOA refresh</td>
<td>--refresh</td>
<td>Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.</td>
</tr>
<tr>
<td>SOA retry</td>
<td>--retry</td>
<td>Sets the time, in seconds, to wait before retrying a failed refresh operation.</td>
</tr>
<tr>
<td>SOA expire</td>
<td>--expire</td>
<td>Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.</td>
</tr>
<tr>
<td>SOA minimum</td>
<td>--minimum</td>
<td>Sets the minimum amount of time, in seconds, that data are kept in cache.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Command-Line Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SOA time to live</td>
<td>--ttl</td>
<td>Sets the maximum time, in seconds, that information is kept in the data cache.</td>
</tr>
<tr>
<td>SOA class</td>
<td>--class</td>
<td>Sets the type of record. This is almost always IN, which stands for Internet.</td>
</tr>
<tr>
<td>BIND update policy</td>
<td>--update-policy</td>
<td>Sets the permissions allowed to clients in the DNS zone.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>IMPORTANT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this is set to false, FreeIPA client machines will not be able to add or update their IP address. See Section 9.8, &quot;Enabling Dynamic DNS Updates&quot; for more information.</td>
</tr>
<tr>
<td>Dynamic update</td>
<td>--dynamic-update=TRUE</td>
<td>Enables dynamic updates to DNS records for clients.</td>
</tr>
<tr>
<td>Name server</td>
<td>--ip-address</td>
<td>Adds the DNS name server by its IP address.</td>
</tr>
<tr>
<td>Allow transfer</td>
<td>--allow-transfer=string</td>
<td>Gives a semi-colon-separated listed of IP addresses or network names which are allowed to transfer the given zone.</td>
</tr>
<tr>
<td>Allow query</td>
<td>--allow-query</td>
<td>Gives a semi-colon-separated listed of IP addresses or network names which are allowed to issue DNS queries.</td>
</tr>
<tr>
<td>Allow PTR sync</td>
<td>--allow-sync-ptr=TRUE</td>
<td>Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.</td>
</tr>
<tr>
<td>Zone forwarders</td>
<td>--forwarder=string</td>
<td>Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the FreeIPA domain. To specify multiple forwarders, use the option multiple times.</td>
</tr>
</tbody>
</table>
Chapter 9. Identity: Managing DNS

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward policy</td>
<td>--forward-policy=only</td>
<td>first</td>
</tr>
</tbody>
</table>

9.4.2.1. Editing the Zone Configuration in the Web UI

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

2. Click the name of the DNS zone to edit.

3. Open the **Settings** tab.

4. Change any of the DNS zone settings. The full list of attributes is described in *Table 9.1, “Zone Attributes”*. There are some common attributes to change:
   - **Authoritative name server**, the fully-qualified domain name of the DNS name server.
   - **Dynamic update**, to enable dynamic updates to DNS records for clients.
   - **SOA refresh**, the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.
Modifying DNS Zones

DNS Zones » example.com

DNS ZONE: example.com

DNS ZONE SETTINGS

Zone name: example.com
Status: Enabled
Authoritative nameserver: example.com
Administrator e-mail address: hostmaster.example.com
SOA serial: 2012130201
SOA refresh: 3600
SOA retry: 900
SOA expire: 1209600
SOA minimum: 3600
SOA time to live:
SOA class:
Dynamic update: False
BIND update policy:
5. Click the **Update** link at the top of the settings page.

### 9.4.2.2. Editing the Zone Configuration in the Command Line

The zone can be created with additional attributes and values different from the default by passing additional options with the `dnszone-add` command. Likewise, attributes can be added or modified in the zone entry by passing the same attribute options with the `dnszone-mod` command. These are listed in *Table 9.1, "Zone Attributes".*

If an attribute does not exist in the DNS zone entry, than the `dnszone-mod` command adds the attribute. If the attribute exists, then it overwrites the current value with the specified value.

For example, to set a time to live for SOA records:

```
$ ipa dnszone-mod server.example.com --ttl=1800
```

This adds a new attribute to the DNS zone entry:

```
dn: idnsname=example.com,cn=dns,dc=example,dc=com
idnsname: example.com
idnssoamname: server.example.com.
idnssoarnname: root.server.example.com.
idnszoserial: 2011130701
idnssoarefresh: 3600
idnssoarettry: 000
idnssoarexpire: 1209600
idnssoaminimum: 3600
**dnsttl: 1800**
idnsupdaterecords: grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self * AAAA;
idnszoneactive: TRUE
idnsallowdynupdate: TRUE
nsrecord: server.example.com.
objectclass: top
objectclass: idnsrecord
objectclass: idnszone
```

### 9.4.3. Enabling and Disabling Zones

Active zones can have clients added to them, are available for lookups, and are used by FreeIPA services like Kerberos. Deleting a DNS zone removes the zone entry and all the associated configuration.

There can be situations when it is necessary to remove a zone from activity without permanently removing the zone. This is done by *disabling* the zone.

#### 9.4.3.1. Disabling Zones in the Web UI

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone to edit.
3. Open the **Settings** tab.

4. Scroll down to the **Active zone** field. To disable the zone, set the value to **Disabled**.
5. Click the **Update** link at the top of the settings page.

**9.4.3.2. Disabling Zones in the Command Line**

Disabling a zone is done by using the `dnszone-disable` command.

For example:

```
$ ipa dnszone-disable server.example.com
```

When the zone needs to be brought back online, it can be re-enabled using the `dnszone-enable` command.

**9.5. Managing DNS Record Entries**

**9.5.1. Adding Records to DNS Zones**
FreeIPA supports several different types of DNS records, listed in Table 9.2, “DNS Record Types”.

![Table 9.2. DNS Record Types](image)

9.5.1.1. Adding DNS Resource Records from the Web UI

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the named service or configure the BIND service to poll the Directory Server automatically for zone changes. See Section 9.6.2, “Enabling Zone Refreshes and Persistent Searches”.

1. Open the Identity tab, and select the DNS subtab.
2. Click the name of the DNS zone to which to add records.
3. In the DNS Resource Records tab, click the Add link.
4. Select the type of record to create in the **Record Type** drop-down menu. The required data is different, depending on the record type. For example, a CNAME record requires a hostname. The data field name updates automatically to indicate what kind of information to supply.

Although FreeIPA supports many different record types, there are four frequent record types that are used:

- **A**. This is a basic map for a hostname and an ordinary IPv4 address. The **Record Name** is a hostname, such as `www`. The **IP Address** value is a standard IPv4 address, such as `192.168.1.2`. 
More information about A records is in RFC 1035\(^2\).

- **AAAA.** This is a basic map for a hostname and an IPv6 address. The **Record Name** is a hostname, such as `www`. The **IP Address** value is a standard hexadecimal IPv6 address, such as `fe80::20c:29ff:fe02:a1b3`.
  
  More information about AAAA records is in RFC 3596\(^3\).

- **SRV.** Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. The **Record Name** has the format `_service._protocol`, such as `_ldap._tcp`. There are individual fields to set the priority, weight, port number, and hostname for the target service.
  
  More information about SRV records is in RFC 2782\(^4\).

- **PTR.** A pointer record type (PTR) record adds a reverse DNS record, which maps an IP address to a domain name. In this case, the **Record Name** is the record ID number for the DNS entry of the resource and the **Hostname** value is the hostname with a terminal period, such as `server.example.com`.
  
  More information about PTR records is in RFC 1035\(^5\).

5. Click the **Add** button to save the new resource record.

### 9.5.1.2. Adding DNS Resource Records from the Command Line

The same script, `ipa dnsrecord-add`, is used to add resource records of any type, but the options for the script and the required data are different, based on the resource record type.

#### 9.5.1.2.1. About the Commands to Add DNS Records

The `ipa dnsrecord-add` command adds records to DNS zones, based on the type. Adding a record has the same basic command format:

```
$ ipa dnsrecord-add zoneName recordName --recordType-option=data
```

The `zoneName` is the name of the DNS zone to which the record is being added. The `recordName` is an identifier for the new DNS resource record.

*Table 9.3, “Common dnsrecord-add Options”* lists options for the most common resource record types: A (IPv4), AAAA (IPv6), SRV, and PTR. The options for all of the supported records (*Table 9.2, “DNS Record Types”*) are listed in Section C.3.2, “ipa dnsrecord-add”.

**NOTE**

The `ipa dnsrecord-add` command only creates forward entries, not reverse entries.

<table>
<thead>
<tr>
<th>General Record Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Option</strong></td>
</tr>
<tr>
<td>-------------</td>
</tr>
</tbody>
</table>
### General Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ttl=number</td>
<td>Sets the time to live for the record.</td>
</tr>
<tr>
<td>--class=IN</td>
<td>CS</td>
</tr>
<tr>
<td>--structured</td>
<td>Parses the raw DNS records and returns them in a structured format.</td>
</tr>
</tbody>
</table>

### "A" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--a-rec=ARECORD</td>
<td>Passes a comma-separated list of A records.</td>
</tr>
<tr>
<td>--a-ip-address=string</td>
<td>Gives the IP address for the record.</td>
</tr>
</tbody>
</table>

### "AAAA" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--aaaa-rec=AAAARECORD</td>
<td>Passes a comma-separated list of AAAA (IPv6) records.</td>
</tr>
<tr>
<td>--aaaa-ip-address=string</td>
<td>Gives the IPv6 address for the record.</td>
</tr>
</tbody>
</table>

### "PTR" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ptr-rec=PTRRECORD</td>
<td>Passes a comma-separated list of PTR records.</td>
</tr>
<tr>
<td>--ptr-hostname=string</td>
<td>Gives the hostname for the record.</td>
</tr>
</tbody>
</table>

### "SRV" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--srv-rec=SRVRECORD</td>
<td>Passes a comma-separated list of SRV records.</td>
</tr>
<tr>
<td>--srv-priority=number</td>
<td>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.</td>
</tr>
<tr>
<td>--srv-weight=number</td>
<td>Sets the weight of the record. This helps determine the order of SRV records with the same priority.</td>
</tr>
<tr>
<td>--srv-port=number</td>
<td>Gives the port for the service on the target host.</td>
</tr>
<tr>
<td>--srv-target=string</td>
<td>Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.</td>
</tr>
</tbody>
</table>
9.5.1.2.2. Examples of Adding DNS Resource Records

**TIP**

To make new resource records immediately resolvable without restarting the name service, enable persistent searches with the named service or configure the BIND service to poll the Directory Server automatically for zone changes. See Section 9.6.2, “Enabling Zone Refreshes and Persistent Searches”.

Example 9.2. IPv4 Record
Type A resource records map hostnames to IPv4 addresses. The record value for these commands, then, is a standard IPv4 address. The URL label is usually www.

```sh
$ ipa dnsrecord-add example.com www --a-rec 10.64.14.165
```

This creates the record www.example.com with the IP address 10.64.14.165.

More information about A records is in RFC 1035.

Example 9.3. IPv6 Record
Type AAAA resource records (quad-A records) map hostnames to IPv6 addresses. The record value for these commands is an IPv6 address. As with Type A records, the URL label is usually www.

```sh
$ ipa dnsrecord-add example.com www --aaaa-rec fe80::20c:29ff:fe02:a1b3
```

This creates the record www.example.com with the IP address fe80::20c:29ff:fe02:a1b3. More information about AAAA records is in RFC 3596.

Example 9.4. SRV Record
Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the DNS server which manages it.

As with Type A and Type AAAA records, SRV records specify a way to connect to and identify the service, but the record format is different.

The recordName identifies the service type and the connection protocol, in the format _service._protocol.

---

6 http://tools.ietf.org/html/rfc1035
The record information has the format "priority weight port target".

```
$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="0 100 389 server1.example.com."
$ ipa dnsrecord-add server.example.com _ldap._tcp --srv-rec="1 100 389 server2.example.com."
```

More information about SRV records is in RFC 2782.

**Example 9.5. PTR Record**

A pointer record type (PTR) record adds a reverse DNS record, which maps an IP address to a domain name, rather than the other way around.

All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the `in-addr.arpa` domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the `in-addr.arpa` domain appended to it. For example, for the IP address 192.0.1.2, the reverse address is 2.1.0.192.in-addr.arpa.

When adding the reverse DNS record, the format of the `dnsrecord-add` command is also reverse, compared to the usage for adding regular DNS entries:

```
$ ipa dnsrecord-add reverseIpAddress recordId --ptr-rec FQDN
```

The `recordId` is the numeric identifier to use for the entry in the zone.

For example, this adds a record with an ID of 4 for `server2.example.com`:

```
$ ipa dnsrecord-add 2.1.0.192.in-addr.arpa 4 --ptr-rec server2.example.com.
```

More information about PTR records is in RFC 1035.

**NOTE**

Reverse zones can also be configured for IPv6 addresses, with zones in the `.ip6.arpa` domain. For more information about IPv6 reverse zones, see RFC 3596.

### 9.5.2. Deleting Records from DNS Zones

#### 9.5.2.1. Deleting Records with the Web UI

To delete only a specific record from the resource record:

---

9 http://tools.ietf.org/html/rfc1035#page-12
10 http://www.ietf.org/rfc/rfc3596.txt
1. Open the **Identity** tab, and select the **DNS** subtab.

2. Click the name of the DNS zone.

3. In the **DNS Resource Records** tab, click the name of the resource record.

4. Click the checkbox by the name of the record type to delete, and then click the active **Delete** link at the top of the list.
This deletes only that record type while leaving the other configuration intact.

Alternatively, delete all of the records for the resource in the zone:

1. Open the **Identity** tab, and select the **DNS** subtab.
2. Click the name of the DNS zone.
3. In the DNS Resource Records tab, select the checkbox by the name of the resource record to delete. This deletes the entire record.

4. Click the Delete link at the top of the zone records page.
9.5.2.2. Deleting Records with the Command Line
Records are removed from the zone using the `ipa dnsrecord-del` command. As with adding records, records are deleted using an option that specifies the type of record (`--recordType-rec`) and the record value.

For example, to remove the A type record:

```bash
$ ipa dnsrecord-del example.com www --a-rec 10.64.14.213
```

If you run the `ipa dnsrecord-del` command without any options, the command prompts for information about the record to delete.

Alternatively, using the `-del-all` option removes all associated records for the zone.

9.6. Configuring the bind-dyndb-ldap Plug-in
The `bind-dyndb-ldap` system plug-in contains a DNS record cache for zones and a history of successful DNS resolutions. Maintaining the cache improves lookup performance in the Directory Server because it is not necessary to query the directory services every time there is a new DNS request.

When this plug-in is installed and FreeIPA is configured to manage DNS, then a new configuration section is added to the plug-in configuration.

```
Example 9.6. Default dynamic-db Configuration

dynamic-db "ipa" {
    library "ldap.so";
    arg "uri ldapi://%2fvar%2frun%2fslapd-EXAMPLE-COM.socket"
    arg "base cn=dns, dc=example,dc=com"
    arg "fake_mname server.example.com"
    arg "auth_method sasl"
    arg "sasl_mech GSSAPI"
    arg "sasl_user DNS/server.example.com"
};
```

This configuration uses implied default values for other plug-in behaviors, like how long it maintains the cache. The assumed, default configuration can be changed by adding arguments to the `dynamic-db "ipa"` entry.

```
arg "argument value"
```

The additional parameters are listed in `Table 9.4, "Additional bind-dyndb-ldap Configuration Parameters"`.

**NOTE**
Both cache updates and new zone detection can be forced by reloading the name server:

```
# rndc reload
```
### 9.6.1. Changing the DNS Cache Setting

To improve DNS performance, it may be necessary to change the cache setting. By default, DNS records are kept in cache and considered valid for 120 seconds. This means that if a DNS record changes, it will not (necessarily) be propagated to the name server for up to 120 seconds. If the Directory Server has a high traffic volume or if records do not change frequently, then the cache time can be increased to improve performance by adding the `cache_ttl` parameter.

```plaintext
dynamic-db "ipa" {
  ...
  arg "cache_ttl 1800";
};
```

### 9.6.2. Enabling Zone Refreshes and Persistent Searches

The DNS service receives its information through the `bind-dyndb-ldap` plug-in. The plug-in resolves only zones which were configured and enabled in the Directory Server when the name server started. When the name service restarts, the plug-in reloads its configuration and identifies any new zones or any new resource records.

However, the `bind-dyndb-ldap` plug-in pulls zone and resource record information from the FreeIPA LDAP directory, and it is possible to pull information from that directory apart from simply restarting the plug-in. The `bind-dyndb-ldap` plug-in search for zone changes actively either by refreshing the zone data or by keeping a persistent connection open to the Directory Server and immediately catching any changes.

Periodically checking for new zones is the same as refreshing the zone configuration. This is set in the `zone_refresh` argument.

```plaintext
dynamic-db "ipa" {
  ...
  arg "zone_refresh 30";
};
```
Alternatively, the plug-in can maintain an open connection to the server through a persistent search. Persistent searches provide immediate notification of changes, unlike polling, and maintain a local cache of the configuration data.

**NOTE**

A persistent search catches updates both to zones and to zone resource records.

Persistent searches are disabled by default but can be enabled in the `psearch` argument:

```
dynamic-db "ipa" {
  ...  
  arg "psearch yes";
};
```

Because persistent searches leave an ongoing, open connection with the Directory Server, there can be some performance issues. Performance implications are covered in the *389 Directory Server Administrator's Guide*.

### 9.7. Changing Recursive Queries Against Forwarders

The `ipa-client-install` script sets a configuration statement in the `/etc/named.conf` file that allows name resolution against hosts that are outside the FreeIPA DNS domain. (This requires that the FreeIPA server be set up with DNS configured and with forwarders configured.) What this means is that any host is permitted to issue recursive queries against configured forwarders.

By default, any host is permitted to issue recursive queries against configured forwarders. The FreeIPA installation script automatically adds a line to the `/etc/named.conf` file to allow these recursive queries:

```
forward first;
forwarders { 10.16.36.29; }
allow-recursion { any; }
```

This behavior can be changed in the `allow-recursion` statement.

1. Open the `/etc/named.conf` file.

2. Reset the `allow-recursion` statement. This is set to `any` by default, which allows all hosts to resolve names against all forwarders.

```
forward first;
forwarders { 10.16.36.29; }
allow-recursion { any; }
```

3. Restart the `named` service.

```
service named restart
```

---

Enabling Dynamic DNS Updates

The name server documentation has more details on editing configuration statements.

9.8. Enabling Dynamic DNS Updates

Dynamic DNS updates are not enabled by default for new DNS zones in FreeIPA. If dynamic updates are not allowed, then it may not be possible for the `ipa-client-install` script to join a client to the domain because it cannot add a DNS record pointing to the new client.

9.8.1. Enabling Dynamic DNS Updates in the Web UI

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

2. Click the name of the DNS zone to edit.

3. Open the **Settings** tab.

4. Scroll down to the **Dynamic update** field, and set the value to True.
5. Click the **Update** link at the top of the settings page.

### 9.8.2. Enabling Dynamic DNS Updates in the Command Line

To allow dynamic updates to the DNS zones, set the `-dynamic-update` option.

```
$ ipa dnszone-mod server.example.com --dynamic-update=TRUE
```

### 9.9. Configuring Forwarders and Forward Policy

A **DNS forwarder** is a server which passes DNS queries on to another, external DNS name server for resolution. Within the FreeIPA DNS domain, there are three configuration properties that define how forwarders are used:

- A list of global forwarders which are used by all zones in FreeIPA
- A list of forwarders which are used by a single, specific zone (as part of the zone configuration)
- A policy which defines how the zone sends requests to the forwarders

#### 9.9.1. Configuring Global Forwarders

Global forwarders are configured as part of the FreeIPA server configuration itself. Forwarders are (optionally) set up when the server is installed with the `setup-dns` option or when the `ipa-dns-install` script is used.

After server configuration, the list of global forwarders can be edited using the `dnsconfig-mod` command. For example:

```
[jsmith@server ~]$ ipa dnsconfig-mod --forwarder=0.9.8.7
Global forwarders: 0.9.8.7
```

#### 9.9.2. Configuring Zone Forwarders

Forwarders can be configured to be used with a specific DNS zone as part of the zone configuration. The `--forwarder` option sets a semi-colon-separated list of forwarders to use with the zone.

For example:

```
[jsmith@server ~]$ ipa dnszone-mod --forwarder=1.2.3.4;255.255.255.255 example.com
  Zone name: example.com
  ...  
  Zone forwarders: 1.2.3.4;255.255.255.255
```

**NOTE**

DNS forwarders must be specified as IP addresses, not as hostnames.

#### 9.9.3. Configuring Forwarder Policy for a Zone

Once forwarders are configured, there are different ways that the zone can use them to service requests.
Chapter 9. Identity: Managing DNS

The zone can use the forwarders only for servicing name resolution requests; this is called a forward-only zone. A forward-only zone does not check its own name records. Only the forwarder server records are checked. If the record does not exist on the configured forwarders, then the zone returns a negative response to the client.

Alternatively, the zone can check the forwarder records first, and then fallback on its own resource records. This has a first policy.

This configuration is set in the --forward-policy option, using a policy of either only or first. For example:

```
[jsmith@server ~]$ ipa dnszone-mod --forward-policy=only example.com
```

```
   Zone name: example.com
   ...
   Zone forwarders: 1.2.3.4;255.255.255.255
   Forward policy: only
```

9.10. Enabling Zone Transfers

Name servers maintain authoritative data for the zones; as changes are made to the zones, those changes must be sent to and distributed among the name servers for the DNS domain. A zone transfer moves resource records from one name server to another. An authoritative transfer (AXFR) is a zone transfer which includes that authoritative data for the zone (as opposed to an incremental transfer, which only delivers the most immediate zone change).

Zone transfers are defined in RFC 1034 and RFC 5936.

Zone transfers can be enabled when the zone is created or when it is modified by using the --allow-transfer option to set a list of name servers to which the zone records can be transferred.

For example:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-transfer=255.255.255.255;0.0.0.0;1.2.3.4 example-zone
```

The default is any, which the zone to be transferred anywhere in the DNS domain.

Once it is enabled in the bind service, FreeIPA DNS zones can be transferred, by name, by clients like dig:

```
[root@server ~]# dig @ipa-server zone_name AXFR
```

9.11. Defining DNS Queries

To resolve hostnames within the DNS domain, a DNS client issues a query to the DNS name server. For some security contexts or for performance, it may be advisable to restrict what clients can query DNS records in the zone.

DNS queries can be configured when the zone is created or when it is modified by using the --allow-query option to set a list of clients which are allowed to issue queries.

---

12 http://www.faqs.org/rfcs/rfc1034.html
13 http://www.faqs.org/rfcs/rfc5936.html
For example:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-query=255.255.255.255;0.0.0.0;1.2.3.4 example-zone
```

The default is any, which allows the zone to be queried by any client.

### 9.12. Synchronizing Forward and Reverse Zone Entries

Forward entries (A and AAAA) are configured separately from reverse entries (PTR). Because these entries are configured independently, it is possible for forward entries to exist without corresponding reverse entries, and vice versa.

A DNS zone can be configured to allow its forward and reverse entries to be synchronized automatically, by setting the `--allow-sync-ptr` option to true. This can be done when the zone is created or when it is edited.

For example, for editing an existing entry:

```
[jsmith@server ~]$ ipa dnszone-mod --allow-sync-ptr example-zone
```

The default is false, which disables synchronization and has better server performance.


The FreeIPA DNS domain can define access controls, based on grant/deny rules, for zones. This creates an `update-policy` statement in the `/etc/named.conf` file, which defines the DNS access rule.

```
--update-policy "grant|deny zoneName policyName recordName recordType"
```

- **zoneName** is the FreeIPA DNS zone to which to apply the rule.
- **policyName** is the name to use for the BIND rule.
- **recordName** sets the resource records to which to apply the rule. Using an asterisk (*) is used for self rules.
- **recordType** is the record type the rule applies to. Update access rules are applied individually for each record type, even within the same DNS zone entry.

For example, to grant the **EXAMPLE.COM** zone the ability to edit its own A and AAAA resource record entries:

```
$ ipa dnszone-mod example.com --update-policy="grant EXAMPLE.COM krb5-self * A; grant EXAMPLE.COM krb5-self * AAAA;"
```

**IMPORTANT**

If the update policy is set to false, FreeIPA client machines will not be able to add or update their IP address. See **Section 9.8, “Enabling Dynamic DNS Updates”** for more information.

It is possible to check the DNS entries for FreeIPA domain members using the `dns-resolve` command. If the record exists and is properly formatted in the DNS configuration, then the command returns the DNS record. If not, the command returns an error, that the hostname is not recognized within the DNS service.

```bash
$ipa dns-resolve server1.example.com
```

This can be helpful with troubleshooting connection problems between servers, clients, and services.

9.15. Changing Load Balancing for FreeIPA Servers and Replicas

As Section 1.3.1, “About FreeIPA Servers and Replicas” touches on, FreeIPA 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:

```bash
$ 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\(^\text{14}\).

---

Policy: Using Automount

Automount is a way of making directories on different servers available, automatically, when requested by a user. This works exceptionally well within an FreeIPA domain since it allows directories on clients within the domain to be shared easily. This is especially important with user home directories (Section 5.1, “Setting up User Home Directories”).

In FreeIPA, automount works with the internal LDAP directory and, if it is configured, DNS services.

10.1. About Automount and FreeIPA

Automount is a way to manage, organize, and access directories across multiple systems. Automount automatically mounts a directory whenever that resource is requested. Automount also provides a coherent structure to the way that these directories are organized. Every single directory, or mount point is called a key. Multiple keys that are grouped together are a map, and maps are associated according to their physical or conceptual location.

The base configuration file for autofs is the auto.master file in the /etc/ directory. There can be multiple auto.master configuration files in separate server locations, if necessary.

When autofs is configured on a server and that server is a client in a FreeIPA domain, then all of the configuration information for automount is stored in the FreeIPA directory. Rather than being stored in separate text files, the autofs configuration — maps, locations, and keys — are stored as LDAP entries. For example, the default map file, auto.master, is stored as:

dn: automountmapname=auto.master,cn=default,cn=automount,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master

IMPORTANT

FreeIPA does not set up or configure autofs. That must be done separately. FreeIPA works with an existing autofs deployment.

Each new location is added as a container entry under cn=automount,dc=example,dc=com, and each map and each key are stored beneath that location.

As with other FreeIPA domain services, automount works with FreeIPA natively. The automount configuration can be managed by FreeIPA tools:

• Locations, using ipa automountlocation* commands
• Both direct and indirect maps, using ipa automountmap* commands
• Keys, using ipa automountkey* commands
For automount to work within the FreeIPA domain, the NFS server must be configured as a FreeIPA client. Configuring NFS itself is covered in the *Red Hat Enterprise Linux Storage Administration Guide*.

## 10.2. Configuring Automount

### IMPORTANT

FreeIPA does not set up or configure autofs. That must be done separately, as described in these procedures. FreeIPA works with an existing autofs deployment.

### TIP

Test that the `/home` directory can be mounted from the command line successfully before changing the automount configuration. Making sure that NFS is already working properly makes it easier to troubleshoot any potential FreeIPA automount configuration errors later.

### 10.2.1. Configuring autofs on Fedora

1. Edit the `/etc/sysconfig/autofs` file to specify the schema attributes that autofs searches for:

   ```
   # Other common LDAP naming
   #
   MAP_OBJECT_CLASS="automountMap"
   ENTRY_OBJECT_CLASS="automount"
   MAP_ATTRIBUTE="automountMapName"
   ENTRY_ATTRIBUTE="automountKey"
   VALUE_ATTRIBUTE="automountInformation"
   ```

2. Specify the LDAP configuration. There are two ways to do this. The simplest is to let the automount service discover the LDAP server and locations on its own:

   ```
   LDAP_URI="ldap:///dc=example,dc=com"
   ```

   Alternatively, explicitly set which LDAP server to use and the base DN for LDAP searches:

   ```
   LDAP_URI="ldap://ipa.example.com"
   SEARCH_BASE="cn=location,cn=automount,dc=example,dc=com"
   ```

---

3. Edit the `/etc/autofs_ldap_auth.conf` file so that autofs allows client authentication with the FreeIPA LDAP server. Change `authrequired` to yes and set the principal to the Kerberos host principal:

```xml
<autofs_ldap_sasl_conf
  usetls="no"
  tlsrequired="no"
  authrequired="yes"
  authtype="GSSAPI"
  clientprinc="host/server.example.com@EXAMPLE.COM"
/>
```

If necessary, run `klist -k` to get the exact host principal information.

4. Check the `/etc/nsswitch.conf` file, so that LDAP is listed as a source for automount configuration:

```
automount: files ldap
```

5. Restart autofs:

```
# service autofs restart
```

6. Test the configuration by listing a user's `/home` directory:

```
# ls /home/userName
```

If this does not mount the remote file system, check the `/var/log/messages` file for errors. If necessary, increase the debug level in the `/etc/sysconfig/autofs` file by setting the `LOGGING` parameter to `debug`. 

---

**Note**

The default value for `location` is `default`. If additional locations are added (Section 10.5, “Configuring Locations”), then the client can be pointed to use those locations, instead.
Chapter 10. Policy: Using Automount

TIP

If there are problems with automount, then cross-reference the automount attempts with the 389 Directory Server access logs for the FreeIPA 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.

10.2.2. Configuring Automount on Solaris

NOTE

Solaris uses a different schema for autofs configuration than the schema used by FreeIPA. FreeIPA uses the 2307bis-style automount schema which is defined for 389 Directory Server (and used in FreeIPA's internal Directory Server instance).

1. If the NFS server is running on Fedora, 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,dc=example,dc=com
   -a serviceSearchDescriptor=group:cn=groups,dc=example,dc=com
   -a serviceSearchDescriptor=auto_mast:automountMapName=auto.master,cn=location
   -a serviceSearchDescriptor=auto_home:automountMapName=auto_home,cn=location
   -a objectClassMap=shadow:shadowAccount=posixAccount
   -a searchTimelimit=15
   -a bindTimeLimit=5
   ```

3. Enable `automount`:

   ```
   # svcadm enable svc:/system/filesystem/autofs
   ```

4. Test the configuration.
Section 10.3. Setting up a Kerberized NFS Server

FreeIPA can be used to set up a Kerberized NFS server.

**NOTE**

The NFS server does not need to be running on Fedora.

### 10.3.1. Setting up a Kerberized NFS Server

1. Obtain a Kerberos ticket before running FreeIPA tools.

   `[jsmith@server ~]$ kinit admin

2. If the NFS host machine has not been added as a client to the FreeIPA domain, then create the host entry. See Section 6.2, “Adding Host Entries”.

3. Create the NFS service entry in the FreeIPA domain. For example:

   `[jsmith@server ~]$ ipa service-add nfs/nfs-server.example.com

   For more information, see Section 6.5.1, “Adding and Editing Service Entries and Keytabs”.

4. Generate an NFS service keytab for the NFS server using the `ipa-getkeytab` command.

   The NFS server may be on a Fedora machine in the FreeIPA domain or a different Unix machine. For a Fedora machine, the `ipa-getkeytab` command can be run on the NFS server machine. Otherwise, the `ipa-getkeytab` command should be run on a Fedora machine in the FreeIPA domain and then copied over to the NFS server.

   If `ipa-getkeytab` command is run on the NFS server, then save the keys directly to the host keytab. For example:

   For a Fedora machine, that's all you need to do.

```bash
# ldapclient -l auto_master
dn: automountKey=/
    home,automountmapname=auto.master,cn=location,cn=automount,dc=example,dc=com
objectClass: automount
objectClass: top
automountKey: /home
automountInformation: auto.home

# ls /home/userName

[jsmith@server ~]$ kinit admin

[jsmith@server ~]$ ipa service-add nfs/nfs-server.example.com

For more information, see Section 6.5.1, “Adding and Editing Service Entries and Keytabs”.

[jsmith@server ~]$ ipa-getkeytab

If ipa-getkeytab command is run on the NFS server, then save the keys directly to the host keytab. For example:

For a Fedora machine, that's all you need to do.
```
NOTE

Only DES keys are supported on Red Hat Enterprise Linux 5.

When generating keys to copy over to another system, then generate the key but do not save it in the host keytab. The key must be added separately to the keytab after it is copied to the NFS server:

a. Save the keytab to a temporary file. For example:

```
[jsmith@server ~]$ ipa-getkeytab -s server.example.com -p nfs/nfs-server.example.com -k /tmp/nfs.keytab
```

b. Copy the keytabs over to the NFS server.

c. Set the file permissions to 0700.

d. Add the service key to the keytab file.

```
[root@nfs-server ~]# ( echo rkt /tmp/nfs.keytab; echo wkt /etc/krb5.keytab) | ktpass
```

TIP

Verify that the NFS service has been properly configured in FreeIPA, with its keytab, by checking the service entry:

```
[jsmith@server ~]$ ipa service-show nfs/ipaclient2.example.com
Principal: NFS/ipaclient2.example.com@EXAMPLE.COM
Keytab: True
```

5. Install the NFS packages. For example:

```
[root@nfs-server ~]# yum install nfs-utils
```

6. Configure weak crypto support. This is required for every NFS client if any client (such as a Red Hat Enterprise Linux 5 client) in the domain will use older encryption options like DES.

a. Edit the `krb5.conf` file to allow weak crypto.

```
[root@nfs-server ~]# vim /etc/krb5.conf
allow_weak_crypto = true
```

b. Update the FreeIPA server Kerberos configuration to support the DES encryption type.
7. Edit the NFS server configuration to use NFSv4 security by uncommenting the `SECURE_NFS` line.

```
[root@nfs-server ~]# vim /etc/sysconfig/nfs
SECURE_NFS="yes"
```

8. If the NFS server and client are in different DNS domains, then configure the NFS domain.

```
[root@nfs-server ~]# vim /etc/idmapd.conf
Domain = example.com
```

9. Edit the `/etc/exports` file and add the Kerberos information:

```
/export  *(rw,sec=sys:krb5:krb5i:krb5p)
```

10. Restart the NFS server.

```
[root@nfs-server ~]# service nfs restart
```

11. Configure the NFS server as an NFS client, following the directions in Section 10.3.2, “Setting up a Kerberized NFS Client”.

### 10.3.2. Setting up a Kerberized NFS Client

1. Obtain a Kerberos ticket before running FreeIPA tools.

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

2. If the NFS client is not enrolled as a client in the FreeIPA domain, then set up the required host entries, as described in Section 6.2, “Adding Host Entries”.

3. Generate an NFS service keytab for the NFS client using the `ipa-getkeytab` command.

The NFS client may be on a Fedora machine in the FreeIPA domain or a different Unix machine. For a Fedora machine, the `ipa-getkeytab` command can be run on the NFS client machine. Otherwise, the `ipa-getkeytab` command should be run on a Fedora machine in the FreeIPA domain and then copied over to the NFS client.
Chapter 10. Policy: Using Automount

If **ipa-getkeytab** command is run on the NFS client, then save the keys directly to the host keytab. For example:

For a Fedora machine, that's all you need to do.

When generating keys to copy over to another system, then generate the key but do not save it in the host keytab. The key must be added separately to the keytab after it is copied to the NFS server:

a. Save the keytab to a temporary file. For example:

   ```bash
   [jsmith@server ~]$ ipa-getkeytab -p host/nfs-client-server.example.com@EXAMPLE.COM -k /tmp/nfs.keytab
   ```

b. Copy the keytabs over to the NFS client.

c. Set the file permissions to 0700.

d. Add the service key to the keytab file.

   ```bash
   [root@nfs-client-server ~]# ( echo rkt /root/nfs-client.keytab; echo wkt /etc/krb5.keytab) | ktutil
   ```

4. Edit the NFS common configuration to enable client-side secure NFS, by uncommenting the `SECURE_NFS` line.

   ```bash
   [root@nfs-client-server ~]# vim /etc/sysconfig/nfs
   SECURE_NFS="yes"
   ```

5. If the NFS server and client are in different DNS domains, then configure the NFS domain. The `idmapd.conf` must be the same on the NFS client as it is on the NFS server.

   ```bash
   [root@nfs-client-server ~]# vim /etc/idmapd.conf
   Domain = example.com
   ```

6. Start the GSS daemon.

   ```bash
   [root@nfs-client-server ~]# service rpcgssd start
   [root@nfs-client-server ~]# service rpcbind start
   [root@nfs-client-server ~]# service rpcidmapd start
   ```

7. Mount the directory.

   ```bash
   [root@nfs-client-server ~]# echo "$NFSERVER:/this /mnt/this nfs4 sec=krb5i,rw,proto=tcp,port=2049" >>/etc/fstab
   [root@nfs-client-server ~]# mount -av
   ```
10.4. Configuring Kerberized CIFS

While FreeIPA and Samba can be integrated together, that is not done automatically. The FreeIPA server needs to be configured to create and manage Samba groups, and then the Fedora machine can be configured to use a Kerberos-aware CIFS client.

10.4.1. Setting up Samba Groups in FreeIPA

FreeIPA is not configured to create Samba groups by default. It is possible to change the FreeIPA configuration so that groups are automatically configured as Samba groups that work with the CIFS server.

**NOTE**

FreeIPA works with a Samba file server, not a Samba domain controller.

1. Obtain the Samba Windows security ID (SID) for the Samba domain. This ID is used as part of the FreeIPA group configuration.

   ```
   [root@ipaserver ~]# net getlocalsid
   SID for EXAMPLE domain  is: S-1-2-3-4
   ```

2. Obtain a Kerberos ticket before editing the FreeIPA configuration.

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

3. Add two Samba-related object classes, `sambaSAMAccount` for users and `sambaGroupMapping` for groups, to the FreeIPA configuration entry.

   **IMPORTANT**

   The object class list is the complete list of object classes for the user and group entries. Be sure to include all *existing* object classes in the list along with the new attribute, or new entries will be created with the wrong object classes and will not work in the FreeIPA domain.

   Add `sambaSAMAccount` for users:

   ```
   $ ipa config-mod --
   userobjectclasses=top,person,organizationalperson/inetorgperson/inetuser,posixaccount,krbprincipalaux,krbticketpolicyaux,ipaobject,sambaSAMAccount
   ```

   Add `sambaGroupMapping` for groups:

   ```
   $ ipa config-mod --
   groupobjectclasses=top,groupofnames,nestedgroup,ipausergroup,ipaobject,sambaGroupMapping
   ```
Chapter 10. Policy: Using Automount

4. Unique Samba IDs must be created for groups as they are added, with the Samba file server SID used as a prefix to identify the CIFS domain. This is configured by creating a Distributed Numeric Attribute Plug-in instance in the internal 389 Directory Server instance for the FreeIPA server.

The DNA Plug-in configuration includes:

- The attribute to create on entries (\textit{dnatype}).
- The entries to add the attribute to, based on an LDAP filter (\textit{dnafilter}).
- The Samba file server SID to prepend to the attribute numbers (\textit{dnaprefix}); this include a hyphen on the end of the number.
- The directory suffix to check for matching entries (\textit{dnascope}); since this includes both users and groups, it should be the root suffix.
- The number to use to begin counting Samba IDs (\textit{dnanextvalue}).

```
[root@server ~]# ldapadd -x -D "cn=Directory Manager" -w secret

dn: cn=SambaGroupSid,cn=Distributed Numeric Assignment Plugin,cn=plugins,cn=config
objectClass: top
objectClass: extensibleObject
cn: SambaSid
dnatype: sambaSID
dnaprefix: S-1-2-3-4-
dnafilter: (|(objectclass=sambasamaccount)(objectclass=sambagroupmapping))
dnascope: dc=example,dc=com
dnanextvalue: 1
```

The DNA Plug-in is described in the \textit{389 Directory Server 9.0 Administrator's Guide}.^2

5. Every Samba groups requires a \textit{sambaGroupType} attribute. Since this value is always 4, this can be defined automatically by using a class of service (CoS) to supply the value. A CoS uses a template entry with a defined value and then automatically applies that value to all entries within the scope of the template.

a. Create the CoS definition entry in the groups subtree \textit{(cn=groups,cn=accounts,dc=example,dc=com)}, so that all the group entries are updated with the \textit{sambaGroupType} attribute.

```
[root@ipaserver ~]# ldapadd -x -D "cn=directory manager" -w secret

dn: cn=SambaCoS,cn=groups,cn=accounts,dc=example,dc=com
objectClass: top
objectClass: cosSuperDefinition
objectClass: cosPointerDefinition
cosTemplateDn: cn=SambaCoS,cn=ipaConfig,dc=etc,dc=example,dc=com
cosAttribute: sambaGroupType
```

b. Create the CoS template entry, which defines the \textit{sambaGroupType} attribute. This is done outside the groups subtree, such as the \textit{cn=ipaConfig} subtree.

```
[root@ipaserver ~]# ldapadd -x -D "cn=directory manager" -w secret

dn: cn=SambaCoS,cn=ipaConfig,dc=etc,dc=example,dc=com
changetype: add
objectclass: top
objectclass: extensibleObject
objectclass: cosTemplate
```
10.4.2. Configuring the CIFS Client

mount.cifs is described in detail in its manpage.

1. Obtain a Kerberos ticket before running FreeIPA tools.

   [j smith@server ~]$ kinit admin

2. If the CIFS client is not enrolled as a client in the FreeIPA domain, then set up the required host entries, as described in Section 6.2, “Adding Host Entries”.

3. Generate an CIFS service keytab for the CIFS client using the ipa-getkeytab command, and save the keys directly to the host keytab. For example:

   [j smith@server ~]$ ipa-getkeytab -k /etc/krb5.keytab -p host/cifs-client.example.com@EXAMPLE.COM

4. If the CIFS server and client are in different DNS domains, then configure the CIFS domain. The idmapd.conf must be the same on the CIFS client as it is on the CIFS server.

   [root@cifs-client ~]# vim /etc/idmapd.conf
   Domain = example.com

5. Start the GSS daemon.

   [root@cifs-client ~]# service rpcgssd start
   [root@cifs-client ~]# service rpcbind start
   [root@cifs-client ~]# service rpcidmapd start


   [root@cifs-client ~]# vim /etc/fstab
   //cifs-client.example.com /mnt/this cifs sec=krb5i,rw,proto=tcp,port=2049

7. Mount the directory.

   [root@cifs-client ~]# mount -t cifs

10.5. 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.

---


4 http://linux.die.net/man/8/mount.cifs
IMPORTANT

FreeIPA does not set up or configure autofs. That must be done separately. FreeIPA works with an existing autofs deployment.

10.5.1. Configuring Locations through the Web UI

1. Click the Policy tab.

2. Click the Automount subtab.

3. Click the Add link at the top of the list of automount locations.

4. Enter the name for the new location.
5. Click the **Add and Edit** button to go to the map configuration for the new location. Create maps, as described in Section 10.6.1.1, "Configuring Direct Maps from the Web UI" and Section 10.6.2.1, "Configuring Indirect Maps from the Web UI".

### 10.5.2. Configuring Locations through the Command Line

To create a map, using the **automountlocation-add** and give the location name.

```bash
$ ipa automountlocation-add location
```

For example:

```bash
$ 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:

```bash
$ ipa automountlocation-tofiles raleigh
/etc/auto.master:          /etc/auto.direct
---------------------------
/etc/auto.direct:
```

### 10.6. 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. FreeIPA 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**

FreeIPA does not set up or configure autofs. That must be done separately. FreeIPA works with an existing autofs deployment.
Chapter 10. Policy: Using Automount

10.6.1. Configuring Direct Maps
Direct maps define exact locations, meaning absolute paths, to the file mount. In the location entry, a direct map is identified by the preceding forward slash:

```
/etc/auto.direct:
/shared/man server.example.com:/shared/man
```

10.6.1.1. Configuring Direct Maps from the Web UI
1. Click the Policy tab.
2. Click the Automount subtab.
3. Click name of the automount location to which to add the map.
4. In the Automount Maps tab, click the + Add link to create a new map.
5. In pop-up window, select the **Direct** radio button and enter the name of the new map.

6. In the **Automount Keys** tab, click the **+ Add** link to create a new key for the map.
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.

### 10.6.1.2. Configuring Direct Maps from the Command Line

The key defines the actual mount point (in the key name) and any options. A map is a direct or indirect map based on the format of its key.

Each location is created with an `auto.direct` item. The simplest configuration is to define a direct mapping by adding an automount key the existing direct map entry. It is also possible to create different direct map entries.

Add the key for the direct map to the location's `auto.direct` file. The `--key` option identifies the mount point, and `--info` gives the network location of the directory, as well as any `mount` options to use. For example:

```bash
$ ipa automountkey-add raleigh auto.direct --key=/share --info="-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?
```

### 10.6.2. 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`.

#### 10.6.2.1. Configuring Indirect Maps from the Web UI

1. Click the **Policy** tab.
2. Click the **Automount** subtab.
3. Click name of the automount location to which to add the map.
4. In the **Automount Maps** tab, click the **+ Add** link to create a new map.
5. In pop-up window, select the **Indirect** radio button and enter the required information for the indirect map:

- The name of the new map
• The mount point. The Mount field sets the base directory to use for all the indirect map keys.

• Optionally, a parent map. The default parent is auto.master, but if another map exists which should be used, that can be specified in the Parent Map field.

6. Click the Add button to save the new key.

### 10.6.2.2. 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.

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.

```bash
$ ipa automountmap-add-indirect location mapName --mount=directory [--parentmap=mapName]
```

For example:

```bash
$ ipa automountmap-add-indirect raleigh auto.share --mount=/share
```

2. Add the indirect key for the mount location:

```bash
$ ipa automountkey-add raleigh auto.share --key=docs --info="ipa.example.com:/export/docs"
```

3. To verify the configuration, check the location file list using automountlocation-tofiles:

```bash
$ ipa automountlocation-tofiles raleigh /etc/auto.master: /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:
Chapter 10. Policy: Using Automount

```bash
ldapclient -a
  serviceSearchDescriptor=auto_share:automountMapName=auto.share,cn=location,cn=automount,dc=example,dc=com?
one
```

### 10.6.3. Importing Automount Maps

If there are existing automount maps, these can be imported into the FreeIPA automount configuration.

```bash
ipa automountlocation-import location map_file [--continuous]
```

The only required information is the FreeIPA 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:

```bash
$ ipa automountlocation-import raleigh /etc/custom.map
```
Chapter 11.

Policy: Defining Password Policies

All users must have a password which they use to authenticate to the Kerberos domain. FreeIPA defines and enforces rules about password complexity, password histories, and account lockouts in order to maintain security.

NOTE

FreeIPA, by default, does not expose passwords to clients, even hashed passwords, for system security.

11.1. 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 FreeIPA 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 FreeIPA 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 FreeIPA.

Password policies are assigned either globally or to groups in FreeIPA, not to individual users. The password policy is assigned a weight, so that if a user belongs to multiple groups with different password policies, the policy with the highest priority will take precedence.

The different policy attributes that can be set are listed in Table 11.1, “Password Policy Settings”.

<table>
<thead>
<tr>
<th>Configuration Property</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Password Lifetime</td>
<td>--minlife</td>
<td>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.</td>
</tr>
<tr>
<td>Configuration Property</td>
<td>Command-Line Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Maximum Password Lifetime</td>
<td>--maxlife</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Minimum Number of Character Classes     | --minclasses        | Sets the minimum number of different classes, or types, of character that must exist in a password before it is considered valid. For example, setting this value to 3 requires that any password must have characters from at least three categories in order to be approved. The default value is zero (0), meaning there are no required classes. There are six character classes:  
  • Upper-case characters  
  • Lower-case characters  
  • Digits  
  • Special characters (for example, punctuation)  
  • 8-bit characters (characters whose decimal code starts at 128 or below)  
  • Number of repeated characters  
  This weights in the opposite direction, so that if you have too many repeated characters you will not meet the quorum to satisfy the "level" expressed by krbPwdMinDiffChars. |
| Minimum Length of Password              | --minlength         | Sets the minimum number of characters for a password. The default value is eight characters.                                                                                                                                                                                                                                               |
| Password History                        | --history           | Sets the number of previous passwords that are stored and which a user is prevented from using. For example, if this is set to ten, FreeIPA prevents a user from reusing any of their |
### Configuration Property | Command-Line Option | Description
--- | --- | ---
 |  | 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.

### 11.2. Viewing Password Policies

There can be multiple password policies configured in FreeIPA. There is always a global policy, which is created with the server. Additional policies can be created for groups in FreeIPA.

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.

### 11.2.1. Viewing the Global Password Policy

The global password policy is created with the server. This applies to every user until a group-level password policy supersedes it.
Chapter 11. Policy: Defining Password Policies

The default settings for the global password policy are listed in Table 11.2, “Default Global Password Policy”.

Table 11.2. Default Global Password Policy

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max lifetime</td>
<td>90 (days)</td>
</tr>
<tr>
<td>Min lifetime</td>
<td>1 (hour)</td>
</tr>
<tr>
<td>History size</td>
<td>0 (unset)</td>
</tr>
<tr>
<td>Character classes</td>
<td>0 (unset)</td>
</tr>
<tr>
<td>Min length</td>
<td>8</td>
</tr>
<tr>
<td>Max failures</td>
<td>6</td>
</tr>
<tr>
<td>Failure reset interval</td>
<td>60</td>
</tr>
<tr>
<td>Lockout duration</td>
<td>600</td>
</tr>
</tbody>
</table>

11.2.1.1. With the Web UI

1. Click the Policy tab, and then click the Password Policies subtab.

2. All of the policies in the UI are listed by group. The global password policy is defined by the global_policy group. Click the group link.

3. The global policy is displayed.
Viewing the Global Password Policy

Password Policies » global_policy

PASSWORD POLICY: global_policy

Settings

Refresh  Reset  Update

<< 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:
Chapter 11. Policy: Defining Password Policies

11.2.1.2. With the Command Line
To view the global policy, simply run the `pwpolicy-show` command with no arguments:

```
$ ipa pwpolicy-show
Group: global_policy
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 0
Character classes: 0
Min length: 8
Max failures: 6
Failure reset interval: 60
Lockout duration: 600
```

11.2.2. Viewing Group-Level Password Policies

11.2.2.1. With the Web UI
1. Click the Policy tab, and then click the Password Policies subtab.
2. All of the policies in the UI are listed by group. Click the name of the group which is assigned the policy.
3. The group policy is displayed.
11.2.2.2. With the Command Line

For a group-level password policy, specify the group name with the command:

```bash
$ ipa pwpolicy-show examplegroup
Group: global_policy
Max lifetime (days): 90
Min lifetime (hours): 1
History size: 3
Character classes: 4
Min length: 8
Max failures: 3
Failure reset interval: 15
Lockout duration: 150
```
11.2.3. Viewing the Password Policy in Effect for a User

A user may belong to multiple groups, each with their own separate password policies. These policies are not additive. Only one policy is in effect at a time and it applies to all password policy attributes. To see which policy is in effect for a specific user, the `pwpolicy-show` command can be run for a specific user. The results also show which group policy is in effect for that user.

```
$ ipa pwpolicy-show --user=jsmith
Group: admins
  Max lifetime (days): 90
  Min lifetime (hours): 1
  History size: 0
  Character classes: 0
  Min length: 8
  Max failures: 6
  Failure reset interval: 60
  Lockout duration: 600
```

11.3. 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.

11.3.1. Creating Password Policies in the Web UI

1. Click the **Policy** tab, and then click the **Password Policies** subtab.

2. All of the policies in the UI are listed by group. The global password policy is defined by the `global_policy` group. Click the group link.
3. Click the **Add** link at the top.

4. In the pop-up box, select the group for which to create the password policy.

5. Set the priority of the policy. The higher the number, the lower the priority.
Only one password policy is in effect for a user, and that is the highest priority policy.

The priority cannot be changed in the UI once the policy is created.

6. Click the **Add and Edit** button so that the policy form immediately opens.

7. Set the policy fields. Leaving a field blank means that attribute is not added the password policy configuration.

   - **Max lifetime** sets the maximum amount of time, in days, that a password is valid before a user must reset it.

   - **Min lifetime** sets the minimum amount of time, in hours, that a password must remain in effect before a user is permitted to change it. This prevents a user from attempting to change a password back immediately to an older password or from cycling through the password history.

   - **History size** sets how many previous passwords are stored. A user cannot re-use a password that is still in the password history.

   - **Character classes** sets the different categories of character that must be used in the password. For example, a character class can be a number, special character, or capital; the complete list of categories is in Table 11.1, "Password Policy Settings". This is part of setting the complexity requirements.
Creating Password Policies with the Command Line

- **Min length** sets how many characters must be in a password. This is part of setting the complexity requirements.

### 11.3.2. Creating Password Policies with the Command Line

Password policies are added with the `pwpolicy-add` command.

```bash
$ ipa pwpolicy-add groupName --attribute-value
```

For example:

```bash
$ ipa pwpolicy-add exampleGroup --minlife=7 --maxlife=49 --history=--priority=1
Group: exampleGroup
Max lifetime (days): 49
Min lifetime (hours): 7
Priority: 1
```

**TIP**

Setting an attribute to a blank value effectively removes that attribute from the password policy.

### 11.3.3. Editing Password Policies with the Command Line

As with most FreeIPA 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
```

### 11.4. Managing Password Expirations

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 11.3, “Creating and Editing Password Policies”**.
Chapter 11. Policy: Defining Password Policies

With password expirations, 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 be done using the FreeIPA CLI with the --setattr option:

```
[bjensen@ipaserver ~]$ kinit
[bjensen@ipaserver ~]$ ipa user-mod jsmith --setattr=krbPasswordExpiration=20121231011529Z
```

If the new expiration date should be applied to multiple entries, it may be simpler to use ldapmodify and edit multiple entries simultaneously through an LDIF file in the -f option. For example, editing a single entry (with a modify statement similar to the LDIF file in -f):

```
[bjensen@ipaserver ~]$ ldapmodify -Y GSSAPI -h ipaserver.example.com -p 389 -vv
dn: uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
changetype: modify
replace: krbpasswordexpiration
krbpasswordexpiration: 20140202203734Z
```

**TIP**

If an administrator resets a password, it expires the previous password and forces the user to update the password. When the user updates the password, it automatically uses the new password policies, including a new expiration date.

### 11.5. Changing the Priority of Group Password Policies

A user may belong to multiple groups, each with different password policies. Since only one policy can be in effect for a user, there has to be a method to assign precedence to policies. That is done through priority.

The highest priority is zero (0). The lower the number, the higher the priority.

This is set initially when the password policy is created. It can be modified after the policy is created by resetting the --priority option.

```
$ ipa pwpolicy-mod examplegroup --priority=10
```

When a user belongs to multiple groups, the group password policy with the lowest number has the priority.
11.6. Setting Account Lockout Policies

A brute force attack occurs when a malefactor attempts to guess a password by simply slamming the server with multiple login attempts. An account lockout policy prevents brute force attacks by blocking an account from logging into the system after a certain number of login failures — even if the correct password is subsequently entered.

NOTE

A user account can be manually unlocked by an administrator using the `ipa user-unlock`. Refer to Section 5.5, “Unlocking User Accounts After Password Failures”.

11.6.1. In the UI

These attributes are available in the password policy form when a group-level password policy is created or when any password policy (including the global password policy) is edited.

1. Click the **Policy** tab, and then click the **Password Policies** subtab.

2. Click the name of the policy to edit.
3. Set the account lockout attribute values.
There are three parts to the account lockout policy:

- The number of failed login attempts before the account is locked (**Max Failures**).

- The time after a failed login attempt before the counter resets (**Failure reset interval**). Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally lapses after a certain amount of time. This is in seconds.

- How long an account is locked after the max number of failures is reached (**Lockout duration**). This is in seconds.

### 11.6.2. In the CLI

There are three parts to the account lockout policy:

- The number of failed login attempts before the account is locked (**--maxfail**).

- How long an account is locked after the max number of failures is reached (**--lockouttime**). This is in seconds.
• The time after a failed login attempt before the counter resets (--failinterval). Since mistakes do happen honestly, the count of failed attempts is not kept forever; it naturally lapses after a certain amount of time. This is in seconds.

These account lockout attributes can all be set when a password policy is created with pwpolicy-add or added later using pwpolicy-mod. For example:

```
[jsmith@ipaserver ~]$ kinit admin
[jsmith@ipaserver ~]$ ipa pwpolicy-mod examplegroup --maxfail=4 --lockouttime=600 --
failinterval=30
```

### 11.7. Enabling a Password Change Dialog

There may be situations when a user exists in FreeIPA but does not have a valid Kerberos ticket, meaning he cannot authenticate to the FreeIPA 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`. 
Policy: Managing the Kerberos Domain

Kerberos authentication is the core of authentication within the FreeIPA domain. The FreeIPA 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/](http://web.mit.edu/kerberos/www/).

**IMPORTANT**

FreeIPA has its own command-line tools to use to manage Kerberos policies. Do not use `kadmin` or `kadmin.local` to manage FreeIPA Kerberos settings.

12.1. 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 a FreeIPA 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 FreeIPA realm.

12.1.1. About Principal Names

The principal identifies not only the user or service, but also the realm that that entity belongs to. A principal name has two parts, the identifier and the realm:

`identifier@REALM`
For a user, the *identifier* is only the Kerberos username. For a service, the *identifier* is a combination of the service name and the hostname of the machine it runs on:

```
service/FQDN@REALM
```

The *service* name is a case-sensitive string that is specific to the service type, like `host`, `ldap`, `http`, and `dns`. Not all services have obvious principal identifiers; the sshd daemon, for example, uses the host service principal.

The host principal is usually stored in `/etc/krb5.keytab`.

When Kerberos requests a ticket, it always resolves the domain name aliases (DNS CNAME records) to the corresponding DNS address (A or AAAA records). The hostname from the address record is then used when service or host principals are created.

For example:

```
www.example.com  CNAME  web-01.example.com
web-01.example.com  A  192.0.2.145
```

A service attempts to connect to the host using its CNAME alias:

```
$ ssh www.example.com
```

The Kerberos server requests a ticket for the resolved hostname, `web-01.example.com@EXAMPLE.COM`, so the host principal must be `host/web-01.example.com@EXAMPLE.COM`.

### 12.1.2. 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`.

### 12.2. 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. FreeIPA 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.

### 12.2.1. Setting Global Ticket Policies

#### 12.2.1.1. From the Web UI

1. Click the Policy tab, and then click the Kerberos Ticket Policy subtab.
2. Change the ticket lifetime policies.

- **Max renew** sets the period after a ticket expires that it can be renewed.
- **Max life** sets the active period (lifetime) of a Kerberos ticket.

3. Click the **Update** link at the top of the policy page.

4. Restart the KDC.

```
# service krb5kdc restart
```

**IMPORTANT**

Any change to the global Kerberos ticket policy requires a restart of the KDC for the changes to take effect.

### 12.2.1.2. 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
12.2.2. 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.

12.3. Refreshing Kerberos Tickets

Kerberos keys are analogous to passwords. Like passwords, security policies may require that Kerberos tickets are manually refreshed after a specified interval.

The version of the key is shown in its *key version number* (KVNO). Refreshing (also called rotating) the principal's key increments the KVNO in the keytab entry. When a key is refreshed, a new entry is added to the keytab with a higher KVNO. The original key remains in the keytab but is no longer used to issue tickets.

Each keytab for the FreeIPA realm has an entry in the FreeIPA LDAP server, which includes its last change time. The principals which need to be refreshed can be regenerated using the `ipa-getkeytab` command.

**NOTE**

The `ipa-getkeytab` command does not delete the old keytab in case it already exists in the file.

1. Find all keytabs issued before the requisite date. For example, this looks for any principals created between midnight on January 1, 2010, and 11:59 PM on December 31, 2010:
Refreshing Kerberos Tickets

# ldapsearch -x -b "cn=computers,cn=accounts,dc=example,dc=com" "(&(krblastpwdchange>=20100101000000)(krblastpwdchange<=20101231235959))" dn krbprincipalname

# ldapsearch -x -b "cn=services,cn=accounts,dc=example,dc=com" "(&(krblastpwdchange>=20100101000000)(krblastpwdchange<=20101231235959))" dn krbprincipalname

- Host (machine) principals are stored under the `cn=computers,cn=accounts,dc=example,dc=com` subtree.

- Service principals are stored under the `cn=services,cn=accounts,dc=example,dc=com` subtree.

- Filter by the last change date (`krblastpwdchange`).

- Limit the search result information to only the entry name and principal by specifying the `dn krbprincipalname` attributes.

Dates are expressed in YYYYMMDD format, and times in HHMMSS format (GMT).

2. Retrieve a new keytab for the principal using the `ipa-getkeytab` command. This requires the location of the original keytab for the service or host (`-k`), the principal (`-p`), and the FreeIPA server hostname (`-s`).

   For example, this refreshes the host principal with a keytab in the default location of `/etc/krb5.keytab`:

   ```bash
   # 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`:

   ```bash
   # 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.

   ```bash
   # 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.
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.

### 12.4. Caching Kerberos Passwords

A machine may not always be on the same network as the FreeIPA domain; for example, a machine may need to be logged into a VPN before it can access the FreeIPA domain. If a user logs into a system when it is offline and then later attempts to connect to FreeIPA services, then the user is blocked because there is no FreeIPA Kerberos ticket for that user. FreeIPA 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 FreeIPA domain:

```ini
[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.

```ini
[domain/example.com]
    ...
    krb5_store_password_if_offline = false
```

The SSSD configuration options for Kerberos authentication is covered in the "Configuring Domains" section of the SSSD chapter in the Red Hat Enterprise Linux Deployment Guide.

### 12.5. Removing Keytabs

Refreshing Kerberos tickets adds a new key to the keytab, but it does not clear the keytab. If a host is being unenrolled and re-added to the FreeIPA 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:

```bash
# 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:

```bash
# ipa-rmkeytab -p ldap/client.example.com -k /etc/krb5.keytab
```

### 12.6. Troubleshooting Kerberos Errors

Kerberos errors frequently become apparent when trying to connect to the realm using *kinit* or a similar client. For information related to Kerberos, first check the Kerberos manpages, help files, and other resources.

**IMPORTANT**

FreeIPA has its own command-line tools to use to manage Kerberos policies. **Do not use** *kadmin* or *kadmin.local* to manage FreeIPA Kerberos settings.

There are several places to look for Kerberos error log information:

- For *kinit* problems or other Kerberos server problems, look at the KDC log in `/var/log/krb5kdc.log`.

- For FreeIPA-specific errors, look in `/var/log/httpd/error_log`.

The FreeIPA logs, both for the server and for FreeIPA-associated services, are covered in Section 18.1.3, “Checking FreeIPA Server Logs”.

#### Problems making connections with SSH when using GSS-API

If there are bad reverse DNS entries in the DNS configuration, then it may not be possible to log into FreeIPA resources using SSH. When SSH attempts to connect to a resource using GSS-API as its security method, GSS-API first checks the DNS records. The bad records prevent SSH from locating the resource.

It is possible to disable reverse DNS lookups in the SSH configuration. Rather than using reverse DNS records, SSH passes the given username directly to GSS-API.

To disable reverse DNS lookups with SSH, add or edit the `GSSAPITrustDNS` directive and set the value to `no`.

```bash
# vim /etc/ssh/ssh_config
GSSAPITrustDNS no
```

#### There are problems connecting to an NFS server after changing a keytab

Clients attempting to mount NFS exports rely on the existence of a valid principal and secret key on both the NFS server and the client host. Clients themselves should not have access to the NFS keytab. The ticket for the NFS connection will be given to clients from the KDC.
Chapter 12. Policy: Managing the Kerberos Domain

Failure to export an updated keytab can cause problems that are difficult to isolate. For example, existing service connections may continue to function, but no new connections may be possible.
Chapter 13.

Policy: Using sudo
FreeIPA provides a mechanism for predictably and consistently apply sudo policies across the FreeIPA domain. The sudo policies apply to domain users and domain hosts.

13.1. About sudo and IPA
The sudo command allows a system administrator to delegate authority to specific users to run specific commands as root or another specified user. sudo provides an audit trail of the commands and their arguments, so access can be tracked.

13.1.1. General sudo Configuration in FreeIPA
sudo uses a local configuration file, /etc/sudoers, which defines the commands and users with sudo access. While this file can be shared among machines, there's no native way to distribute sudo configuration files among machines.

FreeIPA uses its centralized LDAP database to contain the sudo configuration, which makes it globally available to all domain hosts. FreeIPA also has a specialized LDAP schema for sudo entries that allows a lot more flexible and simpler configuration. This schema adds two key features:

- The FreeIPA schema supports host groups in addition to netgroups for sudo, while sudo only supports netgroups.

  For every host group, FreeIPA also creates a corresponding shadow netgroup. This allows FreeIPA administrators to create sudo rules that reference host groups, while the local sudo command uses the corresponding netgroup.

- FreeIPA introduces the concept of a sudo command group. The group contains multiple commands, and then the command group can be referenced in the sudo configuration.

Because sudo does not support host groups and command groups, FreeIPA translates the FreeIPA sudo configuration into native sudo configuration when the sudo rules are created.

Both sudo and FreeIPA support user groups as part of the sudo configuration. User groups can be either Unix or non-POSIX groups. Creating non-POSIX groups can create some access issues because any users in the group inherit non-POSIX rights from the group. Having the choice between Unix and non-POSIX groups allows administrators the choice in group formatting and to avoid problems with inherited permissions or GID information.

13.1.2. sudo and Netgroups
As Section 13.1.1, “General sudo Configuration in FreeIPA” mentions, the LDAP schema used for sudo entries in FreeIPA supports host group-style groups in addition to netgroups. Really, FreeIPA creates two groups, a visible host group and a shadow netgroup. sudo itself only supports NIS-style netgroups for group formats.

One important thing to consider is that even though sudo uses NIS netgroups, it is not necessary to have a NIS server installed or a NIS client configured. When any group is created for sudo, the NIS object is created in the Directory Server instance, and then the information is retrieved by NSS_LDAP or by SSSD. The client (in this case, sudo) then extracts the required NIS information from the information provided by FreeIPA’s Directory Server.

In short, sudo configuration required NIS-formatted netgroups. It does not require NIS.
Chapter 13. Policy: Using sudo

The FreeIPA Directory Server instance uses the standard LDAP schema for NIS objects, defined in RFC 2307.

13.1.3. Supported sudo Clients

Any system which is supported as an FreeIPA client system can be configured as a sudo client in FreeIPA.

13.2. Setting up sudo Commands and Command Groups

Just as in regular sudo configuration, any command which will be governed by sudo access must be listed in the configuration. FreeIPA adds an extra control measure with sudo command groups, which allow a group of commands to be defined and then applied to the sudo configuration as one.

Adding a command or a command group makes it available to FreeIPA to be defined in a sudo rule; simply adding a command does not automatically include it in a sudo rule.

13.2.1. Adding sudo Commands

13.2.1.1. Adding sudo Commands with the Web UI

1. Click the Policy tab.

2. Click the Sudo subtab, and then select the Sudo Commands link.

3. Click the Add link at the top of the list of commands.

4. Enter the full system path and name of the command and, optionally, a description.
5. Click the Add and Edit button to go immediately to the edit pages for the command.

6. In the Sudo Command Groups tab, click the Add button to add the sudo command to a command group.

7. Click the checkbox by the groups for the command to join, and click the right arrows button, >>, to move the group to the selection box.

8. Click the Add button.

13.2.1.2. Adding sudo Commands with the Command Line
To add a single command, use the `sudocmd-add` command. This requires the full, local path to the command executable and a description of the command:

```
$ ipa sudocmd-add --desc "description" /local/path/to/command
```

For example:

```
$ ipa sudocmd-add --desc 'For reading log files' '/usr/bin/less'
Added sudo command "/usr/bin/less"
----------------------------------
sudo Command: /usr/bin/less
Description: For reading log files
```

13.2.2. Adding sudo Command Groups

13.2.2.1. Adding sudo Command Groups with the Web UI
1. Click the Policy tab.

2. Click the Sudo subtab, and then select the Sudo Command Groups link.

3. Click the Add link at the top of the list of command groups.
4. Enter the name and description for the new command group.

5. Click the **Add and Edit** button to go immediately to the edit pages for the group.

6. In the **Sudo Commands** tab, click the **Add** button to add a sudo command to the group.
7. In the Sudo Commands tab, click the Add button to add a sudo command to the group.

8. Click the checkbox by the names of the commands to add, and click the right arrows button, >>, to move the command to the selection box.
Chapter 13. Policy: Using sudo

13.2.2.2. Adding sudo Command Groups with the Command Line
Creating a command group requires creating two entries, one for the group and one for the command itself:

1. Create the command group using the `sudocmdgroup-add` command:

   ```bash
   $ ipa sudocmdgroup-add --desc 'File editing commands' files
   -----------------------------------
   Added sudo command group "files"
   -----------------------------------
   sudo Command Group: files
   Description: File editing commands
   ```

2. Create a command entry using the `sudocmd-add` command:

   ```bash
   $ ipa sudocmd-add --desc 'For editing files' '/usr/bin/vim'
   ----------------------------------
   Added sudo command "/usr/bin/vim"
   ----------------------------------
   ```
3. Add the command, using its full directory location as its name, to the command group using the `sudocmdgroup-add-member` command:

$$
$ ipa sudocmdgroup-add-member --sudocmds '/usr/bin/vim' files
sudo Command Group: files
Description: File editing commands
Member sudo commands: /usr/bin/vim
-------------------------
Number of members added 1
-------------------------
$$

### 13.3. Defining sudo Rules

*sudo* rules are in a sense similar to access control rules: they define users who are granted access, the commands which are within the scope of the rule, and then the target hosts to which the rule applies. In FreeIPA, additional information can be configured in the rule, such as *sudoers* options and run-as settings, but the basic elements always define who, what (services), and where (hosts).

#### 13.3.1. About External Users and Hosts

*sudo* rules define four elements: *who* can do *what*, *where*, and *as whom*. The *who* is the regular user, and the *as whom* is the system or other user identity which the user uses to perform tasks. Those tasks are system commands that can be run (or specifically not run) on a target machine.

Three of those elements — who, as whom, and where — are identities. They are uses and hosts. Most of the time, those identities are going to be entities within the FreeIPA domain because there will be overlap between the system users and machines in the environment and the users and hosts belonging to the FreeIPA domain.

However, that is not necessarily the case with all identities that a *sudo* policy may realistically cover. For example, *sudo* rules could be used to grant root access to member of the IT group in FreeIPA, and that root user is not a user in FreeIPA. Or, for another example, administrators may want to block access to certain hosts that are on a network but are not part of the FreeIPA domain.

The *sudo* rules in FreeIPA support the concept of *external* users and hosts — meaning, hosts and users which are stored and exist outside of the FreeIPA configuration.
When configuring a `sudo` rule, the user, run-as, and host settings all can point to an external identity to be included and evaluated in the `sudo` rule.

### 13.3.2. About `sudo` Options Format

The `sudo` rule can be configured to use any supported `sudoers` options. (The complete list of options is in the `sudoers` manpage and at [http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options](http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options).)

However, the `sudo` rule configuration in FreeIPA does not allow the same formatting as the configuration in the `/etc/sudoers` file. Specifically, FreeIPA does not allow whitespaces in the options parameter, whether it is set in the UI or the CLI.

For example, in the `/etc/sudoers` file, it is permissible to list options in a comma-separate list with spaces between:

```
mail_badpass, mail_no_host, mail_no_perms, syslog = local2
```

However, in FreeIPA, that same configuration would be interpreted as different arguments — including the equals sign (=) since it has spaces around it.
Likewise, linebreaks that are ignored in the `/etc/sudoers` file are not allowed in the FreeIPA configuration:

```
env_keep = "COLORS DISPLAY EDITOR HOSTNAME HISTSIZE INPUTRC
            KDEDIR LESSSECURE LS_COLORS MAIL PATH PS1 PS2
            QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE LC_COLLATE
            LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES
            LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE
            LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
            XAUTHORITY"
```

To use multiple `sudoers` options in FreeIPA, configure each one as a separate option setting, rather than all on one line, as is allowed in the `/etc/sudoers` file.

### 13.3.3. Defining sudo Rules in the Web UI

1. Click the **Policy** tab.

2. Click the **Sudo** subtab, and then select the **Sudo Rules** link.

3. Click the **Add** link at the top of the list of sudo rules.

4. Enter the name for the rule.
5. Click the **Add and Edit** button to go immediately to set the configuration for the rule.

There are a number of configuration areas for the rule. The most basic elements are set in the **Who**, **Access This Host**, and **Run Commands** areas; the others are optional and are used to refine the rule.

6. **Optional.** In the **Options** area, add any **sudoers** options. The complete list of options is in the **sudoers** manpage and at [http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options](http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options).

   **NOTE**

   As described in Section 13.3.2, "About sudo Options Format", do not use options with whitespace in the values. Rather than adding a list of options in one line, add a single option setting for each desired option.

   a. Click the + **Add** link at the right of the options list.
b. Enter the `sudoers` option.

c. Click Add.

7. In the **Who** area, select the users or user groups to which the sudo rule is applied.

   a. Click the + Add link at the right of the users list.

   b. Click the checkbox by the users to add to the rule, and click the right arrows button, >>, to move the users to the selection box.
c. Click **Add**.

It is possible to configure both FreeIPA users and external system users (**Section 13.3.1, “About External Users and Hosts”**).

8. In the **Access This Host** area, select the hosts on which the sudo rule is in effect.
   a. Click the **+ Add** link at the right of the hosts list.
b. Click the checkbox by the hosts to include with the rule, and click the right arrows button, >>, to move the hosts to the selection box.

c. Click Add.
Chapter 13. Policy: Using sudo

It is possible to configure both FreeIPA clients and external hosts (Section 13.3.1, “About External Users and Hosts”).

9. In the Run Commands area, select the commands which are included in the sudo rule. The sudo rule can grant access or deny access to commands, and it can grant allow access to one command and also deny access to another.

a. In the Allow/Deny area, click the + Add link at the right of the commands list.

b. Click the checkbox by the commands or command groups to include with the rule, and click the right arrows button, >>, to move the commands to the selection box.
c. Click **Add**.

10. *Optional.* The sudo rule can be configured to run the given commands as a specific, non-root user.

   a. In the **As Whom** area, click the + **Add** link at the right of the users list.
b. Click the checkbox by the users to run the command as, and click the right arrows button, >>, to move the users to the selection box.
c. Click Add.

### 13.3.4. Defining sudo Rules in the Command Line

Each element is added to the rule command using a different command (listed in *Table 13.1, “sudo Commands”).

The basic outline of a `sudo` rule command is:

```bash
$ ipa sudorule-add* options ruleName
```

**Example 13.1. Creating Basic sudo Rules**

In the most basic case, the *sudo* configuration is going to grant the right to one user for one command on one host.

The first step is to add the initial rule entry.

```bash
$ ipa sudorule-add files-commands
-----------------------------
Added sudo rule "files-commands"
-----------------------------
```
Next, add the commands to *grant* access to. This can be a single command, using *--sudocmd*, or a group of commands, using *--sudocmdgroups*.

```
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/vim" files-commands
Rule name: files-commands
Enabled: TRUE
sudo Commands: /usr/bin/vim
-------------------------
Number of members added 1
-------------------------
```

Add a host or a host group to the rule.

```
$ ipa sudorule-add-host --host server.example.com files-commands
Rule name: files-commands
Enabled: TRUE
Hosts: server.example.com
sudo Commands: /usr/bin/vim
-------------------------
Number of members added 1
-------------------------
```

Last, add the user or group to the rule. This is the user who is allowed to use *sudo* as defined in the rule; if no "run-as" user is given, then this user will run the *sudo* commands as root.

```
$ ipa sudorule-add-user --user jsmith files-commands
Rule name: files-commands
Enabled: TRUE
Users: jsmith
Hosts: server.example.com
sudo Commands: /usr/bin/vim"
-------------------------
Number of members added 1
-------------------------
```

**Example 13.2. Allowing and Denying Commands**

The *sudo* rule can grant access or deny access to commands. For example, this rule would allow read access to files but prevent editing:

```
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/less" readfiles
$ ipa sudorule-add-allow-command --sudocmd "/usr/bin/tail" readfiles
$ ipa sudorule-add-deny-command --sudocmd "/usr/bin/vim" readfiles
```

**Example 13.3. Using sudoers Options**

The *sudoers* file has a lot of potential flags that can be set to control the behavior of *sudo* users, like requiring (or not requiring) passwords to offer a user to authenticate to *sudo* or using fully-qualified domain names in the *sudoers* file. The complete list of options is in the *sudoers* manpage and at [http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options](http://www.gratisoft.us/sudo/sudoers.man.html#sudoers_options).

Any of these options can be set for the FreeIPA *sudo* rule using the *sudorule-add-option* command. When the command is run, it prompts for the option to add:

```
$ ipa sudorule-add-option readfiles
```
Defining sudo Rules in the Command Line

Sudo Option: !authenticate
-----------------------------------------------------
Added option "!authenticate" to Sudo rule "readfiles"
-----------------------------------------------------

NOTE
As described in Section 13.3.2, "About sudo Options Format", do not use options with whitespace in the values. Rather than adding a list of options in one line, add a single option setting for each desired option.

Example 13.4. Running as Other Users
The sudo rule also has the option of specifying a non-root user or group to run the commands as. The initial rule has the user or group specified using the --sudorule-add-runasuser or --sudorule-add-runasgroup command, respectively.

$ ipa sudorule-add-runasuser --users=jsmith readfiles
$ ipa sudorule-add-runasgroup --groups=ITadmins readfiles

When creating a rule, the sudorule-add-runasuser or sudorule-add-runasgroup command can only set specific users or groups. However, when editing a rule, it is possible to run sudo as all users or all groups by using the --runasusercat or --runasgroupcat option. For example:

$ ipa sudorule-mod --runasgroupcat=all ruleName

NOTE
The --sudorule-add-runasuser and --sudorule-add-runasgroup commands do not support an all option, only specific user or group names. Specifying all users or all groups can only be used with options with the sudorule-mod command.

Example 13.5. Referencing External Users or Hosts
The "who" in a sudo rule can be a FreeIPA user, but there are many logical and useful rules where one of the referents is a system user. Similarly, a rule may need to grant or deny access to a host machine on the network which is not an FreeIPA client.

In those cases, the sudo policy can refer to an external user or host — an identity created and stored outside of FreeIPA (Section 13.3.1, “About External Users and Hosts”).

There are three options to add an external identity to a sudo rule:

- --externaluser
- --runasexternaluser
- --externalhost
Chapter 13. Policy: Using sudo

For example:

```bash
$ ipa sudorule-add-host --externalhost=external-server.example.com readfiles
$ ipa sudorule-add-user --externaluser=ITAdmin readfiles
$ ipa sudorule-add-runasuser --runaseexternaluser=root readfiles
```

Table 13.1. sudo Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sudorule-add</td>
<td>Adds a sudo rule entry.</td>
</tr>
<tr>
<td>sudorule-add-user</td>
<td>Adds a user or a user group to the sudo rule. This user (or every member of the group) is then entitled to sudo any of the commands in the rule.</td>
</tr>
<tr>
<td>sudorule-add-host</td>
<td>Adds a target host for the rule. These are the hosts where the users are granted sudo permissions.</td>
</tr>
<tr>
<td>sudorule-add-runasgroup</td>
<td>Sets a group to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using sudo-rule.</td>
</tr>
<tr>
<td>sudorule-add-runasuser</td>
<td>Sets a user to run the sudo commands as. This must be a specific user; to specify all users, modify the rule using sudo-rule.</td>
</tr>
<tr>
<td>sudorule-add-allow-command</td>
<td>Adds a command that users in the rule have sudo permission to run.</td>
</tr>
<tr>
<td>sudorule-add-deny-command</td>
<td>Adds a command that users in the rule are explicitly denied sudo permission to run.</td>
</tr>
<tr>
<td>sudorule-add-option</td>
<td>Adds a sudoers flag to the sudo rule.</td>
</tr>
</tbody>
</table>

13.4. An Example of Configuring sudo

Implementing sudo requires setting up the command configuration on the FreeIPA server and then configuring the local sudo client to look for the appropriate configuration.

13.4.1. Server Configuration for sudo Rules

1. *Optional.* Set up a host group, as described in Section 6.10, “Managing Host Groups”.

2. *Optional.* Create a user group and add the users, as described in Section 5.9.1, “Creating User Groups”.

3. Set up a bind user by setting a password for the default FreeIPA sudo bind user.

Using LDAP tools, set the password for the sudo user,
uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com. For example:

```bash
$ ldappasswd -Y GSSAPI -S -h ipaserver.ipadocs.org
uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com
    New password: 
    Re-enter new password: 
    Enter LDAP Password:
```
4. Set up the sudo commands and command groups, as described in Section 13.2, “Setting up sudo Commands and Command Groups”.

5. Set up the sudo rules, as described in Section 13.3, “Defining sudo Rules”.

13.4.2. Client Configuration for sudo Rules

This example specifically configures a Fedora 16 client for sudo rules. The configuration file in step 4 is different, depending on the platform.

1. Configure sudo to look to LDAP for the sudoers file.

```
vim /etc/nsswitch.conf

sudoers:  files ldap
```

Leaving the files option in place allows sudo to check its local configuration before checking the LDAP-based FreeIPA configuration.

2. Enable debug logging for sudo operations in the /etc/ldap.conf file. If this file does not exist, it can be created.

```
vim /etc/ldap.conf

sudoers_debug: 1
```

**TIP**

Adding the sudoers_debug parameter helps with troubleshooting. Valid values for this parameter are 0, 1, and 2. The sudo documentation at http://www.gratisoft.us/sudo/readme_ldap.html has more information on debugging the process.

3. Optionally, enable debugging in SSSD to show what LDAP settings it is using.

```
vim /etc/sssd/sssd.conf

[domain/IPADOMAIN]
debug_level = 6
....
```

The LDAP search base used by SSSD for operations is recorded in the sssd_DOMAINNAME.log log.

4. Edit the NSS/LDAP configuration file and add the following sudo-related lines to the /etc/nslcd.conf file:

```
binddn uid=sudo,cn=sysaccounts,cn=etc,dc=example,dc=com
bindpw sudo_password

ssl start_tls
tls_cacertfile /etc/ipa/ca.crt
tls_checkpeer yes
```
Multiple LDAP servers can be configured in a space-separated list, and other options (like SSL and non-standard ports) can be used with the LDAP URL. The `sudo` LDAP configuration is covered in the `sudo` manpages, http://www.sudo.ws/sudo/man/1.8.2/sudoers.ldap.man.html.

**IMPORTANT**

The `uri` directive must give the fully-qualified domain name of the LDAP server, not an IP address. Otherwise, `sudo` fails to connect to the LDAP server.

---

5. Set a name for the NIS domain in the `sudo` configuration. `sudo` uses NIS 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 FreeIPA `sudo` configuration.

   a. Open the `/etc/rc.d/rc.local` file. Setting the NIS domain name in this file allows the value to persist between reboots.

      ```bash
      # vim /etc/rc.d/rc.local
      
      nisdomainname example.com
      ```

   b. Add the command to set the NIS domain name.

      ```bash
      nisdomainname example.com
      ```

      **IMPORTANT**

      Even though `sudo` uses NIS-style netgroups, it is not necessary to have a NIS server installed. Netgroups require that a NIS domain be named in their configuration, so `sudo` requires that a NIS domain be named for netgroups. However, that NIS domain does not actually need to exist.
Policy: Configuring Host-Based Access Control

FreeIPA can control access to both machines and the services on those machines within the FreeIPA 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 (source) are allowed to access other hosts (targets) within the domain. This access can be further broken down to users and services.

NOTE

Using host-based access control requires SSSD to be installed and configured on the FreeIPA client machine.

14.1. About Host-Based Access Control

Host-based access control rules (which are described in Chapter 14, Policy: Configuring Host-Based Access Control) can be applied to individual hosts. However, using host groups allows centralized, and potentially simplified, access control management because an access control rule only needs to be defined once and then it is applied immediately and consistently to all the hosts within the group.

Figure 14.1. Host Groups and Host-Based Access Control
Chapter 14. Policy: Configuring Host-Based Access Control

NOTE

While access must be explicitly granted to users and hosts within the FreeIPA domain, FreeIPA 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 FreeIPA 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 FreeIPA 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 FreeIPA domain user) and what they can use to access the targets (the source host).

TIP

It is possible to use categories for users, source hosts, 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, source hosts, and target hosts can be added directly to host-based access control rules. However, services must be flagged first and then added to the access control rules.

14.2. Creating Host-Based Access Control Entries for Services and Service Groups

Any PAM service can be identified as to the host-based access control (HBAC) system in FreeIPA. The service entries used in host-based access control are separate from adding a service to the FreeIPA 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.
14.2.1. Adding HBAC Services

14.2.1.1. Adding HBAC Services in the Web UI
1. Click the Policy tab.
2. Click the Host-Based Access Control subtab, and then select the HBAC Services link.
3. Click the Add link at the top of the list of services.
4. Enter the service name and a description.
5. Click the Add button to save the new service.

6. If a service group already exists, then add the service to the desired group, as described in Section 14.2.2.1, “Adding Service Groups in the Web UI”.

14.2.1.2. Adding Services in the Command Line
The service is added to the access control system using the `hbacsvc-add` command, specifying the service by the name that PAM uses to evaluate the service.

For example, this adds the `tftp` service:

```
# ipa hbacsvc-add --desc="TFTP service" tftp
-------------------------
Added HBAC service "tftp"
-------------------------
```

If a service group already exists, then the service can be added to the group using the `hbacsvgroup-add-member` command, as in Section 14.2.2.2, “Adding Service Groups in the Command Line”.

14.2.2. Adding Service Groups
Once the individual service is added, it can be added to the access control rule. However, if there a large number of services, then it can require frequent updates to the access control rules as services change. FreeIPA also allows groups of services to 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.

14.2.2.1. Adding Service Groups in the Web UI
1. Click the Policy tab.

2. Click the Host-Based Access Control subtab, and then select the HBAC Service Groups link.

3. Click the Add link at the top of the list of service groups.
4. Enter the service group name and a description.

5. Click the Add and Edit button to go immediately to the service configuration page.

6. At the top of the HBAC Services tab, click the Add link.
7. Click the checkbox by the names of the services to add, and click the right arrows button, >>, to move the command to the selection box.
8. Click the **Add** button to save the group membership.

### 14.2.2.2. Adding Service Groups in the Command Line

First create the service group entry, then create the service, and then add that service to the service group as a member. For example:

```
$ ipa hbacsvcgroup-add --desc="login services" login
                        ---------------
Added HBAC service group "login"
                        ---------------
    Service group name: login
    Description: login services

$ ipa hbacsvc-add --desc="SSHD service" sshd
                        ---------------
Added HBAC service "sshd"
                        ---------------

$ ipa hbacsvcgroup-add-member --hbacsvcs=sshd login
                        ---------------
Service group name: login
    Description: login services
```

Chapter 14. Policy: Configuring Host-Based Access Control

14.3. Defining Host-Based Access Control Rules

Access controls, at a high level, define who has access to what. The who can be either a user or a host (the source host), and the what can be either a host (target host), service, or service group, or a combination of the three.

14.3.1. Setting Host-Based Access Control Rules in the Web UI

1. Click the Policy tab.

2. Click the Host-Based Access Control subtab, and then select the HBAC Rules link.

3. Click the Add link at the top of the list of host-based access control rules.

4. Enter the name for the rule.
5. Click the Add and Edit button to go immediately to set the configuration for the rule.

There are a number of configuration areas for the rule. The four basic elements are who the rule applies to, what hosts users can use to access resources (the source), what hosts allow access (the target), and, optionally, what services can be accessed.

6. In the WHO area, select the users or user groups to which the access control rule is applied.

To apply the rule to all FreeIPA users, select the Anyone radio button.

To apply the rule to a specific set of users or user groups:

a. Select the Specified Users and Groups radio button.

b. Click the + Add link at the right of the users list.

c. Click the checkbox by the users to add to the rule, and click the right arrows button, >>, to move the users to the selection box.
d. Click **Add**.

7. In the **Accessing** area, select the target hosts which can be accessed through this access control rule.

   To apply the rule to all FreeIPA hosts, select the **Any Host** radio button.

   To apply the rule to a specific set of hosts or host groups:

   a. Select the **Specified Hosts and Groups** radio button.

   b. Click the **+ Add** link at the right of the hosts list.
c. Click the checkbox by the hosts to include with the rule, and click the right arrows button, >>, to move the hosts to the selection box.

d. Click Add.
Chapter 14. Policy: Configuring Host-Based Access Control

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 FreeIPA hosts, select the **Any Service** radio button.

   To apply the rule to a specific set of hosts or host groups:

   a. Select the **Specified Services and Groups** radio button.

   b. Click the **+ Add** link at the right of the commands list.

   c. Click the checkbox by the services or groups to include with the rule, and click the right arrows button, >>, to move the services to the selection box.
d. Click **Add**.

9. In the **From** area, select the source hosts, from which users can access the target hosts.

   To apply the rule to all FreeIPA hosts, select the **Any Host** radio button.

   To apply the rule to a specific set of hosts or host groups:

   a. Select the **Specified Hosts and Groups** radio button.

   b. Click the **+ Add** link at the right of the hosts list.
c. Click the checkbox by the hosts to include with the rule, and click the right arrows button, >>, to move the hosts to the selection box.

d. Click Add.
14.3.2. Setting Host-Based Access Control Rules in the Command Line

Access control rules are created using the `hbacrule-*` commands (listed in Table 14.1, “Host-Based Access Control Command and Options”). The first step is to create a container entry; from there, users, hosts, and services can be added to the access control entry.

The basic outline of all the access control commands is:

```
$ ipa hbacrule-add* options ruleName
```

**TIP**

To set every user, every host as a source host, or every host as a target, use the category options, such as `--usercat=all`.

**Example 14.1. Granting All Access to One Host**

One simple rule is to grant every user and every source host access to a single server. The first command creates the entry and uses the category options to apply every user and every host as a source host.

```
$ ipa hbacrule-add --usercat=all --srchostcat=all allGroup
```

```
Added HBAC rule "allGroup"
```

```
Rule name: allGroup
User category: all
Source host category: all
Enabled: TRUE
```

The second rule adds the target host, using the `hbacrule-add-host` command:

```
$ ipa hbacrule-add-host --hosts=server.example.com allGroup
```

```
Rule name: allGroup
User category: all
Source host category: all
Enabled: TRUE
```

```
Successful hosts/hostgroups:
  member host: server.example.com
```

```
Number of members added 1
```

**Example 14.2. 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 --srchostcat=all sshd-jsmith
```
Since the access control rule applies to a specific user, the user is added to the rule using the `hbacrule-add-user` command:

```
$ ipa hbacrule-add-user --users=jsmith sshd-jsmith
```

Then, the service is added to the access control rule. (The service should have already been added to the access control system using the `hbacsvc-add` command.) This is the service that the user can use to connect to the machine.

```
$ ipa hbacrule-add-service --hbacsvcs=sshd sshd-jsmith
```

**Example 14.3. Adding a Service Group to the Rule**

While a single service can be added to a rule, it is also possible to add an entire service group. Like a single service, this uses the `hbacrule-add-service` command, only with the `--hbacsvcgroups` option that specifies the group name.

```
$ ipa hbacrule-add-service --hbacsvcgroups=login loginRule
```

### Table 14.1. Host-Based Access Control Command and Options

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Arguments</th>
<th>Source or Target Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbacrule-add</td>
<td>Adds a new host-based access control rule.</td>
<td>• --usercat=all, which applies the rule to every user</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --hostcat=all, which sets every host as an allowed target server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --srhoctcat=all, which sets every host as an allowed source server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• --servicecat=all, which sets every configured service as an allowed target service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <code>ruleName</code>, which is the required unique identifier for the new rule</td>
<td></td>
</tr>
<tr>
<td>hbacrule-add-host</td>
<td>Adds a target host to the access control rule. A target host can be accessed by other servers and users in the domain.</td>
<td>• --hosts, which adds an individual server or command-separated list of servers as an allowed target server</td>
<td>Target</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Arguments</td>
<td>Source or Target Entry</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hostgroups, which adds a host group to the rule and every host within the host group is an allowed target server</td>
<td>Source</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ruleName, which is the rule to which to add the target server</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hbacsvcs, which adds an individual service type or a comma-separated list of service type as an allowed target service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hbacsvcgroups, which adds a service group to the rule and every service within the service group is an allowed target service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ruleName, which is the rule to which to add the target service</td>
<td></td>
</tr>
<tr>
<td>hbacrule-add-service</td>
<td>Adds a service type to the rule.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hosts, which adds an individual server or command-separated list of servers as an allowed source server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hostgroups, which adds a host group to the rule and every host within the host group is an allowed source server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ruleName, which is the rule to which to add the source server</td>
<td>Source</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hbacrule-add-sourcehost</td>
<td>Adds a server to the list of servers that can be used to access domain services and target servers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--users, which adds an individual user or command-separated list of servers as an allowed source server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--hostgroups, which adds a host group to the rule and every host within the host group is an allowed source server</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ruleName, which is the rule to which to add the source server</td>
<td></td>
</tr>
<tr>
<td>hbacrule-add-user</td>
<td>Adds a user to the access control rule.</td>
<td>--users, which adds an individual user or command-separated list of servers as an allowed source server</td>
<td>Source</td>
</tr>
</tbody>
</table>
Chapter 14. Policy: Configuring Host-Based Access Control

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Arguments</th>
<th>Source or Target Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The user is then able to access any allowed target host or service within</td>
<td>command-separated list of users to the rule</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the domain, from any configured source host.</td>
<td>--groups, which adds a user group to the rule and, thus, every user</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>within the group</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ruleName, which is the rule to which to add the user</td>
<td></td>
</tr>
<tr>
<td>hbacrule-disable</td>
<td>Disables or enables a host-based access control rule. Rules can be</td>
<td>ruleName, which is the rule to disable or enable</td>
<td></td>
</tr>
<tr>
<td>hbacrule-enable</td>
<td>disabled if their behavior needs to be evaluated (for troubleshooting or</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>to test a new rule).</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14.4. 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.

14.4.1. The Limits of Host-Based Access Control Configuration

When a request is made to one of the domain services, the identifier of the source host is sent to a PAM interface for the login application (like OpenSSH, telnet, or rlogin) which is used to access SSSD on the target host machine. This identifier can be the hostname, the IP address, or even the alias of the source host.

This is essentially a long chain, from the source host to the target host to the service. The access control mechanism uses the identifier that is sent to the SSSD service on the target machine to verify that the source host has proper authorization granted in an access control rule.

However, if the identifier sent to SSSD cannot be traced back to the source host, then authorization fails. This can happen if any part of the host information does not match the access control rule configuration:

- The source host uses an alias which is not listed in the rule.
- The domain service on the target host sends an IP address instead of the hostname.
- DNS is misconfigured and does not return the proper information for the DNS lookup or reverse lookup.
- The client sends a malformed string to SSSD.
The name that is sent by the client must match the name in the rule, otherwise host-based access control fails.

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.

FreeIPA 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.

**14.4.2. Test Scenarios for Host-Based Access Control (CLI-Based)**

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`).
- From the source host X (`--srchost`).
- Using the login client Y (`--service`).
- To target host Z (`--host`).
- The rule to test (`--rules`); if this is not used, then all enabled rules are tested.

*Optional* The `hbactest` returns detailed information about which rules were matched, not matched, or invalid. This detailed rule output can be disabled using `--nodetail`, so the test simply runs and returns whether access was granted.

### NOTE

The `hbactest` script does not actually connect to the target host. Instead, it uses the rules within the FreeIPA database to simulate how those rules would be applied in a specific situation as if an SSSD client were connecting to the FreeIPA 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 14.4. Testing All Active Rules**

The most basic command checks all active rules. It requires a specific connection scenario, so the user, login service, source host, and target host have to be given, and the testing tool checks the connection.

```bash
$ ipa hbactest --user=jsmith --srchost=source.example.com --host=target.example.com --service=ssh
```
Example 14.5. Testing a Specific Rule

It is possible to check a specific rule (or several rules).

```bash
$ ipa hbactest --user=jsmith --srchost=source.example.com --host=target.example.com --service=ssh --rules=myrule
```

```
Access granted: True
---------------------
notmatched: myrule
```

Example 14.6. 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.

```bash
$ ipa hbactest --user=jsmith --srchost=source.example.com --host=target.example.com --service=ssh --rules=myrule --enabled
```

```
Access granted: True
---------------------
matched: my-second-rule
notmatched: my-third-rule
matched: myrule
matched: allow_all
```

It is possible to run a similar comparison against disabled rules by using the `--disable` option. With the `--rules` option, the specified rule plus all of the disabled rules are checked. With the `--disabled` option, all disabled rules are checked.

14.4.3. Testing Host-Based Access Control Rules in the UI

As Section 14.4.1, “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.

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).
- From the source host X (From); this can be an FreeIPA client or an external host.
• The rule to test; if this is not used, then all enabled rules are tested (Rules).

The test environment is defined on the HBAC TEST page in the Host Based Access Control tab under Policy. A series of tabs is set up for each configuration step.

![Figure 14.2. The From Tab to Set up an HBAC Test](image)

Once the environment is defined, then the test is run simply by clicking a button on the Run Test page. The results show clearly whether access was granted or denied to the users, and then runs through the rules which matched the given parameters.
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.
Policy: Defining SELinux User Maps

Security-enhanced Linux (SELinux) sets rules over what system users can access processes, files, directories, and system settings. Both the system administrator and applications themselves can define security contexts that restrict or allow user access and even access from other applications.

As part of defining centralized security policies in the FreeIPA domain, FreeIPA provides a way to map FreeIPA users to SELinux users and automatically grant or restrict access to clients and services within the FreeIPA domain, per host, based on the defined SELinux policies.

15.1. About FreeIPA, SELinux, and Mapping Users

Security-enhanced Linux defines kernel-level, mandatory access controls for how users, processes, and applications can interact with other resources on a system. These rules for interactions, called contexts, look at the data and behavior characteristics of different objects on the system and then set rules, called policies, which create contexts based on the security implications of each specific object. This is in contrast to higher-level discretionary access controls which are concerned primarily with file ownership and user identity, without accounting for data criticality or application behavior.

System users are associated with an SELinux role. The role is assigned both a multi-layer security context (MLS) a multi-category security context (MCS). The MLS/MCS contexts confine users to what processes, files, and operations they can access on the system.
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 FreeIPA can cleanly integrate an identity domain with local SELinux services. FreeIPA can map FreeIPA users to configured SELinux roles per host. Mapping SELinux and FreeIPA users improves user administration:

- Remote users can be granted appropriate SELinux user contexts based on their FreeIPA 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.

---

About FreeIPA, SELinux, and Mapping Users

- 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 FreeIPA 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 FreeIPA user, and an FreeIPA 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 FreeIPA user.

This arrangement allows administrators to set different SELinux users for the same FreeIPA 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 FreeIPA 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 FreeIPA server, so there is a central and universal list. These are SELinux users which are configured on every host in the FreeIPA domain. By default, there are five common SELinux users defined:

- guest_u (also used as a default for FreeIPA users)
- xguest_u
- user_u
- staff_u
- unconfined_u

In the FreeIPA 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 FreeIPA user and host configuration is very flexible. Users and hosts can be explicitly and individually assigned to an SELinux user map individually, or user groups or host groups can be explicitly assigned to the map.

An extra layer of security is possible by using host-based access control rules. As long as the host-based access control rule defines a user and a host, it can be used for an SELinux user map. Host-based access control rules (described in Chapter 14, Policy: Configuring Host-Based Access Control) help integrate SELinux user maps with other access controls in FreeIPA 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.

15.2. Configuring SELinux Users in FreeIPA

SELinux user maps, as the name implies, creates an association between an SELinux user and an FreeIPA user. Before that association can be established, the FreeIPA server has to be aware of what SELinux users are configured on the systems it manages.

The available SELinux users are part of the FreeIPA 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 FreeIPA server configuration can also set a default SELinux user (which is part of the larger SELinux map list) to use for otherwise unmapped FreeIPA user entries.

15.2.1. 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 FreeIPA 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 FreeIPA users.
4. Click the **Update** link at the top of the page to save the changes.

**15.2.2. 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 FreeIPA server configuration:

```
[jsmith@server ~]$ ipa config-show
...
SELinux user map order: guest_u:s0$guest_u:s0$user_u:s0-s0:c0.c1023$staff_u:s0-s0:c0.c1023$unconfined_u:s0-s0:c0.c1023
Default SELinux user: guest_u:s0
```

The SELinux user settings can be edited using the **config-mod** command.
Example 15.1. 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:

```bash
[jsmith@server ~]$ ipa config-mod --ipaselinuxusermaporder="guest_u:s0$guest_u:s0$s0:$unconfined_u:s0-s0:c0.c1023$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 15.2. Default SELinux User

FreeIPA users are not required to have a specific SELinux user mapped to their account. However, the local system still checks the FreeIPA entry for an SELinux user to use for the FreeIPA user account. The default SELinux user sets the fallback user to use for unmapped FreeIPA user entries; this is, by default, a relatively confined user such as guest_u.

This default user can be changed with the `--ipaselinuxusermapdefault`. For example:

```bash
[jsmith@server ~]$ ipa config-mod --ipaselinuxusermapdefault="guest_u:s0"
```

15.3. Mapping SELinux Users and FreeIPA Users

An SELinux map associates an SELinux user with a FreeIPA user (or users). However, SELinux settings are local to each host system, so a map not only needs to map the SELinux user with a FreeIPA user but also with a host system.

The rule definition primarily identifies the SELinux user; the SELinux user is the basis of the rule.

The other half of the map is comprised of defined FreeIPA users and defined FreeIPA hosts. (There can be one single user or host or multiple users and hosts or user and host groups in the map.) The users and hosts can be defined either by explicitly listing users and hosts or by referencing a host-based access control rule.
15.3.1. In the Web UI

1. In the top menu, click the **Policy** main tab and the **SELinux User Mappings** subtab.

2. In the list of mappings, click the **Add** button to create a new map.

3. Enter the name for the map and the SELinux user exactly as it appears in the FreeIPA server configuration. SELinux users have the format `SELinux_username:MLS[:MCS]`.

4. Click **Add and Edit** to add the FreeIPA user information.

5. As described in the introduction, an SELinux map has three parts: the SELinux user and an FreeIPA user/host pairing. That FreeIPA user/host pair can be defined in one of two ways: it can
be set for explicit users on explicit hosts, or it can be defined using a host-based access control rule.

To set a host-based access control rule, select the rule from the drop-down menu in the General area of the configuration. Using a host-based access control rule also introduces access controls on what hosts a remote user can use to access a target machine. Only one host-based access control rule can be set.

Alternatively, scroll down the Users and Hosts areas, and click the Add link to assign users, user groups, hosts, or host groups to the SELinux map.
Select the users (or hosts or groups) on the left, click the right arrows button (>>) to move them to the Prospective column, and click the Add button to add them to the rule.
Chapter 15. Policy: Defining SELinux User Maps

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.

15.3.2. In the CLI

An SELinux map rule has three fundamental parts:

- The SELinux user (--selinuxuser)
In the CLI

- 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 15.3. Creating a New SELinux Map

The `--selinuxuser` value must be the SELinux user name exactly as it appears in the FreeIPA server configuration. SELinux users have the format `SELinux_username:MLS[:MCS]`.

Both a user and a host (or appropriate groups) must be specified for the SELinux mapping to be valid. Users, hosts, or groups can be specified in comma-separated lists.

```
[jsmith@server ~]$ ipa selinuxusermap-add --users=jsmith,bjensen,jrockford --hosts=server.example.com,test.example.com --selinuxuser="xguest_u:s0" selinux1
```

### Example 15.4. Creating an SELinux Map with a Host-Based Access Control Rule

The `--hbacrule` value identifies the host-based access control rule to use for mapping. Using a host-based access control rule introduces access controls on what hosts a remote user can use to access a target machine, along with applying SELinux contexts after the remote user as logged into the target machine.

The access control rule must specify both users and hosts appropriately so that the SELinux map can construct the SELinux user, FreeIPA 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 14, Policy: Configuring Host-Based Access Control.

### Example 15.5. Adding a User to an SELinux Map

While all of the users and hosts can be added to a map when it is created, users and hosts can also be added after the rule is created. This is done using a specific command, either `selinuxusermap-add-user` or `selinuxusermap-add-host`.

```
[jsmith@server ~]$ ipa selinuxusermap-add-user --users=jsmith selinux1
```

It is not necessary to use a separate command to add a host-based access control rule after the rule is configured because there can only be one. If the `selinuxusermap-mod` command is used with the `--hbacrule` option, it adds the host-based access control rule or overwrites the previous one.

A specific user or host can be removed from an SELinux map by using either the `selinuxusermap-remove-host` or `selinuxusermap-remove-user` command.
Example 15.6. Removing a User from an SELinux Map
As with adding a user to a mapping, the value identifies the host-based access control rule to use for mapping. The access control rule must specify both users and hosts appropriately so that the SELinux map can construct the SELinux user, FreeIPA user, and host triple.

[jsmith@server ~]$ ipa selinuxusermap-remove-user --users=jsmith selinux1

15.4. Troubleshooting SELinux Login Problems
SELinux maps only work for remote users, not for users with a local account.

When a remote user logs in, authenticating against the FreeIPA server, then the PAM SELinux modules create a file for that user in `/etc/selinux/policy_name/logins/login`.

If that file does not exist, then it means that SSSD is not properly configured to use the FreeIPA server as one of its identity providers. This is required for SELinux mapping to work. Configuring SSSD is covered in the [Red Hat 6 Deployment Guide](http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/Configuring_Domains.html).

If the file exists but the remote user was given the wrong SELinux context, then the `pam_selinux` module may not be properly configured in the PAM stack. This is the module that reads the SELinux information and sets the user context. If the module is missing, then nothing processes the SELinux map and the user is defined a default context on the system.
Policy: Defining Automatic Group Membership for Users and Hosts

Most of the policies and configuration within the FreeIPA domain are based on groups. Settings from sudo rules to automount to access control are defined for groups, and then those settings are applied to group members.

Managing group membership is an important factor in managing users and hosts. Creating automember groups defines rules to add users and hosts to specified groups automatically, as soon as a new entry is added.

16.1. About Automembership

One of the most critical tasks for managing policies, identities, and security is managing group membership in FreeIPA. 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 ipasusers group. Even if custom groups are configured and all policy configuration is in place, users and hosts cannot take advantage of those policies until they are joined to groups. Of course, this can be done manually, but it is both more efficient and more consistent if group membership can be assigned automatically.

This is done with automembership groups.

Automembership is essentially an automatic, global entry filter that organizes entries, at least in part, based on specific criteria. An automember rule, then, is the way that that filter is specified.

For example, there can be a lot of different, repeatable ways that to categorize identities within the IT and organizational environment:

• Adding all hosts or all users to a single global group.

• Adding employees to specific groups based on their employee type, ID number, manager, or physical location.

• Dividing hosts based on their IP address or subnet.

Automembers provide a way to pre-sort those entries. That makes it easier to configure the actual behavior that you want to configure — like granting different sudo rules to different user types or machines on different subnets or have different automount settings for different users.

**NOTE**

Automembership only applies to new users or groups. Changing the configuration on an existing user or group does not affect group membership, either by adding or removing the user/host in the group.

Automembership is a flag or a target set on an existing user group or host group. An automembership rule is created as a policy. This is a sister entry to the actual group entry and it signals that the given group is used for automatic group membership.
Chapter 16. Policy: Defining Automatic Group Membership for Users and Hosts

Once the rule is created — once the group is identified as being a target — then the next step is to define automember conditions. Conditions are regular expression filters that are used to identify group members. Conditions can be inclusive or exclusive, meaning that matching entries can be added or ignored based on those conditions.

There can be multiple conditions in a single rule. A user or host entry can match multiple rules and be added to multiple groups.

Automembership is a way of imposing reliable order on user and host group entries as they are created.

The key to using automember groups effectively is to plan your overall FreeIPA 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 FreeIPA
- 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

16.2. Defining Automembership Rules (Basic Procedure)

16.2.1. From the Web UI

1. Create the user group (Section 5.9.1, “Creating User Groups”) or host group (Section 6.10.1.1, “Creating Host Groups from the Web UI”).

2. Open the Policy tab, and select the Automembers subtab.

3. In the top of the Automembers area, select the type of autogroup to create, either USER GROUP RULES or HOST GROUP RULES.

4. In the drop-down menu, select the group for which to create the automember rule.
5. Click the **Add and Edit** button.

6. In the edit page for the rule, click the **+ Add** by the type of condition to create to identify entries.
7. Select the attribute to use as the basis for the search and then set the regular expression to use to match the attribute value.

Conditions can look for entries either to include in the group or to explicitly exclude from the group. The format of a condition is a Perl-compatible regular expression (PCRE). For more information on PCRE patterns, see the `pcresyntax(3)` man page\(^1\).

**NOTE**

Exclude conditions are evaluated first and take precedence over include conditions.
8. Click **Add and Add Another** to add another condition. A single rule can have multiple include and exclude conditions. When all conditions have been configured, click the **Add** button to save the last condition and close the dialog window.

### 16.2.2. From the CLI

There are two commands used to define an automember policy:

- A command to flag the group as an automember group, **automember-add**
- A command to add regular expression conditions to identify group members, **automember-add-condition**

For example:

1. Create the user group (**Section 5.9.1.2, “With the Command Line”**) or host group (**Section 6.10.1.2, “Creating Host Groups from the Command Line”**).

2. Create the automember policy entry for the group. Use the **--type** to identify whether the target group is a user group (**group**) or a host group (**hostgroup**). This command has the format:

   ```
   ipa automember-add --type=group|hostgroup groupName
   ```

   For example:

   ```
   [jsmith@server ~]$ ipa automember-add --type=group exampleGroup
   ```

3. Create the conditions for the rule. To set multiple patterns, either give a comma-separated list of patterns in the **--inclusive-regex|--exclusive-regex** options or run the command multiple times.

   This command has the format:

   ```
   ipa automember-add-condition --type=group|hostgroup --key=attribute --inclusive-regex=regex | --exclusive-regex=regex groupName
   ```
Chapter 16. Policy: Defining Automatic Group Membership for Users and Hosts

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
```

**TIP**

The regular expression can match any port 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 it must be an exact 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.

### 16.3. Examples of Using Automember Groups

**NOTE**

These examples are shown using the CLI; the same configuration can be performed in the web UI.

**A Note on Creating Default Groups**

One common environment requirement is to have some sort of default group that users or hosts are added to. There are a couple of different ways to approach that.

- All entries can be added to a single, global group regardless of what other groups they are also added to.
Entries can be added to specific automember groups. If the new entry does not match any autogroup, then it is added to a default or fallback group.

These strategies are mutually exclusive. If an entry matches a global group, then it does match an automember group and would, therefore, not be added to the fallback group.

### 16.3.1. 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
```

Every host added after that is automatically added to the **allhosts** group:

```
[jsmith@server ~]$ ipa host-add test.example.com
```

For more information on PCRE patterns, see the `pcresyntax(3)` man page³.

### 16.3.2. 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
```

Chapter 16. Policy: Defining Automatic Group Membership for Users and Hosts

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
```

### 16.3.3. Using Automembership Groups with Windows Users

When a user is created in FreeIPA, 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 FreeIPA, by using an automember group. Every Windows user is added with the `ntUser` object class; that object class can be used as an inclusive filter to identify new Windows users to add to the automember group.

First, define the `ipausers` group as an automember group:

```
[jsmith@server ~]$ ipa automember-add --type=group ipausers
```

Then, use the `ntUser` object class as a condition to add users:

```
[jsmith@server ~]$ ipa automember-add-condition ipausers --key=objectclass --type=group --inclusive-regex=ntUser
```
Chapter 17.

Configuration: Defining Access Control within FreeIPA

Access control is a security system which defines who can access certain resources — from machines to services to entries — and what kinds of operations they are allowed to perform. FreeIPA provides several access control areas to make it very clear what kind of access is being granted and to whom it is granted. As part of this, FreeIPA draws a distinction between access controls to resources within the domain and access control to the FreeIPA configuration itself.

This chapter details the different internal access control mechanisms that are available for users within FreeIPA to the FreeIPA server and other FreeIPA users.

17.1. About Access Controls for FreeIPA Entries

Access control defines the rights or permissions users have been granted to perform operations on other users or objects.

17.1.1. A Brief Look at Access Control Concepts

The FreeIPA access control structure is based on standard LDAP access controls. Access within the FreeIPA server is based on the FreeIPA users (who are stored in the backend Directory Server instance) who are allowed to access other FreeIPA entities (which are also stored as LDAP entries in the Directory Server instance).

An access control rule has three parts:

- **Who can perform the operation.** This is the entity who is being granted permission to do something; this is the actor. In LDAP access control models, this is called the *bind rule* because it defines who the user is (based on their bind information) and can optionally require other limits on the bind attempt, such as restricting attempts to a certain time of day or a certain machine.

- **What can be accessed.** This defines the entry which the actor is allowed to perform operations on. This is the *target* of the access control rule.

- **What type of operation can be performed.** The last part is determining what kinds of actions the user is allowed to perform. The most common operations are add, delete, write, read, and search. In FreeIPA, all users are implicitly granted read and search rights to all entries in the FreeIPA domain, with restrictions only for sensitive attributes like passwords and Kerberos keys. (Anonymous users are restricted from seeing security-related configuration, like *sudo* rules and host-based access control.)

The only rights which can be granted are add, delete, and write — the permissions required to *modify* an entry.
NOTE

FreeIPA does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In FreeIPA, read and search access are implicitly granted to any authenticated user.

Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

When any operation is attempted, the first thing that the FreeIPA client does is send user credentials as part of the bind operation. The backend Directory Server checks those user credentials and then checks the user account to see if the user has permission to perform the requested operation.

17.1.2. Access Control Methods in FreeIPA

To make access control rules simple and clear to implement, FreeIPA divides access control definitions into three categories:

- **Self-service rules**, which define what operations a user can perform on his own personal entry. The access control type only allows write permissions to attributes within the entry; it does not allow add or delete operations for the entry itself.

- **Delegation rules**, which allow a specific user group to perform write (edit) operations on specific attributes for users in another user group. Like self-service rules, this form of access control rule is limited to editing the values of specific attributes; it does not grant the ability to add or remove whole entries or control over unspecified attributes.

- **Role-based access control**, which creates special access control groups which are then granted much broader authority over all types of entities in the FreeIPA 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 FreeIPA. Special roles can be created to manage any type of entry in specific ways, such as hosts, automount configuration, netgroups, DNS settings, and FreeIPA configuration.

17.2. 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 FreeIPA entity) can edit on their personal entries.

Two self-service rules exist by default:

- A rule for editing some general attributes in the personal entry, including given name and surname, phone numbers, and addresses.

- A rule to edit user passwords, including two Samba passwords, the Kerberos password, and the general user password.
17.2.1. 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 the Add link at the top of the list of self-service ACIs.

3. Enter the name of the rule in the pop-up window. Spaces are allowed.
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.

17.2.2. Creating Self-Service Rules from the Command Line

A new self-service rule can be added using the `selfservice-add` command. There are two required options, `--permissions` to set whether the ACI grants write, add, or delete permission and `--attrs` to give the full list of attributes which this ACI grants permission to.

```
$ ipa selfservice-add "Users can manage their own name details" --permissions=write --attrs=givenname,displayname,title,initials
----------------------------------------------
Added selfservice "Users can manage their own name details"
----------------------------------------------
Self-service name: Users can manage their own name details
Permissions: write
Attributes: givenname, displayname, title, initials
```

17.2.3. 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.
Editing Self-Service Rules

With the command line, self-service rules are edited using the `ipa selfservice-mod` command. The `--attrs` option overwrites whatever the previous list of supported attributes was, so always include the complete list of attributes along with any new attributes.

```
$ ipa selfservice-mod "Users can manage their own name details" --attrs=givenname,displayname,title,initials,surname
--------------------------------------------------------------
Modified selfservice "Users can manage their own name details"
--------------------------------------------------------------
Self-service name: Users can manage their own name details
Permissions: write
Attributes: givenname, displayname, title, initials
```
17.3. 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 FreeIPA user groups instead of roles specifically created for access controls.

17.3.1. Delegating Access to User Groups in the Web UI

1. Open the IPA Server tab in the top menu, and select the Delegations subtab.

2. Click the Add link at the top of the list of delegation ACIs.

3. Name the new delegation ACI.

4. In the User group drop-down menu, select the group whose entries can be edited by members of the delegation group.
5. In the **Member user group** drop-down menu, select the group *who is being granted permissions to* the entries of users in the user group.

6. In the attributes box, select the checkboxes by the attributes to which the member user group is being granted permission.

7. Click the **Add** button to save the new delegation ACI.

### 17.3.2. 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:

```bash
$ ipa delegation-add "basic manager attrs" --attrs=manager,title,employeetype,employeenumber --group=engineering_managers --membergroup=engineering
```

--------------------------------------
Chapter 17. Configuration: Defining Access Control within FreeIPA

```
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.

```
$ ipa delegation-mod "basic manager attrs" --
  attrs=manager,title,employeetype,employeenumber,displayname
-----------------------------------------
Modified delegation "basic manager attrs"
-----------------------------------------
  Delegation name: basic manager attrs
  Permissions: write
  Attributes: manager, title, employeetype, employeenumber, displayname
  Member user group: engineering
  User group: engineering_managers
```

**IMPORTANT**

Include all of the attributes when modifying a delegation rule, including existing ones.

### 17.4. Defining Role-Based Access Controls

Role-based access control grants a very different kind of authority to users compared to self-service and delegation access controls. Role-based access controls are fundamentally administrative, with the potential to add, delete, and significantly modify entries.

There are three parts to role-based access controls:

- The *permission*. The permission defines a specific operation or set of operations (write, add, or delete) and the target entries within the FreeIPA LDAP directory to which those operations apply. Permissions are building blocks; they can be assigned to multiple privileges as needed.

- The *privileges* available to a role. A privilege is essentially a group of permissions. Permissions are not applied directly to a role. Permissions are added to a privilege so that the privilege creates a coherent and complete picture of a set of access control rules. For example, a permission can be created to add, edit, and delete automount locations. Then that permission can be combined with another permission relating to managing FTP services, and they can be used to create a single privilege that relates to managing filesystems.

- The *role*. This is the list of FreeIPA users who are able to perform the actions defined in the privileges.

It is possible to create entirely new permissions, as well as to create new privileges based on existing permissions or new permissions. A list of the default privileges and their associated permissions are in Table 17.1, “Privileges and Permissions in FreeIPA”.

346
NOTE

FreeIPA does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In FreeIPA, read and search access are implicitly granted to any authenticated user.

Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

Table 17.1. Privileges and Permissions in FreeIPA

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Associated Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automount Administrators</td>
<td>Add_Automount_maps</td>
</tr>
<tr>
<td></td>
<td>Remove_Automount_maps</td>
</tr>
<tr>
<td></td>
<td>Add_Automount_keys</td>
</tr>
<tr>
<td></td>
<td>Remove_Automount_keys</td>
</tr>
<tr>
<td>Certificate Administrators</td>
<td>Retrieve_Certificates_from_the_CA</td>
</tr>
<tr>
<td></td>
<td>Request_Certificate</td>
</tr>
<tr>
<td></td>
<td>Request_Certificates_from_a_different_host</td>
</tr>
<tr>
<td></td>
<td>Get_Certificates_status_from_the_CA</td>
</tr>
<tr>
<td></td>
<td>Revoke_Certificate</td>
</tr>
<tr>
<td></td>
<td>Certificate_Remove_Hold</td>
</tr>
<tr>
<td>Delegation Administrator</td>
<td>Add_Roles</td>
</tr>
<tr>
<td></td>
<td>Remove_Roles</td>
</tr>
<tr>
<td></td>
<td>Modify_Roles</td>
</tr>
<tr>
<td></td>
<td>Modify_Role_membership</td>
</tr>
<tr>
<td></td>
<td>Modify_privilege_membership</td>
</tr>
<tr>
<td>DNS Administrators (for users)</td>
<td>add_dns_entries</td>
</tr>
<tr>
<td></td>
<td>remove_dns_entries</td>
</tr>
<tr>
<td></td>
<td>update_dns_entries</td>
</tr>
<tr>
<td>DNS Servers (for machines)</td>
<td>add_dns_entries</td>
</tr>
<tr>
<td></td>
<td>remove_dns_entries</td>
</tr>
<tr>
<td></td>
<td>update_dns_entries</td>
</tr>
<tr>
<td>Group Administrators</td>
<td>Add_Groups</td>
</tr>
<tr>
<td></td>
<td>Remove_Groups</td>
</tr>
<tr>
<td></td>
<td>Modify_Groups</td>
</tr>
<tr>
<td></td>
<td>Modify_Group_membership</td>
</tr>
<tr>
<td>HBAC Administrator</td>
<td>Add_HBAC_rule</td>
</tr>
<tr>
<td></td>
<td>Delete_HBAC_rule</td>
</tr>
<tr>
<td></td>
<td>Modify_HBAC_rule</td>
</tr>
<tr>
<td></td>
<td>Manage_HBAC_rule_membership</td>
</tr>
<tr>
<td></td>
<td>Add_HBAC_services</td>
</tr>
<tr>
<td></td>
<td>Delete_HBAC_services</td>
</tr>
<tr>
<td></td>
<td>Add_HBAC_service_groups</td>
</tr>
<tr>
<td>Privilege</td>
<td>Associated Permissions</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Delete_HBAC_service_groups</td>
<td>Manage_HBAC_service_group_membership</td>
</tr>
<tr>
<td>Host Administrators</td>
<td>Add_Hosts</td>
</tr>
<tr>
<td></td>
<td>Remove_Hosts</td>
</tr>
<tr>
<td></td>
<td>Modify_Hosts</td>
</tr>
<tr>
<td></td>
<td>Manage_host_keytab</td>
</tr>
<tr>
<td></td>
<td>Enroll_a_host</td>
</tr>
<tr>
<td></td>
<td>Add_krbPrincipalName_to_a_host</td>
</tr>
<tr>
<td>Host Enrollment</td>
<td>Manage_host_keytab</td>
</tr>
<tr>
<td></td>
<td>Enroll_a_host</td>
</tr>
<tr>
<td></td>
<td>Add_krbPrincipalName_to_a_host</td>
</tr>
<tr>
<td>Host Group Administrators</td>
<td>Add_Hostgroups</td>
</tr>
<tr>
<td></td>
<td>Remove_Hostgroups</td>
</tr>
<tr>
<td></td>
<td>Modify_Hostgroups</td>
</tr>
<tr>
<td></td>
<td>Modify_Hostgroup_membership</td>
</tr>
<tr>
<td>Modify Users and Reset Passwords</td>
<td>Modify_Users</td>
</tr>
<tr>
<td>Netgroups Administrators</td>
<td>Add_netgroups</td>
</tr>
<tr>
<td></td>
<td>Remove_netgroups</td>
</tr>
<tr>
<td></td>
<td>Modify_netgroups</td>
</tr>
<tr>
<td></td>
<td>Modify_netgroup_membership</td>
</tr>
<tr>
<td>Password Policy Administrator</td>
<td>Add_Group_Password_Policy_costemplate</td>
</tr>
<tr>
<td></td>
<td>Delete_Group_Password_Policy_costemplate</td>
</tr>
<tr>
<td></td>
<td>Modify_Group_Password_Policy_costemplate</td>
</tr>
<tr>
<td></td>
<td>Add_Group_Password_Policy</td>
</tr>
<tr>
<td></td>
<td>Delete_Group_Password_Policy</td>
</tr>
<tr>
<td></td>
<td>Modify_Group_Password_Policy</td>
</tr>
<tr>
<td>Replication Administrators</td>
<td>Add_Replication_Agreements</td>
</tr>
<tr>
<td></td>
<td>Remove_Replication_Agreements</td>
</tr>
<tr>
<td></td>
<td>Modify_Replication_Agreements</td>
</tr>
<tr>
<td>Service Administrators</td>
<td>Add_Services</td>
</tr>
<tr>
<td></td>
<td>Remove_Services</td>
</tr>
<tr>
<td></td>
<td>Modify_Services</td>
</tr>
<tr>
<td></td>
<td>Manage_service_keytab</td>
</tr>
<tr>
<td>Sudo Administrator</td>
<td>Add_Sudo_rule</td>
</tr>
<tr>
<td></td>
<td>Delete_Sudo_rule</td>
</tr>
<tr>
<td></td>
<td>Modify_Sudo_rule</td>
</tr>
<tr>
<td></td>
<td>Add_Sudo_command</td>
</tr>
<tr>
<td></td>
<td>Delete_Sudo_command</td>
</tr>
<tr>
<td></td>
<td>Modify_Sudo_command</td>
</tr>
<tr>
<td></td>
<td>Add_Sudo_command_group</td>
</tr>
<tr>
<td></td>
<td>Delete_Sudo_command_group</td>
</tr>
<tr>
<td></td>
<td>Manage_Sudo_command_group_membership</td>
</tr>
<tr>
<td>User Administrators</td>
<td>Change_a_user_password</td>
</tr>
<tr>
<td></td>
<td>Add_user_to_default_group</td>
</tr>
<tr>
<td></td>
<td>Unlock_user_accounts</td>
</tr>
<tr>
<td></td>
<td>Remove_Users</td>
</tr>
<tr>
<td></td>
<td>Modify_Users</td>
</tr>
<tr>
<td></td>
<td>Add_Users</td>
</tr>
</tbody>
</table>
Creating Roles

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Associated Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write IPA Configuration</td>
<td>Write_IPA_Configuration</td>
</tr>
</tbody>
</table>

*This permission can only be granted to servers, not to users.*

## 17.4.1. Creating Roles

### 17.4.1.1. Creating Roles in the Web UI

1. Open the IPA Server tab in the top menu, and select the Role Based Access Control subtab.

2. Click the Add link at the top of the list of role-based ACIs.

3. Enter the role name and a description.
4. Click the **Add and Edit** button to save the new role and go to the configuration page.

5. Open the **Privileges** tab in the role configuration page.

6. Click the **Add** link at the top of the list of privileges to add a new privilege.

7. Enter the role name and a description.
17.4.1.2. Creating Roles in the Command Line

1. Add the new role:

   ```
   # ipa role-add --desc="User Administrator" useradmin
   ------------------------
   Added role "useradmin"
   ------------------------
   Role name: useradmin
   Description: User Administrator
   ```

2. Add the required privileges to the role:

   ```
   # ipa role-add-privilege --privileges="User Administrators" useradmin
   Role name: useradmin
   Description: User Administrator
   Privileges: user administrators
   -------------------------------
   Number of privileges added 1
   -------------------------------
   ```

3. Add the required groups to the role. In this case, we are adding only a single group, useradmin, which already exists.

   ```
   # ipa role-add-member --groups=useradmins useradmin
   Role name: useradmin
   Description: User Administrator
   Member groups: useradmins
   Privileges: user administrators
   -------------------------------
   Number of members added 1
   -------------------------------
   ```
17.4.2. Creating New Permissions

**NOTE**

FreeIPA does not provide a way to grant read access explicitly, and this is an important distinction from standard LDAP access control rules. In LDAP, all operations, including read, are implicitly denied and must be explicitly granted. In FreeIPA, read and search access are implicitly granted to any authenticated user.

Because read access is already granted, there is no way through the UI to grant read access. However, there is an option in the CLI tools to grant read access for special cases where there may be a broad deny rule set but read access should be granted to specific attributes. For example, read access is blocked to password attributes, but could be allowed by a special read permission.

17.4.2.1. Creating New Permissions from the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.

2. Select the **Permissions** task link.

3. Click the **Add** link at the top of the list of permissions.
4. Enter the name of the new permission.

5. Select the checkboxes next to the allowed operations for this permission.
6. Select the method to use to identify the target entries from the **Target** drop-down menu. There are four different methods:

- **Type** looks for an entry type like user, host, or service and then provides a list of all possible attributes for that entry type. The attributes which will be accessible through this ACI are selected from the list.

- **Filter** uses an LDAP filter to identify which entries the permission applies to.

- **Subtree** targets every entry beneath the specified subtree entry. All attributes within the matching entries can be modified.

- **Target group** specifies a user group, and all the user entries within that group are available through the ACI. All attributes within the matching entries can be modified.
Creating New Permissions

**NOTE**

For **Filter**, **Subtree**, and **Target group** methods, no attributes are set in the UI. These must be added later using `ipa permission-mod --attrs`. If no attributes are set for the permission then, by default, all attributes are excluded.

7. Fill in the required information to identify the target entries, depending on the selected type.
8. Click the **Add** button to save the permission.
9. For **Filter**, **Subtree**, and **Target group** methods, set the attributes for the ACI to include. This must be done from the command line.

For example:

```
[jsmith@ipaserver ~]$ ipa permission-mod "manage Windows groups" --attrs=description,member
```

### 17.4.2.2. Creating New Permissions from the Command Line

A new permission is added using the `permission-add` command. All permissions require a list of attributes over which permission is granted (`--attr`), a list of allowed actions (`--permissions`), and the target entries for the ACI. There are four methods to identify the target entries:

- **--type** looks for an entry type like user, host, or service and then provides a list of all possible attributes for that entry type.
- **--filter** uses an LDAP filter to identify which entries the permission applies to.
- **--subtree** targets every entry beneath the specified subtree entry.
- **--targetgroup** specifies a user group, and all the user entries within that group are available through the ACI.

**Example 17.1. Adding a Permission with a Filter**

A filter can be any valid LDAP filter.

```
$ ipa permission-add "manage Windows groups" --filter="(!(objectclass=posixgroup))" --permissions=write --attrs=description
```

**NOTE**

The `permission-add` command does not validate the given LDAP filter. Verify that the filter returns the expected results before configuring the permission.
Example 17.2. Adding a Permission for a Subtree

All a subtree filter requires is a DN within the directory. Since FreeIPA uses a simplified, flat directory tree structure, this can be used to target some types of entries, like automount locations, which are containers or parent entries for other configuration.

```
$ ipa permission-add "manage automount locations" --subtree="ldap://ldap.example.com:389/cn=automount,dc=example,dc=com" --permissions=write --attrs=automountmapname,automountkey,automountInformation
```

Example 17.3. Adding a Permission Based on Object Type

There seven object types that can be used to form a permission:

- user
- group
- host
- service
- hostgroup
- netgroup
- dnsrecord

Each type has its own set of allowed attributes, in a comma-separated list.

```
$ ipa permission-add "manage service" --permissions=all --type=service --attrs=krbprincipalkey,krbprincipalname,managedby
```

The attributes (**-attrs**) must exist and be allowed attributes for the given object type, or the permission operation fails with schema syntax errors.

### 17.4.3. Creating New Privileges

#### 17.4.3.1. Creating New Privileges from the Web UI

1. Open the **IPA Server** tab in the top menu, and select the **Role Based Access Control** subtab.
2. Select the **Privileges** task link.
3. Click the **Add** link at the top of the list of privileges.
4. Enter the name and a description of the privilege.

5. Click the Add and Edit button to go to the privilege configuration page to add permissions.
6. Select the Permissions tab.

7. Click the Add link at the top of the list of permissions to add permission to the privilege.

8. Click the checkbox by the names of the permissions to add, and click the right arrows button, >>, to move the permissions to the selection box.
9. Click the Add button.

**17.4.3.2. 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.

   ```bash
   $ ipa privilege-add "managing filesystems" --desc="for filesystems"
   ```

2. Assign the desired permissions. For example:

   ```bash
   $ ipa privilege-add-permission "managing filesystems" --permissions="managing automount","managing ftp services"
   ```
Configuration: Configuring the FreeIPA Server

The FreeIPA servers and backend services are configured with default settings that are applicable in most environments.

There are some configuration areas where the FreeIPA server configuration can be tweaked to improve security or performance in certain situations.

This chapter covers information about the FreeIPA configuration, including files and logs used by the FreeIPA server, and procedures for updating the FreeIPA server configuration itself.

18.1. FreeIPA Files and Logs

FreeIPA 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. FreeIPA directly manages each of these elements through their own configuration files and tools.

This section covers the directories, files, and logs used specifically by FreeIPA. For more information about the configuration files or logs for a specific server used within FreeIPA, see the product documentation.

18.1.1. A Reference of FreeIPA Server Configuration Files and Directories

<table>
<thead>
<tr>
<th>Directory or File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/ipa</td>
<td>The main FreeIPA configuration directory.</td>
</tr>
<tr>
<td>/etc/ipa/default.conf</td>
<td>The primary configuration file for FreeIPA.</td>
</tr>
<tr>
<td>/etc/ipa/ca.crt</td>
<td>The CA certificate issued by the FreeIPA server’s CA.</td>
</tr>
<tr>
<td>~/.ipa/</td>
<td>A user-specific FreeIPA directory that is created on the local system in the system user’s home directory the first time the user runs a FreeIPA command.</td>
</tr>
<tr>
<td>~/.ipa/log/cli.log</td>
<td>The log file for all XML-RPC calls and responses by the FreeIPA command-line tools. This is created in the home directory for the system user who runs the tools, who may have a different name than the FreeIPA user.</td>
</tr>
<tr>
<td>/var/log/ipaclient-install.log</td>
<td>The installation log for the client service.</td>
</tr>
<tr>
<td>/var/log/ipaserver-install.log</td>
<td>The installation log for the FreeIPA server.</td>
</tr>
<tr>
<td>System Services</td>
<td></td>
</tr>
<tr>
<td>/etc/rc.d/init.d/ipa</td>
<td>The FreeIPA server init script.</td>
</tr>
</tbody>
</table>
### Directory or File

<table>
<thead>
<tr>
<th>Directory or File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/rc.d/init.d/ipa_kpasswd</td>
<td>The init script for the FreeIPA control daemon for Kerberos passwords.</td>
</tr>
<tr>
<td>/var/run/ipa_kpasswd.pid</td>
<td>The PID file for the Kerberos password daemon used by the FreeIPA service.</td>
</tr>
<tr>
<td></td>
<td><strong>Web UI</strong></td>
</tr>
<tr>
<td>/etc/ipa/html</td>
<td>A symlink directory in the main configuration directory for the HTML files used by the FreeIPA web UI.</td>
</tr>
<tr>
<td>/etc/httpd/conf.d/ipa.conf</td>
<td>The configuration files used by the Apache host for the web UI application.</td>
</tr>
<tr>
<td>/etc/httpd/conf.d/ipa-rewrite.conf</td>
<td>The configuration files used by the Apache host for the web UI application.</td>
</tr>
<tr>
<td>/etc/httpd/conf/ipa.keytab</td>
<td>The keytab file used by the web UI service.</td>
</tr>
<tr>
<td>/usr/share/ipa</td>
<td>The main directory for all of the HTML files, scripts, and stylesheets used by the web UI.</td>
</tr>
<tr>
<td>/usr/share/ipa/ipa-rewrite.conf</td>
<td>The configuration files used by the Apache host for the web UI application.</td>
</tr>
<tr>
<td>/usr/share/ipa/ipa.conf</td>
<td>Contains any updated files, schema, and other elements for FreeIPA.</td>
</tr>
<tr>
<td>/usr/share/ipa/html</td>
<td>Contains the HTML files, JavaScript files, and stylesheets used by the web UI.</td>
</tr>
<tr>
<td>/usr/share/ipa/ipaclient</td>
<td>Contains the JavaScript files used to access Firefox's autoconfiguration feature and set up the Firefox browser to work in the FreeIPA Kerberos realm.</td>
</tr>
<tr>
<td>/usr/share/ipa/migration</td>
<td>Contains HTML pages, stylesheets, and Python scripts used for running the FreeIPA server in migration mode.</td>
</tr>
<tr>
<td>/usr/share/ipa/ui</td>
<td>Contains all of the scripts used by the UI to perform FreeIPA operations.</td>
</tr>
<tr>
<td>/var/log/httpd</td>
<td>The log files for the Apache web server.</td>
</tr>
<tr>
<td></td>
<td><strong>Kerberos</strong></td>
</tr>
<tr>
<td>/etc/krb5.conf</td>
<td>The Kerberos service configuration file.</td>
</tr>
<tr>
<td></td>
<td><strong>SSSD</strong></td>
</tr>
<tr>
<td>/etc/sssd/sssd.api.d/sssd-ipa.conf</td>
<td>The configuration file used to identify the FreeIPA server, FreeIPA Directory Server, and other FreeIPA services used by SSSD.</td>
</tr>
<tr>
<td>/var/log/sssd</td>
<td>The log files for SSSD.</td>
</tr>
<tr>
<td></td>
<td><strong>389 Directory Server</strong></td>
</tr>
<tr>
<td>/var/lib/dirsrv/slapd-REALM_NAME</td>
<td>All of the schema, configuration, and database files associated with the Directory Server instance used by the FreeIPA server.</td>
</tr>
<tr>
<td>/var/log/dirsrv/slapd-REALM_NAME</td>
<td>Log files associated with the Directory Server instance used by the FreeIPA server.</td>
</tr>
<tr>
<td></td>
<td><strong>Dogtag Certificate System</strong></td>
</tr>
<tr>
<td>/etc/pki-ca</td>
<td>The main directory for the FreeIPA CA instance.</td>
</tr>
</tbody>
</table>
### Directory or File

<table>
<thead>
<tr>
<th>Directory or File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/etc/pki-ca/conf/CS.cfg</code></td>
<td>The main configuration file for the FreeIPA CA instance.</td>
</tr>
<tr>
<td><code>/var/lib/dirsrv/slapd-PKI-IPA/</code></td>
<td>All of the schema, configuration, and database files associated with the Directory Server instance used by the FreeIPA CA.</td>
</tr>
<tr>
<td><code>/var/log/dirsrv/slapd-PKI-IPA/</code></td>
<td>Log files associated with the Directory Server instance used by the FreeIPA CA.</td>
</tr>
</tbody>
</table>

#### Cache Files

<table>
<thead>
<tr>
<th>Cache Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/var/cache/ipa</code></td>
<td>Cache files for the FreeIPA server and the FreeIPA Kerberos password daemon.</td>
</tr>
</tbody>
</table>

#### System Backups

<table>
<thead>
<tr>
<th>System Backups</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/var/lib/ipa/sysrestore</code></td>
<td>Contains backups of all of the system files and scripts that were reconfigured when the FreeIPA server was installed. These include the original <code>.conf</code> files for NSS, Kerberos (both <code>krb5.conf</code> and <code>kdc.conf</code>), and NTP.</td>
</tr>
<tr>
<td><code>/var/lib/ipa-client/sysrestore</code></td>
<td>Contains backups of all of the system files and scripts that were reconfigured when the FreeIPA client was installed. Commonly, this is the <code>sssd.conf</code> file for SSSD authentication services.</td>
</tr>
</tbody>
</table>

### 18.1.2. About default.conf and Context Configuration Files

Certain global defaults — like the realm information, the LDAP configuration, and the CA settings — are stored in the `default.conf` file. This configuration file is referenced when the FreeIPA 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 FreeIPA server is started or when the `ipa` command is run, respectively. The FreeIPA 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 FreeIPA directory, `~/.ipa/`. This optional file is merged with `default.conf` and used by the local FreeIPA services.
18.1.3. Checking FreeIPA Server Logs

FreeIPA 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 18.2. FreeIPA Log Files

<table>
<thead>
<tr>
<th>Service</th>
<th>Log File</th>
<th>Description</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeIPA server</td>
<td><code>/var/log/ipaserver-install.log</code></td>
<td>Server installation log</td>
<td></td>
</tr>
<tr>
<td>FreeIPA server</td>
<td><code>~/.ipa/log/cli.log</code></td>
<td>Command-line tool log</td>
<td></td>
</tr>
<tr>
<td>FreeIPA client</td>
<td><code>/var/log/ipaclient-install.log</code></td>
<td>Client installation log</td>
<td></td>
</tr>
<tr>
<td>Apache server</td>
<td><code>/var/log/httpd/access</code></td>
<td>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 FreeIPA-specific messages will be recorded in the error log along with the Apache messages.</td>
<td>Apache log chapter¹</td>
</tr>
<tr>
<td>Dogtag Certificate System</td>
<td><code>/var/log/pki-ca-install.log</code></td>
<td>The installation log for the FreeIPA CA.</td>
<td></td>
</tr>
<tr>
<td>Dogtag Certificate System</td>
<td><code>/var/log/pki-ca/debug</code></td>
<td>These logs mainly relate to certificate operations. In FreeIPA, this is used for service principals, hosts, and other entities which use certificates.</td>
<td>Logging chapter²</td>
</tr>
<tr>
<td>389 Directory Server</td>
<td><code>/var/log/dirsrv/slapd-REALM/access</code></td>
<td>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.</td>
<td>The access log is buffered, so the server only writes to the log every 30 seconds, by default. Monitoring servers and databases³ Log entries explained⁴</td>
</tr>
<tr>
<td>389 Directory Server</td>
<td><code>/var/log/dirsrv/slapd-REALM/access</code></td>
<td>This directory server instance is used by the FreeIPA CA to store</td>
<td>The access log is buffered, so the server only writes to the log</td>
</tr>
</tbody>
</table>
### Checking FreeIPA Server Logs

<table>
<thead>
<tr>
<th>Service</th>
<th>Log File</th>
<th>Description</th>
<th>Additional Information</th>
</tr>
</thead>
</table>
|         | `/var/log/dirsrv/slapd-REALM/audit` | certificate information. Most operational data here will be related to server-replica interactions. | every 30 seconds, by default.  
• *Monitoring servers and databases*  
• *Log entries explained* |
| Kerberos | `/var/log/dirsrv/slapd-REALM/errors` | | |
| 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. |
| DNS | `/var/log/messages` | DNS error messages are included with other system messages. | DNS logging is *not* enabled by default. DNS logging is enabled by running the `querylog` command:  
```
/usr/sbin/rndc querylog
```
This begins writing log messages to the system's `/var/log/messages` file. To turn off logging, run the `querylog` command again. |

### 18.1.3.1. Enabling Server Debug Logging

Debug logging for the FreeIPA server is set in the `server.conf` file.
NOTE

Editing the `defaults.conf` configuration file affects all FreeIPA components, not only the FreeIPA server.

1. Edit or create the `server.conf` file.

```
vim server.conf
```

2. Add the `debug` line and set its value to true.

```
[global]
default=debug=True
```

3. Restart the Apache daemon to load the changes.

```
service httpd restart
```

### 18.1.3.2. Debugging Command-Line Operations

Any command-line operation with the `ipa` command can return debug information by using the `-v` option. For example:

```
$ ipa -v user-show admin
ipa: INFO: trying https://ipaserver.example.com/ipa/xml
First name: John
Last name: Smythe
User login [jsmythe]:
ipa: INFO: Forwarding 'user_add' to server u'https://ipaserver.example.com/ipa/xml'
---------------------
Added user "jsmythe"
---------------------
User login: jsmythe
First name: John
Last name: Smythe
Full name: John Smythe
Display name: John Smythe
Initials: JS
Home directory: /home/jsmythe
GECOS field: John Smythe
Login shell: /bin/sh
Kerberos principal: jsmythe@EXAMPLE.COM
UID: 1966800003
GID: 1966800003
Keytab: False
Password: False
```

Using the option twice, `-vv`, displays the XML-RPC exchange:

```
$ ipa -vv user-add
ipa: INFO: trying https://ipaserver.example.com/ipa/xml
First name: Jane
```
Checking FreeIPA Server Logs

Last name: Russell
User login [j russell]:

ipa: INFO: Forwarding 'user_add' to server u'https://ipaserver.example.com/ipa/xml'
send: u'POST /ipa/xml HTTP/1.0
Host: ipaserver.example.com
Accept-Language: en-us
Authorization: negotiate
User-Agent: xmlrpclib.py/1.0.1 (by www.pythonware.com)
Content-Type: text/xml
Content-Length: 1240

send: <?xml version='1.0' encoding='UTF-8'?>
<methodCall>
<methodName>user_add</methodName>
<params>
<param>
<value><array><data>
<value><string>jrussell</string>
</data></array></value>
</param>
<param>
<value><struct>
<member>
<name>all</name>
<value><boolean>0</boolean>
</member>
<member>
<name>displayname</name>
<value><string>Jane Russell</string>
</member>
<member>
<name>cn</name>
<value><string>Jane Russell</string>
</member>
<member>
<name>noprivate</name>
<value><boolean>0</boolean>
</member>
<member>
<name>uidnumber</name>
<value><int>999</int>
</member>
<member>
<name>raw</name>
<value><boolean>0</boolean>
</member>
<member>
<name>version</name>
<value><string>2.11</string>
</member>
<member>
<name>gecos</name>
<value><string>Jane Russell</string>
</member>
<member>
<name>initials</name>
<value><string>JR</string>
</member>
</struct></value>
</param>
</params>
</methodCall>
reply: 'HTTP/1.1 200 OK
Date: Thu, 15 Sep 2011 00:50:39 GMT
Server: Apache/2.2.15 (Red Hat)
WWW-Authenticate: Negotiate YIGZBgkqhkiG9xIBAgICAg+BiTCBhqADAgEFoQMCAQ6iBwMFACAAACjggGHYE1bgZCCAX
Connection: close
Content-Type: text/xml; charset=utf-8

<methodResponse>
<params>
<param>
<value><struct>
<member>
<name>result</name>
<value><struct>
<member>
<name>dn</name>
<value><string>uid=jrussell,cn=users,cn=accounts,dc=example,dc=com</string>
</member>
<member>
<name>has_keytab</name>
<value><boolean>0</boolean>
</member>
<member>
<name>displayname</name>
<value><array><data>
<value><string>Jane Russell</string>
</data></array></value>
</member>
<member>
<name>uid</name>
<value><array><data>
<value><string>jrussell</string>
</data></array></value>
</member>
<member>
<name>objectclass</name>
<value><array><data>
<value><string>top</string>
<value><string>person</string>
<value><string>organizationalperson</string>
<value><string>inetorgperson</string>
<value><string>inetuser</string>
<value><string>posixaccount</string>
<value><string>krbprincipalaux</string>
<value><string>krbticketpolicyaux</string>
<value><string>ipaobject</string>
</data></array></value>
</member>
<member>
<name>loginshell</name>
<value><array><data>
<value><string>/bin/sh</string>
</data></array></value>
</member>
<member>
<name>objectclass</name>
<value><array><data>
<value><string>posixaccount</string>
</data></array></value>
</member>
<member>
<name>inetorgperson</name>
<value><array><data>
<value><string>inetuser</string>
</value></data></array></value>
</member>
<member>
<name>posixaccount</name>
<value><array><data>
<value><string>krbprincipalaux</string>
</value></data></array></value>
</member>
<member>
<name>account</name>
<value><array><data>
<value><string>krbprincipal</string>
</value></data></array></value>
</member>
<member>
<name>loginshell</name>
<value><array><data>
<value><string>bin/sh</string>
</value></data></array></value>
</member>
</struct></value>
</param>
</params>
</methodResponse>

"/xml"
18.2. Disabling Anonymous Binds

Accessing domain resources and running client tools always require Kerberos authentication. However, the backend LDAP directory used by the FreeIPA server allows anonymous binds by default. This potentially opens up all of the domain configuration to unauthorized users, including information about users, machines, groups, services, netgroups, and DNS configuration.

It is possible to disable anonymous binds on the 389 Directory Server instance by using LDAP tools to reset the `nsslapd-allow-anonymous-access` attribute.

1. Change the `nsslapd-allow-anonymous-access` attribute to `rootdse`.

```
ldapmodify -x -D "cn=Directory Manager" -w secret -h server.example.com -p 389
```
Configuring Alternate Certificate Authorities

Enter LDAP Password:

dn: cn=config
changetype: modify
replace: nsslapd-allow-anonymous-access
nsslapd-allow-anonymous-access: rootdse

**IMPORTANT**

Anonymous access can be completely allowed (on) or completely blocked (off). However, completely blocking anonymous access also blocks external clients from checking the server configuration. LDAP and web clients are not necessarily domain clients, so they connect anonymously to read the root DSE file to get connection information.

The `rootdse` allows access to the root DSE and server configuration **without** any access to the directory data.

2. Restart the 389 Directory Server instance to load the new setting.

```
    service dirsrv restart
```

18.3. Configuring Alternate Certificate Authorities

FreeIPA 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 FreeIPA server certificates.

**TIP**

Save an ASCII copy of the CA certificate as `/usr/share/ipa/html/ca.crt`. This allows users to download the correct certificate when they configure their browsers.

1. Use the `ipa-server-certinstall` command to install the certificate.

```
    # /usr/sbin/ipa-server-certinstall -d /path/to/pkcs12.p12
```

2. To keep using browser autoconfiguration in Firefox, regenerate the `/usr/share/ipa/html/configure.jar` file.
   a. Create a directory, and then create the new security databases in that directory.

```
    # mkdir /tmp/signdb
    # certutil -N -d /tmp/signdb
```

   b. Import the PKCS #12 file for the signing certificate into that directory.
# pk12util -i /path/to/pkcs12.p12 -d /tmp/signdb

c. Make a temporary signing directory, and copy the FreeIPA 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_certNickname -Z /usr/share/ipa/html/ configure.jar -e .html /tmp/sign

### 18.4. Configuring CRLs and OCSP Responders

A certificate is created with a validity period, meaning it has a point where it expires and is no longer valid. The expiration date is contained in the certificate itself, so a client always checks the validity period in the certificate to see if the certificate is still valid.

However, a certificate can also be revoked before its validity period is up, but this information is not contained in the certificate. A CA publishes a certificate revocation list (CRL), which contains a complete list of every certificate that was issued by that CA and subsequently revoked. A client can check the CRL to see if a certificate within its validity period has been revoked and is, therefore, invalid.

Validity checks are performed using the online certificate status protocol (OCSP), which sends a request to an OCSP responder. Each CA integrated with the FreeIPA server uses an internal OCSP responder, and any client which runs a validity check can check the FreeIPA CA's internal OCSP responder.

Every certificate issued by the FreeIPA CA puts its OCSP responder service URL in the certificate. For example:

```plaintext
http://ipaserver.example.com:9180/ca/ocsp
```

**NOTE**

For the FreeIPA OCSP responder to be available, port 9180 needs to be open in the firewall.

### 18.4.1. Using an OCSP Responder with SELinux

Clients can use the FreeIPA 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 FreeIPA 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 FreeIPA 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
  ```

### 18.4.2. Changing the CRL Update Interval

The CRL file is automatically generated by the Dogtag Certificate System CA every four hours. This interval can be changed by editing the Dogtag Certificate System configuration.

1. Stop the CA server.

   ```
   service pki-ca stop
   ```

2. Open the `CS.cfg` file.

   ```
   vim /etc/pki-ca/CS.cfg
   ```

3. Change the `ca.crl.MasterCRL.autoUpdateInterval` to the new interval setting.

4. Restart the CA server.

   ```
   service pki-ca start
   ```

### 18.4.3. Changing the OCSP Responder Location

Each FreeIPA server generates its own CRL. Likewise, each FreeIPA server uses its own OCSP responder, with its own OCSP responder URL in the certificates it issues.

A DNS CNAME can be used by FreeIPA clients, and then from there be redirected to the appropriate FreeIPA server OCSP responder.

1. Open the certificate profile.

   ```
   vim /var/lib/pki-ca/profiles/ca/caIPAserviceCert.cfg
   ```

2. Change the `policyset.serverCertSet.9.default.params.crlDistPointsPointName_0` parameter to the DNS CNAME hostname.

3. Restart the CA server.

   ```
   service pki-ca restart
   ```
That change must be made on every FreeIPA server, with the `crlDistPointsPointName_0` parameter set to the same hostname.

### 18.5. Setting a FreeIPA Server as an Apache Virtual Host

If a standard Apache instance is already running on port 80, then the FreeIPA server can be configured to run on a secondary port, such as port 8089.

**NOTE**

In this configuration, FreeIPA uses standard HTTP, not a secure connection using SSL.

**NOTE**

Do not use port 8080. This port is used by the FreeIPA web UI.

1. Log in as the root user.

2. Edit the `/etc/httpd/conf.d/ipa.conf` file so that it is formatted as an Apache virtual host. Add three lines to the beginning of the file to set the new port number for the FreeIPA web service and to create the root `<VirtualHost>` element.

```plaintext
Listen 8089
NameVirtualHost *:8089
<VirtualHost *:8089>
```

3. Close the `<VirtualHost>` element by adding this line to the end of the file:

```plaintext
</VirtualHost>
```

4. Comment out the rewrite rules from the `/etc/httpd/conf.d/ipa.conf` file:

```plaintext
---
# Redirect to the fully-qualified hostname. Not redirecting to secure # port so configuration files can be retrieved without requiring SSL.
RewriteCond %{HTTP_HOST} !^host.foo.com$ [NC]
RewriteRule ^/(.*) http://host.foo.com/$1 [L,R=301]
# Redirect to the secure port if not displaying an error or retrieving # configuration.
RewriteCond %{SERVER_PORT} !^443$
RewriteCond %{REQUEST_URI} !^/(errors|config|favicon.ico)
RewriteRule ^/(.*) https://host.foo.com/$1 [L,R=301,NC]
---
```

5. Restart the `httpd` service.

```plaintext
# service httpd reload
```
18.6. Setting DNS Entries for Multi-Homed Servers

Some server machines may support multiple network interface cards (NICs). Multi-homed machines typically have multiple IPs, all assigned to the same hostname. This works fine in FreeIPA most of the time because it listens on all available interfaces, except localhost. For a server to be available through any NIC, edit the DNS zone file and add entries for each IP address. For example:

```plaintext
ipaserver IN A 192.168.1.100
ipaserver IN A 192.168.1.101
ipaserver IN A 192.168.1.102
```

18.7. Managing Replication Agreements Between FreeIPA Servers

Information is shared between the FreeIPA servers and replicas using *multi-master replication*. What this means is that servers and replicas all receive updates and, therefore, are data masters. The domain information is copied between the servers and replicas using *replication*.

As replicas are added to the domain, mutual replication agreements are automatically created between the replica and the server it is based on. Additional replication agreements can be created between other replicas and servers or the configuration of the replication agreement can be changed using the `ipa-replica-manage` command.

When a replica is created, the replica install script creates two replication agreements: one going from the master server to the replica and one going from the replica to the master server.
As more replicas and servers are added to the domain, there can be replicas and servers that have replication agreements to other servers and replicas but not between each other. For example, the first FreeIPA server is Server A. Then, the admin creates Replica B, and the install script creates a Server A => Replica B replication agreement and a Replica B => Server A replication agreement. Next, the admin creates Replica C based on Server A. The install script creates a Server A => Replica C replication agreement and a Replica C => Server A replication agreement. Replica B and Replica C both have replication agreements with Server A — but they do not have agreements with each other. For data availability, consistency, failover tolerance, and performance, it can be beneficial to create a pair of replication agreements between Replica B and Replica C, even though their data will eventually be replicated over to each other through replication with Server A.

### 18.7.1. Listing Replication Agreements

The `ipa-replica-manage` command can list all of the servers and replicas in the replication topology, using the `list` command:

```bash
# ipa-replica-manage list
srv1.example.com
srv2.example.com
srv3.example.com
srv4.example.com
```

After getting the server/replica list, then it is possible to list the replication agreements for the server. These are the other servers/replicas to which the specified server sends updates.
Creating and Removing Replication Agreements

Replication agreements are created by connecting one server to another server.

```
# ipa-replica-manage list srv1.example.com
srv2.example.com
srv3.example.com
```

18.7.2. Creating and Removing Replication Agreements

Replication agreements are created by connecting one server to another server.

```
ipa-replica-manage server1 server2
```

If only one server is given, the replication agreements are created between the local host and the specified server.

For example:

```
# ipa-replica-manage connect srv2.example.com srv4.example.com
```

Replication occurs over standard LDAP; to enable SSL, then include the CA certificate for the local host (or the specified server). The CA certificate is then installed in the remote server’s certificate database to enable TLS/SSL connections. For example:

```
# ipa-replica-manage connect --cacert=/etc/ipa/ca.crt srv2.example.com srv4.example.com
```

To remove a replication agreement between specific servers/replicas, use the `disconnect` command:

```
# ipa-replica-manage disconnect srv2.example.com srv4.example.com
```

Using the `disconnect` command removes that one replication agreement but leaves both the server/replica instances in the overall replication topology. To remove a server entirely from the FreeIPA replication topology, with all its data, (and, functionally, removing it from the FreeIPA domain as a server), use the `del` server:

```
# ipa-replica-manage del srv2.example.com
```

18.7.3. Forcing Replication

Replication between servers and replicas occurs on a schedule. Although replication is frequent, there can be times when it is necessary to initiate the replication operation manually. For example, if a server is being taken offline for maintenance, it is necessary to flush all of the queued replication changes out of its changelog before taking it down.

To initiate a replication update manually, use the `force-sync` command. The server which receives the update is the local server; the server which sends the updates is specified in the `--from` option.

```
# ipa-replica-manage force-sync --from srv1.example.com
```

18.7.4. Reinitializing FreeIPA Servers

When a replica is first created, the database of the master server is copied, completely, over to the replica database. This process is called `initialization`. If a server/replica is offline for a long period of
time or there is some kind of corruption in its database, then the server can be re-initialized, with a fresh and updated set of data.

This is done using the **re-initialize** command. The target server being initialized is the local host. The server or replica from which to pull the data to initialize the local database is specified in the **--from** option:

```
# ipa-replica-manage re-initialize --from srv1.example.com
```

### 18.7.5. Resolving Replication Conflicts

Changes — both for FreeIPA domain data and for certificate and key data — are replicated between FreeIPA servers and replicas (and, in similar paths, between FreeIPA and Active Directory servers).

Even though replication operations are run continuously, there is a chance that changes can be made on one FreeIPA server at the same time different changes are made to the same entry on a different FreeIPA server. When replication begins to process those entries, the changes collide — this is a replication conflict.

Every single directory modify operation is assigned a server-specific **change state number** (CSN) to track how those modifications are propagated during replication. The CSN also contains a modify timestamp. When there is a replication conflict, the timestamp is checked and the last change wins.

Simply accepting the most recent change is effective for resolving conflicts with attribute values. That method is too blunt for some types of operations, however, which affect the directory tree. Some operations, like `modrdn`, DN changes, or adding or removing parent and child entries, require administrator review before the conflict is resolved.

**NOTE**

Replication conflicts are resolved by editing the entries directory in the LDAP database.

When there is a replication conflict, both entries are added to the directory and are assigned a `nsds5ReplConfict` attribute. This makes it easy to search for entries with a conflict:

```
ldapsearch -x -D "cn=directory manager" -w password -b "dc=example,dc=com" "nsds5ReplConfict=*" \* nsds5ReplConfict
```

### 18.7.5.1. Solving Naming Conflicts

When two entries are added to the FreeIPA domain with the same DN, both entries are added to the directory, but they are renamed to use the `nsuniqueid` attribute as a naming attribute. For example:

```
nsuniqueid=0a950601-435311e0-86a2f5bd-3cd26022+uid=jsmith,cn=users,cn=accounts,dc=example,dc=com
```

Those entries can be searched for and displayed in the FreeIPA CLI, but they cannot be edited or deleted until the conflict is resolved and the DN is updated.

To resolve the conflict:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:
2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
ldapmodify -x -D "cn=directory manager" -w secret -h ipaserver.example.com -p 389
dn: cn=TempValue,dc=example,dc=com
  changetype: modify
  delete: uid
  dc: jsmith
  delete: nsds5ReplConflict
```

**NOTE**

The unique identifier attribute *nsuniqueid* cannot be deleted.

3. Rename the entry with the intended attribute-value pair. For example:

```
ldapmodify -x -D "cn=directory manager" -w secret -h ipaserver.example.com -p 389
dn: cn=TempValue,dc=example,dc=com
  changetype: modrdn
  newrdn: uid=jsmith
  deleteoldrdn: 1
```

Setting the value of the *deleteoldrdn* attribute to 1 deletes the temporary attribute-value pair *cn=TempValue*. To keep this attribute, set the value of the *deleteoldrdn* attribute to 0.

### 18.7.5.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a **glue** entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

**Glue entries** are temporary entries that include the object classes **glue** and **extensibleObject**. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the **glue** object class and the **nsds5ReplConflict** attribute.

  In such cases, either modify the glue entry to remove the **glue** object class and the **nsds5ReplConflict** attribute to keep the entry as a normal entry or delete the glue entry and its child entries.
Chapter 18. Configuration: Configuring the FreeIPA Server

- The server creates a minimalistic entry with the `glue` and `extensibleObject` object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

### 18.8. Moving CRL Generation from the Master (Original) Server to Another Replica

The first FreeIPA server installed owns the master CA in the PKI hierarchy. The master CA is the authoritative CA; it has the root CA signing key and generates CRLs which are distributed among the other servers and replicas in the topology. All subsequent replica databases are cloned (or copied) directly from that master database as part of running `ipa-replica-install`.

**NOTE**

The only reason to replace the master server is if the master server is being taken offline. There has to be a root CA which can issue CRLs and ultimately validate certificate checks.

### 18.8.1. About the Master Server, Replicas, and Generating CRLs

As explained in Section 1.3.1, "About FreeIPA Servers and Replicas", all servers and replicas work together to share data. This arrangement is the server topology.

Servers — which are created with `ipa-server-install` — can host DNS or CA services. These are the original services. When a replica is created (with `ipa-replica-install`), it is based on the configuration of an existing server. A replica, likewise, can host DNS and CA services, but this is not required.

After they are created, servers and replicas are equal peers in the server topology. They are all read-write data masters and replicate information to each other through multi-master replication. Servers and replicas which host a CA are also equal peers in the topology. They can all issue certificates and keys to FreeIPA clients, and they all replicate information amongst themselves.

The only difference between a server and a replica is which FreeIPA server issues the CRL.

When the first server is installed, it is configured to issue CRLs. In its CA configuration file (`/var/lib/pki-ca/conf/CS.cfg`), it has CRL generation enabled:

```
cr.issuingPointId.enableCRLCache=true
ca.crl.issuingPointId.enableCRLUpdates=true
ca.certStatusUpdateInterval=600
ca.listenToCloneModifications=false
```

All replicas point to that master CA as the source for CRL information and disable the CRL settings:

```
master.ca.agent.host=host
master.ca.agent.port=port number
cr.certStatusUpdateInterval=0
cr.issuingPointId.enableCRLUpdates=false
```

*Promoting* a replica to a master server changes its configuration and enables it to issue CRLs and function as the root CA.
18.8.2. Promoting a Replica

1. If the replica was originally installed without a CA, then create a CA. This requires the replica file that was used to create the replica. (A new one can be created from the original FreeIPA server, if need be.)

   [root@ipareplica ~]# ipa-ca-install -p DMpassword -w adminpassword /var/lib/ipa/repli-info-ipareplica.example.com.gpg

2. On the replica server, stop the CA server.
   
   service pki-cad stop

3. Open the CA's configuration directory.
   
   cd /var/lib/pki-ca/conf

4. Edit the *CS.cfg* file to configure the replica's CA as a master.
   
   a. Delete each line which begins with the *ca.crl.* prefix.

   b. Copy each line beginning with the *ca.crl.* prefix from the CA *CS.cfg* file on the master server into the replica server's CA *CS.cfg* file.

   c. Enable control of the database maintenance thread; the default value for a master CA is **600**.

      ca.certStatusUpdateInterval=600

   d. Enable monitoring database replication:

      ca.listenToCloneModifications=true

   e. Enable maintenance of the CRL cache:

      ca.crl.IssuingPointId.enableCRLCache=true

   f. Enable CRL generation:

      ca.crl.IssuingPointId.enableCRLUpdates=true

   g. Disable the redirect settings for CRL generation requests:

      master.ca.agent.host=hostname
      master.ca.agent.port=port number

5. Start the CA server.
   
   service pki-cad start
18.9. Removing a Replica

Deleting or demoting a replica removes the FreeIPA replica from the server/replica topology so that it no longer processes FreeIPA requests and it also remove the host machine itself from the FreeIPA domain.

1. On an FreeIPA server, obtain a Kerberos ticket before running FreeIPA tools.

   [root@replica ~]# kinit admin

2. List all of the configured replication agreements for the FreeIPA domain.

   [root@replica ~]# ipa-replica-manage list
   Directory Manager password:
   ipaserver.example.com: master
   ipaserver2.example.com: master
   replica.example.com: master
   replica2.example.com: master

3. Removing the replica from the topology involves deleting all the agreements between the replica and the other servers in the FreeIPA domain and all of the data about the replica in the domain configuration.

   [root@replica ~]# ipa-replica-manage del replica.example.com

4. If the replica was configured with its own CA, then also use the ipa-csreplica-manage command to remove all of the replication agreements between the certificate databases for the replica.

   This is required if the replica itself was configured with a Dogtag Certificate System CA. It is not required if only the master server or other replicas were configured with a CA.

   [root@replica ~]# ipa-csreplica-manage del replica.example.com

5. On the replica, uninstall the replica packages.

   [root@replica ~]# ipa-server-install --uninstall -U

18.10. Troubleshooting

18.10.1. Starting FreeIPA with Expired Certificates

If FreeIPA administrative server certificates expire, then most FreeIPA 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.
There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.

**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`.

   ```bash
   [root@ipaserver ~]# vim /etc/httpd/conf.d/nss.conf
   NSSEnforceValidCerts off
   ```

2. Restart Apache.

   ```bash
   [root@ipaserver ~]# service httpd restart
   ```

3. Change the `nsslapd-validate-cert` attribute in the 389 Directory Server configuration to `warn` instead of `true` to disable validity checks.

   ```bash
   [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
   ```


   ```bash
   [root@ipaserver ~]# service dirsrv restart
   ```

**18.10.2. There are SASL, GSS-API, and Kerberos errors in the 389 Directory Server logs when the replica starts.**

When the replica starts, there can be a series of SASL bind errors recorded in the 389 Directory Server logs stating that the GSS-API connection failed because it could not find a credentials cache:

```
slapd_ldap_sasl_interactive_bind - Error: could not perform interactive bind for id [] mech [GSSAPI]: error -2 (Local error) (SASL(-1): generic failure: GSSAPI Error: Unspecified GSS failure. Minor code may provide more information (Credentials cache file '/tmp/krb5cc_496' not found)) ...
```

The replica is looking for a credentials cache in `/tmp/krb5cc_496` (where 496 is the 389 Directory Server user ID) and cannot find it.
There may also be messages that the server could not obtain Kerberos credentials for the host principal:

```
set_krb5creds - Could not get initial credentials for principal [ldap/ replica1.example.com] in keytab [WRFILE:/etc/dirsrv/ds.keytab]: -1765328324 (Generic error)
```

These errors are both related to how and when the 389 Directory Server instance loads its Kerberos credentials cache.

While 389 Directory Server itself supports multiple different authentication mechanisms, FreeIPA only uses GSS-API for Kerberos connections. The 389 Directory Server instance for FreeIPA keeps its Kerberos credentials cache in memory. When the 389 Directory Server process ends — like when the FreeIPA replica is stopped — the credentials cache is destroyed.

Also, the 389 Directory Server is used as the backend storage for the principal information for the KDC.

When the replica then restarts, the 389 Directory Server instance starts first, since it supplies information for the KDC, and then the KDC server starts. This start order is what causes the GSS-API and Kerberos connection errors.

The 389 Directory Server attempts to open a GSS-API connection, but since there is no credentials cache yet and the KDC is not started, the GSS connection fails. Likewise, any attempt to obtain the host credentials also fails.

These errors are transient. The 389 Directory Server re-attempts the GSS-API connection after the KDC starts and it has a credentials cache. The 389 Directory Server logs then record a `bind resumed` message.

These startup GSS-API connection failures can be ignored as long as that connection is successfully established.
Migrating from an LDAP Directory to FreeIPA

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 FreeIPA instance, without losing user or password data.

FreeIPA has migration tools to help move directory data and only requires minimal updates to clients. However, the migration process assumes a very simple deployment scenario (one LDAP directory namespace to one FreeIPA namespace).

19.1. An Overview of LDAP to FreeIPA Migration

The actual migration part of moving from an LDAP server to FreeIPA — 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 FreeIPA. 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, FreeIPA deployment.

A secondary, but significant, consideration is planning how to migrate passwords. FreeIPA requires Kerberos hashes for every user account in addition to passwords. Some of the considerations and migration paths for passwords are covered in Section 19.1.2, “Planning Password Migration”.

19.1.1. Planning the Client Configuration

FreeIPA 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 FreeIPA domain. Administrators must decide which scenario is best for each individual client.

19.1.1.1. Initial Client Configuration (Pre-Migration)

Before deciding where you want to go with the client configuration in FreeIPA, 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.
Chapter 19. Migrating from an LDAP Directory to FreeIPA

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 a FreeIPA server, particularly in schema support and the structure of the directory tree. (For more background on those differences, see Section 1.1, “FreeIPA v. LDAP: A More Focused Type of Service”.) While those differences may impact data (especially with the directory tree, which affects entry names), they have little impact on the client configuration, so it really has little impact on migrating clients to FreeIPA.

19.1.1.2. Recommended Configuration for Fedora Clients

Fedora 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 FreeIPA and leverage the full authentication and identity features in FreeIPA. 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 Fedora Deployment Guide.

Unlike generic LDAP directory services (using pam_ldap and nss_ldap), SSSD establishes relationships between identity and authentication information by defining domains. A domain in SSSD defines four backend functions: authentication, identity lookups, access, and password changes. The SSSD domain is then configured to use an identity provider to supply the information for any one (or all) of those four functions.

SSSD can use FreeIPA for all of its backend functions. This is the ideal configuration because it provides the full range of FreeIPA 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 FreeIPA.

TIP

During the migration process from an LDAP directory to FreeIPA, SSSD can seamlessly migrate user passwords without additional user interaction.
Planning the Client Configuration

Figure 19.2. Clients and SSSD with a FreeIPA Backend

The **ipa-client-install** script automatically configured SSSD to use FreeIPA for all four of its backend services, so Fedora clients are set up with the recommended configuration by default.

![SSSD + IPA](image)

**NOTE**

This client configuration is only supported for Fedora 15 and later, which supports the latest versions of SSSD and **ipa-client**. Older versions of Fedora can be configured as described in [Section 19.1.1.3, "Alternative Supported Configuration"](#).

### 19.1.1.3. Alternative Supported Configuration

Unix and Linux systems such as Mac, Solaris, HP-UX, AIX, and Scientific Linux support all of the services that FreeIPA manages but do not use SSSD. Likewise, older Fedora versions (15) support SSSD but have an older version, which does not support FreeIPA 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 FreeIPA server as if it were an LDAP directory service for identity lookups (using **nss_ldap**) and to FreeIPA as if it were a regular Kerberos KDC (using **pam_krb5**).

![LDAP + Kerberos](image)

**Figure 19.3. Clients and FreeIPA with LDAP and Kerberos**

If a Fedora client is using an older version of SSSD, SSSD can still be configured to use the FreeIPA server as its identity provider and its Kerberos authentication domain; this is described in the SSSD configuration section of the [Fedora Deployment Guide](https://fedoraproject.org).
Any FreeIPA domain client can be configured to use `nss_ldap` and `pam_krb5` to connect to the FreeIPA 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).

### 19.1.2. Planning Password Migration

Probably the most visible issue that can impact LDAP-to-FreeIPA migration is migrating user passwords.

FreeIPA (by default) uses Kerberos for authentication and requires that each user have Kerberos hashes stored in the FreeIPA Directory Server in addition to the standard user passwords. To generate these hashes, the user password needs to be available to the FreeIPA server in cleartext. This is the case when the user is created in FreeIPA. 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 FreeIPA domain or access FreeIPA resources until they have Kerberos hashes.

If a user does not have a Kerberos hash\(^1\), that user cannot log into the FreeIPA 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 FreeIPA 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.

#### 19.1.2.1. Method 1: Using Temporary Passwords and Requiring a Change

When passwords are changed in FreeIPA, 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 FreeIPA, 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.

#### 19.1.2.2. Method 2: Using the Migration Web Page

When it is running in migration mode, FreeIPA 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

---

\(^1\) It is possible to use LDAP authentication in FreeIPA instead of Kerberos authentication, which means that Kerberos hashes are not required for users. However, this limits the capabilities of FreeIPA and is not recommended.
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.

**19.1.2.3. Method 3: Using SSSD (Recommended)**

SSSD can work with FreeIPA 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 FreeIPA 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 performs a plaintext LDAP bind over a secure connection.
5. FreeIPA intercepts this bind request. If the user has a Kerberos principal but no Kerberos hashes, then the FreeIPA identity provider generates the hashes and stores them in the user entry.
6. If authentication is successful, SSSD disconnects from FreeIPA 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 known, they simply log into a client service and it works as normal.

**19.1.2.4. 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 FreeIPA server, their cleartext passwords are not migrated over. FreeIPA 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 FreeIPA 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 19.1.2.3, “Method 3: Using SSSD (Recommended).”

**19.1.2.5. Automatically Resetting Passwords That Do Not Meet Requirements**

If user passwords in the original directory do not meet the password policies defined in FreeIPA, then the passwords must be reset after migration.

Password resets are done automatically the first time the users attempts to *kinit* into the FreeIPA domain.

```
[jsmith@server ~]$ kinit
```
19.1.3. Migration Considerations and Requirements

As you are planning migrating from an LDAP server to FreeIPA, make sure that your LDAP environment is able to work with the FreeIPA migration script.

19.1.3.1. LDAP Servers Supported for Migration

The migration process from an LDAP server to FreeIPA 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 FreeIPA has been tested with 389 Directory Server.

**NOTE**

Migration using the migration script is not supported for Microsoft Active Directory because it is not an LDAPv3-compliant directory.

19.1.3.2. Migration Environment Requirements

There are many different possible configuration scenarios for both 389 Directory Server and FreeIPA, 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 FreeIPA realm. No consolidation is involved.

- User passwords are stored as a hash in the LDAP directory that the FreeIPA 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 FreeIPA.

19.1.3.3. Migration Tools

FreeIPA 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 FreeIPA server.

The FreeIPA server must be configured to run in migration mode, and then the migration script can be used.
19.1.3.4. Migration Sequence
There are four major steps when migrating to FreeIPA, 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 a FreeIPA server is configured:

1. Deploy SSSD.
2. Reconfigure clients to connect to the current LDAP server and then fail over to FreeIPA.
3. Install the FreeIPA server.
4. Migrate the user data using the FreeIPA `ipa-migrate-ds` script. This exports the data from the LDAP directory, formats it for the FreeIPA schema, and then imports it into FreeIPA.
5. Take the LDAP server offline and allow clients to fail over to FreeIPA transparently.

With a server migration, the LDAP to FreeIPA migration comes first:

1. Install the FreeIPA server.
2. Migrate the user data using the FreeIPA `ipa-migrate-ds` script. This exports the data from the LDAP directory, formats it for the FreeIPA schema, and then imports it into FreeIPA.
3. Optional. Deploy SSSD.
4. Reconfigure clients to connect to FreeIPA. It is not possible to simply replace the LDAP server. The FreeIPA 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 FreeIPA 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 FreeIPA 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.

19.2. 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
```
Chapter 19. Migrating from an LDAP Directory to FreeIPA

It is possible to customize how the `migrate-ds` commands identifies and exports data. This is useful if the original directory tree has a unique structure or if some entries or attributes within entries should be excluded from migration.

19.2.1. 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.

19.2.2. 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.

On 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:
Excluding Entry Attributes

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:

```bash
[root@ipaserver ~]# ipa migrate-ds --group-objectclass=groupOfNames,groupOfUniqueNames ldap://ldap.example.com:389
```

Positively specifying user and groups to migrate based on object class implicitly excludes all other users and groups from migration.

Alternatively, it can be useful to migrate all user and group entries except for just a small handful of entries. Specific user or group accounts can be excluded while all others of that type are migrated. For example, this excludes a hobbies group and two users:

```bash
[root@ipaserver ~]# ipa migrate-ds --exclude-groups="Golfers Group" --exclude-users=jsmith,bjensen ldap://ldap.example.com:389
```

Specifying an object class to migrate can be used together with excluding specific entries. For example, this specifically includes users with the `fullTimeEmployee` object class, yet excludes three managers:

```bash
[root@ipaserver ~]# ipa migrate-ds --user-objectclass=fullTimeEmployee --exclude-users=jsmith,bjensen,mreynolds ldap://ldap.example.com:389
```

### 19.2.3. 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 FreeIPA 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:

```bash
[root@ipaserver ~]# ipa migrate-ds --user-ignore-attribute=userCertificate --user-ignore-objectclass=strongAuthenticationUser --group-ignore-objectclass=groupOfCertificates ldap://ldap.example.com:389
```
Chapter 19. Migrating from an LDAP Directory to FreeIPA

19.2.4. Setting the Schema to Use

By default, FreeIPA uses RFC2307bis\(^2\) schema to define user, host, hostgroup, and other network identities. This schema option can be reset to use RFC2307\(^3\) schema instead:

```
[root@ipaserver ~]# ipa migrate-ds --schema=RFC2307 ldap://ldap.example.com:389
```

19.3. Scenario 1: Using SSSD as Part of Migration

**IMPORTANT**

This is a general migration procedure, but it may not work in every environment.

It is strongly recommended that you set up a test LDAP environment and test the migration process before attempting to migrate the real LDAP environment.

1. Set up SSSD. Using SSSD allows the required Kerberos keys and server certificates to be delivered to the clients.
   a. Install SSSD on every client machine:

   ```
   # yum install sssd
   ```

   b. Configure an LDAP identity provider in SSSD to use the existing Directory Server for all functions (authentication, identity lookups, access, and password changes). This ensures every client works properly with the existing directory service.

2. Install FreeIPA, including any custom LDAP directory schema\(^4\), on a different machine from the existing LDAP directory.

3. Enable the FreeIPA server to allow migration:

   ```
   # ipa config-mod --enable-migration=TRUE
   ```

4. Disable the compat plug-in.


\(^3\) [http://www.faqs.org/rfcs/rfc2307.html](http://www.faqs.org/rfcs/rfc2307.html)

Scenario 1: Using SSSD as Part of Migration

5. Restart the FreeIPA Directory Server instance.

   # service dirsrv restart

6. Run the FreeIPA migration script, `ipa migrate-ds`. At its most basic, this requires only the LDAP URL of the LDAP directory instance to migrate:

   # ipa migrate-ds ldap://ldap.example.com:389

   Simply passing the LDAP URL migrates all of the directory data using common default settings. The user and group data can be selectively migrated by specifying other options, as covered in Section 19.2, “Examples for Using migrate-ds”.

   Once the information is exported, the script adds all required FreeIPA object classes and attributes and converts DNs in attributes to match the FreeIPA directory tree.

7. Re-enable the compat plug-in.

   # ipa-compat-manage enable


   # service dirsrv restart

9. Move clients that have SSSD installed from the LDAP backend to the FreeIPA backend and enroll them as client with FreeIPA. This downloads the required keys and certificates.

   On Fedora clients, this can be done using the `ipa-client-install` command. For example:

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

10. Have users log into a machine with SSSD and FreeIPA backend. This generates the required Kerberos keys for the user.

    To monitor the user migration process, query the existing LDAP directory to see which user accounts have a password but do not yet have a Kerberos principal key.

    ```
    $ ldapsearch -LL -x -D 'cn=Directory Manager' -w secret -b 'ou=people,dc=example,dc=com' '(&(!(krbprincipalkey=*))(|(userpassword=*)))(uid)
    ```

    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 FreeIPA domain, as required.

12. When the migration of all clients and users is complete, decommission the LDAP directory.

### 19.4. Scenario 2: Migrating an LDAP Server Directly to FreeIPA

**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 FreeIPA server, including any custom LDAP directory schema\(^5\), on a different machine from the existing LDAP directory.

2. Disable the compat plug-in.

   ```bash
   # ipa-compat-manage disable
   ```


   ```bash
   # service dirsrv restart
   ```

4. Enable the FreeIPA server to allow migration:

   ```bash
   # ipa config-mod --enable-migration=TRUE
   ```

5. Run the FreeIPA migration script, `ipa migrate-ds`. At its most basic, this requires only the LDAP URL of the LDAP directory instance to migrate:

   ```bash
   # 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 19.2, "Examples for Using migrate-ds".

   Once the information is exported, the script adds all required FreeIPA object classes and attributes and converts DNs in attributes to match the FreeIPA directory tree.

6. Re-enable the compat plug-in.

   ```bash
   # ipa-compat-manage enable
   ```

# service dirsrv restart

8. Update the client configuration to use PAM_LDAP and NSS_LDAP to connect to FreeIPA 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 19.1.2, “Planning Password Migration”.*
   a. Install SSSD on every client machine:
   ```
   # yum install sssd
   ```
   b. Run the `ipa-client-install` to configure SSSD and related services to use the FreeIPA server for identity and Kerberos authentication.

10. Instruct users to log into FreeIPA 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 FreeIPA.
   ```
   https://ipaserver.example.com/ipa/migration
   ```


```
# yum install sssd
```

```
ipa-client-install
```

```
https://ipaserver.example.com/ipa/migration
```

```
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.
Appendix A. Frequently Asked Questions

Q: Is it possible to change the IP address of the master server?
A: Yes. If you are only changing the IP address, it is sufficient to update the /etc/hosts file, the system configuration, and the DNS entry.

Q: Why are there restrictions on the length of user and group names? How can I change this?
A: User and group name lengths are specified in the policy. The default maximum username length is 32 characters. The maximum configurable length for user or group names is 255 characters. This complements some supported client operating systems which limit the length of usernames.

The default settings can be changed in the FreeIPA UI or using the ipa config-mod command. For example:

```
4 ipa config-mod --maxusername=50
```

Q: What is the difference between a replica and a master server?
A: A master server maintains a certificate authority. A replica server has its certificate issued by the master CA.

Q: Can I promote a replica to function as the master?
A: Yes. This is described in Section 18.8, “Moving CRL Generation from the Master (Original) Server to Another Replica”.

Q: Why does the ipa-client-install script fail to find the IPA server on a network that uses Active Directory DNS?
A: Active Directory has its own SRV records for Kerberos and LDAP. The ipa-client-install script can retrieve those records instead of any that have been added for the FreeIPA domain.

When running ipa-client-install, manually enter the server information to ensure that the script uses the FreeIPA SRV records instead of Active Directory records. The ipa-client-install options are listed in Section C.6.1, “ipa-client-install”.

Q: Can an administrator who is connected to “Server B” revoke a certificate issued by “Server A”?
A: Yes, assuming that Servers A and B contain non-cloned CAs which have their database information replicated to share revocation information only.
Appendix B. Working with certmonger

Part of managing machine authentication is managing machine certificates. On clients, FreeIPA manages the certificate lifecycle with the certmonger service, which works together with the certificate authority (CA) provided by FreeIPA.

The certmonger daemon and its command-line clients simplify the process of generating public/private key pairs, creating certificate requests, and submitting requests to the CA for signing. As part of managing certificates, the certmonger daemon monitors certificates for expiration and can renew certificates that are about to expire. The certificates that certmonger monitors are tracked in files stored in a configurable directory. The default location is `/var/lib/certmonger/requests`.

certmonger uses the FreeIPA getcert command to manage all certificates. As covered in Section 2.3.3.2, “Using Different CA Configurations”, an FreeIPA server can be configured to use different types of certificate authorities. The most common (and recommended) configuration is to use a full CA server, but it is also possible to use a much more limited, self-signed CA. The exact getcert command used by certmonger to communicate with the FreeIPA backend depends on which type of CA is used. The `ipa-getcert` command is used with a full CA, while the `selfsign-getcert` command is used with a self-signed CA.

NOTE

Because of general security issues, self-signed certificates are not typically used in production, but can be used for development and testing.

B.1. Requesting a Certificate with certmonger

With the FreeIPA CA, certmonger uses the `ipa-getcert` command.

Certificates and keys are stored locally in plaintext files (.pem) or in an NSS database, identified by the certificate nickname. When requesting a certificate, then, the request should identify the location where the certificate will be stored and the nickname of the certificate. For example:

```
# ipa-getcert request -d /etc/pki/nssdb -n Server-Cert
```

The `/etc/pki/nssdb` file is the global NSS database, and `Server-Cert` is the nickname of this certificate. The certificate nickname must be unique within this database.

When requesting a certificate to be used with a FreeIPA service, the `-K` option is required to specify the service principal. Otherwise, certmonger assumes the certificate is for a host. The `-N` option must specify the certificate subject DN, and the subject base DN must match the base DN for the FreeIPA server, or the request is rejected.

```
$ ipa-getcert request -d /etc/httpd/alias -n Server-Cert -K HTTP/client1.example.com -N 'CN=client1.example.com,O=EXAMPLE.COM'
```

Example B.1. Using certmonger for a Service

```
$ ipa-getcert request -r -f /etc/httpd/conf/ssl.crt/server.crt -k /etc/httpd/conf/ssl.key/server.key -N CN='hostname --fqdn' -D 'hostname' -U id-kp-serverAuth
```
Appendix B. Working with certmonger

The options vary depending on whether you are using a self-signed certificate (selfsign-getcert) and the desired configuration for the final certificate, as well as other settings. In Example B.1, “Using certmonger for a Service”, these are common options:

- The \texttt{-r} option will automatically renew the certificate if the key pair already exists. This is used by default.
- The \texttt{-f} option stores the certificate in the given file.
- The \texttt{-k} option either stores the key in the given file or, if the key file already exists, uses the key in the file.
- The \texttt{-N} option gives the subject name.
- The \texttt{-D} option gives the DNS domain name.
- The \texttt{-U} option sets the extended key usage flag.

B.2. Storing Certificates in NSS Databases

By default, \texttt{certmonger} uses plaintext files to store the key and the certificate, but these keys and certificates can also be stored in NSS databases. This is done using the \texttt{-d} option to set the security database location and \texttt{-n} to give the certificate nickname which is used for the certificate in the database. These options are used instead of the PEM files given in the \texttt{-f} and \texttt{-k} options.

For example:

```
# ipa-getcert request -d /export/alias -n ServerCert ...
```

B.3. Tracking Certificates with certmonger

\texttt{certmonger} can manage the entire certificate lifecycle. Along with generating requests, \texttt{certmonger} can track a certificate and automatically renew it when it expires at the end of its validity period.

This is done using the \texttt{start-tracking} command with the \texttt{getcert} command. The \texttt{-I} option creates the tracking entry, along with pointers to the key and certificate files, either in an NSS database (\texttt{-d} and \texttt{-n}) or in the PEM file (\texttt{-f} and \texttt{-k}). The \texttt{-r} option tells \texttt{certmonger} to renew the certificate.

```
# ipa-getcert start-tracking -I cert1-tracker -d /export/alias -n ServerCert -r
```

\textbf{TIP}

The \texttt{-r} option can be passed with the \texttt{request} command, in Example B.1, “Using certmonger for a Service”. In that case, the requested certificate is automatically tracked and renewed by \texttt{certmonger}. Then, it is not necessary to configure tracking manually.

A certificate can be untracked by \texttt{certmonger} by using the \texttt{stop-tracking} command.
Appendix C. FreeIPA Tools Reference

FreeIPA provides a variety of command-line tools that can be used to manage FreeIPA servers and clients, allowing an extensive range of functionality that can be managed remotely, scheduled, or scripted.

C.1. Using Special Characters with FreeIPA Tools

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

C.2. ipa

The primary FreeIPA command-line tool is the **ipa** command. This command has several dozen subcommands, grouped by configuration areas, to give specific control over different areas of the FreeIPA domain configuration. These subcommands are really plug-ins, that are implemented and called through the **ipa** command.

One crucial feature of the **ipa** command is that it is **pluggable**. Custom behavior can be defined for the FreeIPA domain through custom **ipa** subcommands.

C.2.1. Location

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool directory</td>
<td>/usr/bin</td>
</tr>
<tr>
<td>Package</td>
<td>ipa-admintools</td>
</tr>
</tbody>
</table>

C.2.2. Syntax

ipa [global_options] commands [command_options]

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

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

ipa objectType-operation objectName --option=value

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

ipa user-add entryName options

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

C.2.2.1. Adding, Editing, and Deleting Entries with ipa

New entries are added using an *-add command. For example:
$ ipa user-add jsmith

For **add** operations, commands usually prompt for any required configuration attributes, which can be passed as command-line options or using **--set**/*addattr** options (Section C.2.5, "Managing Attributes with **--setattr**, **--addattr**, and **--delattr**").

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

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

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

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

$ ipa user-del jsmith

### C.2.2.2. Finding and Displaying Entries with ipa

Entries for an entire type are searched for using the **-*find** command and an optional search criterion. The criterion is a string which can either be an exact match or use an asterisk (*) as a wildcard.

ipa user-find *smith

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

Searches (any **-*find** command) have certain limits imposed as part of the server configuration, specifically how many entries are returned (size limits) and how long a search will run (time limits). This is covered in Section 4.4.2, "Setting FreeIPA Search Limits". Part of the server configuration is setting global defaults for size and time limits on searches. While these limits are always enforced in the web UI, they can be overridden with any **-*find** command with the **-*sizelimit** and **-*timelimit** options. For example, if the default time limit is 60 seconds and a search is going to take longer, the time limit can be increased to 120 seconds:

[jsmith@ipaserver ~]$ ipa user-find *sen --timelimit=120

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

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

To display a specific entry, use the **-*show** command and the entry name. As with searches, only a subset of attributes are displayed with the entry unless the **-*all** option is used.
### C.2.2.3. Adding Members to Groups and Containers with ipa

Group members are added and removed with separate commands, apart from simply modifying an entry. Member commands essentially create a relationship between different FreeIPA entries. While this is obvious in traditional group-member roles, it is also true for some policy entries (like SELinux and sudo policies) where entries are associated with another entry.

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

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

### C.2.2.4. Positional Elements in ipa Commands

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

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

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

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

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

### C.2.3. Help Topics

All of the ipa plug-ins (subcommands) are loosely organized in groups, based on the configuration area that they relate to. These groups are called topics, and the ipa help information can be called for each topic group.

```
$ ipa help topicName
```

For example, to view the description of how FreeIPA handles automount and a list of all commands for managing automount configuration, view the help topic for automount:

```
$ ipa help automount
```

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>automount</td>
<td>Adding and managing automount and NFS configuration.</td>
</tr>
<tr>
<td>cert</td>
<td>Managing certificate operations.</td>
</tr>
<tr>
<td>config</td>
<td>Managing the FreeIPA server configuration.</td>
</tr>
<tr>
<td>delegation</td>
<td>Setting and controlling authorization delegated between groups.</td>
</tr>
<tr>
<td>dns</td>
<td>Creating and managing the DNS entries within the FreeIPA DNS domain.</td>
</tr>
<tr>
<td>group</td>
<td>Creating groups of users.</td>
</tr>
</tbody>
</table>
### Appendix C. FreeIPA Tools Reference

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbac</td>
<td>Setting and testing host-based access controls.</td>
</tr>
<tr>
<td>host</td>
<td>Creating and managing client (host) entries within the FreeIPA domain.</td>
</tr>
<tr>
<td>hostgroup</td>
<td>Creating and managing groups of hosts.</td>
</tr>
<tr>
<td>krbtpolicy</td>
<td>Managing the Kerberos ticket policy.</td>
</tr>
<tr>
<td>migration</td>
<td>Managing migration to FreeIPA.</td>
</tr>
<tr>
<td>misc</td>
<td>Viewing current environment variables and plugins.</td>
</tr>
<tr>
<td>netgroup</td>
<td>Managing netgroups within the FreeIPA domain.</td>
</tr>
<tr>
<td>passwd</td>
<td>Managing user passwords.</td>
</tr>
<tr>
<td>permission</td>
<td>Setting access control rules for users, groups, and roles within FreeIPA to FreeIPA resources.</td>
</tr>
<tr>
<td>privilege</td>
<td>Managing a group of permissions.</td>
</tr>
<tr>
<td>pwpolicy</td>
<td>Managing the FreeIPA domain password policy.</td>
</tr>
<tr>
<td>role</td>
<td>Creating and managing user roles, as part of access control.</td>
</tr>
<tr>
<td>selfservice</td>
<td>Managing rights that users have to their own personal FreeIPA entries.</td>
</tr>
<tr>
<td>service</td>
<td>Creating and managing system services that are managed as an FreeIPA resource.</td>
</tr>
<tr>
<td>sudo</td>
<td>Creating and managing sudo rules and policies.</td>
</tr>
<tr>
<td>user</td>
<td>Creating and managing FreeIPA user accounts.</td>
</tr>
</tbody>
</table>

#### C.2.4. Global Options

Global options are available to every subcommand.

<table>
<thead>
<tr>
<th>Short Option</th>
<th>Long Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>--help</td>
<td>Prints the help for the command and exits.</td>
</tr>
<tr>
<td>-e key=value</td>
<td></td>
<td>Sets a given environment variable (key) to the specified value before running the command.</td>
</tr>
<tr>
<td>-c file</td>
<td></td>
<td>Loads the server configuration from a different file instead of default.conf.</td>
</tr>
<tr>
<td>-d</td>
<td>--debug</td>
<td>Uses debug logging when running the command.</td>
</tr>
<tr>
<td>-v</td>
<td>--verbose</td>
<td>Prints verbose messages to stdout when running the command. If two -v options are used, then the command returns the full XML-RPC request.</td>
</tr>
</tbody>
</table>
### C.2.5. Managing Attributes with --setattr, --addattr, and --delattr

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

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

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

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

Both options have this format:

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

The `--addattr` option adds a new value for an attribute; for a multi-valued attribute, it adds the new value while preserving any existing values.

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

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

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

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

**NOTE**

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

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

### C.2.6. Return Codes

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>An error occurred.</td>
</tr>
<tr>
<td>1</td>
<td>The operation was successful.</td>
</tr>
<tr>
<td>2</td>
<td>A resource or object was not found.</td>
</tr>
</tbody>
</table>

### C.2.7. Commands

**Table C.1. dnsrecord* Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dnsrecord-add</td>
<td>Creates a new DNS zone in the FreeIPA server.</td>
</tr>
<tr>
<td>dnsrecord-del</td>
<td>Deletes a DNS zone from the DNS domain maintained by the FreeIPA server.</td>
</tr>
<tr>
<td>dnsrecord-find</td>
<td>Searches for a DNS zone which matches the filter.</td>
</tr>
<tr>
<td>dnsrecord-mod</td>
<td>Edits the configuration of an existing DNS domain.</td>
</tr>
<tr>
<td>dnsrecord-show</td>
<td>Lists the details for any or all DNS zones, depending on the filter</td>
</tr>
</tbody>
</table>
Table C.2. dnszone-* Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dnszone-add</td>
<td>Creates a new DNS zone in the FreeIPA server.</td>
</tr>
<tr>
<td>dnszone-del</td>
<td>Deletes a DNS zone from the DNS domain maintained by the FreeIPA server.</td>
</tr>
<tr>
<td>dnszone-find</td>
<td>Searches for a DNS zone which matches the filter.</td>
</tr>
<tr>
<td>dnszone-mod</td>
<td>Edits the configuration of an existing DNS domain.</td>
</tr>
<tr>
<td>dnszone-show</td>
<td>Lists the details for any or all DNS zones, depending on the filter</td>
</tr>
<tr>
<td>dnszone-disable</td>
<td>Disables a DNS zone, which removes it from being used but does not delete the zone or its configuration.</td>
</tr>
<tr>
<td>dnszone-enable</td>
<td>Enables an existing DNS zone, which restores it to the FreeIPA domain with its previous configuration intact.</td>
</tr>
</tbody>
</table>

C.3. ipa DNS Commands

There are a collection of commands used to add and manage DNS zones and DNS records.

C.3.1. ipa dnszone-add

Creates a new DNS zone in the FreeIPA server.

C.3.1.1. Syntax

```
```

C.3.1.2. Options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoneName</td>
<td>Sets the name of the new zone. This is required.</td>
</tr>
<tr>
<td>--name-server=name</td>
<td>Gives the name of the authoritative DNS server to use, based on the hostname.</td>
</tr>
<tr>
<td>--ip-address=IPaddress</td>
<td>Sets a DNS name server to use with the FreeIPA domain, based on the IP address.</td>
</tr>
<tr>
<td>--name-from-ip=IPaddress</td>
<td>Gives an IP address for the DNS zone to use to create a reverse zone name.</td>
</tr>
<tr>
<td>--admin-email=email</td>
<td>Gives the email address of the DNS domain administrator.</td>
</tr>
<tr>
<td>--serial=#</td>
<td>Sets the serial number to use for the DNS start of authority (SOA) record.</td>
</tr>
</tbody>
</table>
## Appendix C. FreeIPA Tools Reference

### C.3.2. ipa dnsrecord-add

Adds a new resource record.

Each resource record type has its own set of options, based on the required information to configure the record.

### C.3.2.1. Syntax

```
ipa dnsrecord-add zoneName recordName [ --recordType_options=value ]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--refresh=#</code></td>
<td>Sets the time interval to use to check for updated SOA records.</td>
</tr>
<tr>
<td><code>--retry=#</code></td>
<td>Sets the time to wait before retrying a failed name resolution attempt.</td>
</tr>
<tr>
<td><code>--expire=#</code></td>
<td>Sets the time to hold successful resolution attempts in the cache before they expire.</td>
</tr>
<tr>
<td><code>--minimum=#</code></td>
<td>Sets the time to hold negative or failed resolution attempts in the cache.</td>
</tr>
<tr>
<td><code>--ttl=#</code></td>
<td>Sets how long to hold SOA records in the cache.</td>
</tr>
<tr>
<td><code>--class=name</code></td>
<td>Sets a SOA record class for zone entries.</td>
</tr>
<tr>
<td><code>--update-policy=string</code></td>
<td>Sets a BIND service policy.</td>
</tr>
<tr>
<td>`--dynamic-update=TRUE</td>
<td>FALSE`</td>
</tr>
<tr>
<td><code>--allow-query=string</code></td>
<td>Gives a semi-colon-separated listed of IP addresses or network names which are allowed to issue DNS queries.</td>
</tr>
<tr>
<td><code>--allow-transfer=string</code></td>
<td>Gives a semi-colon-separated listed of IP addresses or network names which are allowed to transfer the given zone.</td>
</tr>
<tr>
<td>`--allow-sync-ptr=TRUE</td>
<td>FALSE`</td>
</tr>
<tr>
<td><code>--forwarder=string</code></td>
<td>Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the FreeIPA domain. To specify multiple forwarders, use the option multiple times.</td>
</tr>
<tr>
<td>`--forward-policy=only</td>
<td>first`</td>
</tr>
<tr>
<td><code>--force</code></td>
<td>Forces a zone to be created in the FreeIPA configuration even if the name server is not recognized by the DNS.</td>
</tr>
</tbody>
</table>
### C.3.2.2. Options

<table>
<thead>
<tr>
<th>General Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--ttl=number</td>
<td>Sets the time to live for the record.</td>
</tr>
<tr>
<td></td>
<td>--class=IN</td>
<td>CS</td>
</tr>
<tr>
<td></td>
<td>--structured</td>
<td>Parses the raw DNS records and returns them in a structured format.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;A&quot; Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--a-rec=ARECORD</td>
<td>Passes a comma-separated list of A records.</td>
</tr>
<tr>
<td></td>
<td>--a-ip-address=string</td>
<td>Gives the IP address for the record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;A6&quot; Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--a6-rec=A6RECORD</td>
<td>Passes a comma-separated list of A6 records.</td>
</tr>
<tr>
<td></td>
<td>--a6-data=string</td>
<td>Gives the data for the record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;AAAA&quot; Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--aaaa-rec=AAAARECORD</td>
<td>Passes a comma-separated list of AAAA (IPv6) records.</td>
</tr>
<tr>
<td></td>
<td>--aaaa-ip-address=string</td>
<td>Gives the IPv6 address for the record.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;AFSDB&quot; Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--afsdb-rec=AFSDBRECORD</td>
<td>Passes a comma-separated list of AFSDB records.</td>
</tr>
<tr>
<td></td>
<td>--afsdb-subtype=1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>--afsdb-hostname=string</td>
<td>Gives the hostname of the AFS database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>&quot;CERT&quot; Record Options</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--cert-rec=CERTRECORD</td>
<td>Passes a comma-separated list of CERT records.</td>
</tr>
<tr>
<td></td>
<td>--cert-type=number</td>
<td>Gives the type of certificate. Common values include 1 (X.509 certificate), 3 (PGP packet), 4 (URL of an X.509 certificate), and 254 (private OID).</td>
</tr>
<tr>
<td></td>
<td>--cert-key-tag=number</td>
<td>Contains a 16-bit computed value that represents the certificate key. This is generated based on the given key algorithm in the certificate.</td>
</tr>
<tr>
<td></td>
<td>--cert-algorithm=number</td>
<td>Contains the algorithm used to create the public key.</td>
</tr>
<tr>
<td></td>
<td>--cert-certificate-or-crl=number</td>
<td>Contains the base 64-encoded blob of the X.509 object (certificate or CRL).</td>
</tr>
</tbody>
</table>
### "CNAME" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--cname-rec=CNAME RECORD</td>
<td>Passes a comma-separated list of CNAME records.</td>
</tr>
<tr>
<td>--cname-hostname=string</td>
<td>Gives the real hostname that the CNAME alias hostname points to.</td>
</tr>
</tbody>
</table>

### "DNAME" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dname-rec=DNAMERECORD</td>
<td>Passes a comma-separated list of DNAME records.</td>
</tr>
<tr>
<td>--dname-target=string</td>
<td>Gives the domain name of the target DNS domain.</td>
</tr>
</tbody>
</table>

### "DS" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ds-rec=DSRECORD</td>
<td>Passes a comma-separated list of DNSSEC key records.</td>
</tr>
<tr>
<td>--ds-key-tag=number</td>
<td>Contains a 16-bit computed value that represents the certificate key. This is generated based on the given key algorithm.</td>
</tr>
<tr>
<td>--ds-algorithm=number</td>
<td>Contains the algorithm used to create the public key.</td>
</tr>
<tr>
<td>--ds-digest-type=number</td>
<td>Contains the algorithm used to create the digest. This is 1, for SHA-1.</td>
</tr>
<tr>
<td>--ds-digest=string</td>
<td>Contains the digest for the DNSKEY. This is calculated by combining the fully qualified owner name of the DNSKEY resource record, the DNSKEY data, and the digest algorithm.</td>
</tr>
</tbody>
</table>

### "KEY" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--key-rec=KEYRECORD</td>
<td>Passes a comma-separated list of KEY records.</td>
</tr>
<tr>
<td>--key-flags=number</td>
<td>Gives a series of bits the indicate aspects of the key. Bits 0-1 show key usage, bits 6-7 show the key entity type (such as a user or account), and bits 12-15 show the key signature settings. For more information, see <a href="http://tools.ietf.org/html/rfc2535#section-3">http://tools.ietf.org/html/rfc2535#section-3</a>.</td>
</tr>
<tr>
<td>--key-protocol=number</td>
<td>Shows what Internet protocols the key can be used with. This value can be 1 (TLS), 2 (email), 3 (DNSSEC), 4 (IPSEC), or 255 (all).</td>
</tr>
<tr>
<td>--key-algorithm=number</td>
<td>Contains the algorithm used to create the key.</td>
</tr>
<tr>
<td>--key-public-key=number</td>
<td>Contains the public key.</td>
</tr>
</tbody>
</table>

### "KX" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--kx-rec=KXRECORD</td>
<td>Passes a comma-separated list of KX records.</td>
</tr>
</tbody>
</table>
### "KX" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--kx-preference=number</td>
<td>Sets the priority of the specified key exchanger. The lower the number, the higher priority.</td>
</tr>
<tr>
<td>--kx-exchanger=string</td>
<td>Gives the hostname of a server which can act as a key exchanger.</td>
</tr>
</tbody>
</table>

### "LOC" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--loc-rec=LOCRECORD</td>
<td>Passes a comma-separated list of location information records.</td>
</tr>
<tr>
<td>--loc-lat-deg=number</td>
<td>Gives the degrees element of the latitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lat-min=number</td>
<td>Gives the minutes element of the latitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lat-sec=decimal</td>
<td>Gives the seconds element of the latitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lat-dir=N</td>
<td>S</td>
</tr>
<tr>
<td>--loc-lon-deg=number</td>
<td>Gives the degrees element of the longitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lon-min=number</td>
<td>Gives the minutes element of the longitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lon-sec=decimal</td>
<td>Gives the seconds element of the longitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-lon-dir=E</td>
<td>W</td>
</tr>
<tr>
<td>--loc-altitude=decimal</td>
<td>Gives the altitude of the physical location of a network object.</td>
</tr>
<tr>
<td>--loc-size=decimal</td>
<td>Gives the diameter of a circle encompassing the object, to give an idea of its physical size.</td>
</tr>
<tr>
<td>--loc-h-precision=decimal</td>
<td>Gives the diameter of a circle which constitutes the margin of error for estimating the physical size of the object.</td>
</tr>
<tr>
<td>--loc-v-precision=decimal</td>
<td>Gives the diameter of a circle which constitutes the margin of error for estimating the altitude of the object.</td>
</tr>
</tbody>
</table>

### "MX" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--mx-rec=MXRECORD</td>
<td>Passes a comma-separated list of MX records.</td>
</tr>
<tr>
<td>--mx-preference=number</td>
<td>Sets the priority of the specified mail server. The lower the number, the higher priority.</td>
</tr>
<tr>
<td>--mx-exchanger=string</td>
<td>Gives the hostname of a mail server.</td>
</tr>
</tbody>
</table>

### "NAPTR" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
### "NAPTR" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--naptr-rec=NAPTRRECORD</td>
<td>Passes a comma-separated list of pointer records.</td>
</tr>
<tr>
<td>--naptr-order=number</td>
<td>Contains a 16-bit unsigned number which sets that order to use to process naming authority pointer records.</td>
</tr>
<tr>
<td>--naptr-preference=number</td>
<td>Contains a 16-bit unsigned number which is the secondary method used to determine the weight or priority of naming authority pointer records with the same order value.</td>
</tr>
<tr>
<td>--naptr-flags=string</td>
<td>Contains an application-specific series of characters which sets how to manage field values.</td>
</tr>
<tr>
<td>--naptr-service=string</td>
<td>Contains application-specific values which are used to configure the NAPTR service.</td>
</tr>
<tr>
<td>--naptr-regexp=string</td>
<td>Contains a substitution expression which is used to help look up the next domain name in the queue.</td>
</tr>
<tr>
<td>--naptr-replacement=string</td>
<td>Contains a fully-qualified domain name to use with <code>--naptr-regexp</code> if the expression is a simple replacement.</td>
</tr>
</tbody>
</table>

### "NS" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ns-rec=NSRECORD</td>
<td>Passes a comma-separated list of NS records.</td>
</tr>
<tr>
<td>--ns-hostname=string</td>
<td>Gives the hostname of the authoritative name server.</td>
</tr>
</tbody>
</table>

### "NSEC" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--nsec-rec=NSECRECORD</td>
<td>Passes a comma-separated list of NSEC key records.</td>
</tr>
<tr>
<td>--nsec-next=string</td>
<td>Gives the next domain name in the canonical order of the DNS zone.</td>
</tr>
<tr>
<td>--nsec-types=types</td>
<td>Contains a list of record types supported in the domain. The types can be SOA, A, AAAA, A6, AFSDB, APL, CERT, CNAME, DHCID, DLV, DNAME, DNSKEY, DS, HIP, IPSECKEY, KEY, KX, LOC, MX, NAPTR, NS, NSEC, NSEC3, NSEC3PARAM, PTR, RRSIG, RP, SIG, SPF, SRV, SSHFP, TA, TKEY, TSIG, TXT.</td>
</tr>
</tbody>
</table>

### "PTR" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ptr-rec=PTRRECORD</td>
<td>Passes a comma-separated list of PTR records.</td>
</tr>
<tr>
<td>--ptr-hostname=string</td>
<td>Gives the hostname for the record.</td>
</tr>
</tbody>
</table>

### "RRSIG" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
### "RRSIG" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--rrsig-rec=RRSIGRECORD</td>
<td>Passes a comma-separated list of RRSIG key records.</td>
</tr>
<tr>
<td>--rrsig-algorithm=number</td>
<td>Contains the algorithm used to sign the resource record set.</td>
</tr>
<tr>
<td>--rrsig-key-tag=number</td>
<td>Contains a 16-bit computed value that represents the key used to sign the resource record set.</td>
</tr>
<tr>
<td>--rrsig-labels=number</td>
<td>Gives the number of labels in the owner name; this is used to validate the signature.</td>
</tr>
<tr>
<td>--rrsig-type-covered=types</td>
<td>Identified what resource record type is covered in the specific RRSIG set. The types can be SOA, A, AAAA, A6, AFSDB, APL, CERT, CNAME, DHCID, DLV, DNAME, DNSKEY, DS, HIP, IPSECKEY, KEY, KX, LOC, MX, NAPTR, NS, NSEC, NSEC3, NSEC3PARAM, PTR, RRSIG, RP, SIG, SPF, SRV, SSHFP, TA, TKEY, TSIG, TXT.</td>
</tr>
<tr>
<td>--rrsig-original-ttl=number</td>
<td>Gives the time-to-live of the records set in the authoritative zone.</td>
</tr>
<tr>
<td>--rrsig-signature-inception=string</td>
<td>Gives the beginning date of the record set's validity period.</td>
</tr>
<tr>
<td>--rrsig-signature-expiration=string</td>
<td>Gives the end date of the record set's validity period.</td>
</tr>
<tr>
<td>--rrsig-signers-name=string</td>
<td>Gives the name of the signer of the DNSKEY for the record set. This must include the name of the zone.</td>
</tr>
<tr>
<td>--rrsig-signature=string</td>
<td>Contains the digital signature for the record set.</td>
</tr>
</tbody>
</table>

### "SIG" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--sig-rec=SIGRECORD</td>
<td>Passes a comma-separated list of SIG key records.</td>
</tr>
<tr>
<td>--sig-algorithm=number</td>
<td>Contains the algorithm used to sign the resource record set.</td>
</tr>
<tr>
<td>--sig-key-tag=number</td>
<td>Contains a 16-bit computed value that represents the key used to sign the resource record set.</td>
</tr>
<tr>
<td>--sig-labels=number</td>
<td>Gives the number of labels in the owner name; this is used to validate the signature.</td>
</tr>
<tr>
<td>--sig-type-covered=types</td>
<td>Identified what resource record type is covered in the specific RRSIG set. The types can be SOA, A, AAAA, A6, AFSDB, APL, CERT, CNAME, DHCID, DLV, DNAME, DNSKEY, DS, HIP, IPSECKEY, KEY, KX, LOC, MX, NAPTR, NS, NSEC, NSEC3, NSEC3PARAM, PTR, RRSIG, RP, SIG, SPF, SRV, SSHFP, TA, TKEY, TSIG, TXT.</td>
</tr>
</tbody>
</table>
### “SIG” Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--sig-original-ttl=number</code></td>
<td>Gives the time-to-live of the records set in the authoritative zone.</td>
</tr>
<tr>
<td><code>--sig-signature-inception=string</code></td>
<td>Gives the beginning date of the record set's validity period.</td>
</tr>
<tr>
<td><code>--sig-signature-expiration=string</code></td>
<td>Gives the end date of the record set's validity period.</td>
</tr>
<tr>
<td><code>--sig-signers-name=string</code></td>
<td>Gives the name of the signer of the DNSKEY for the record set. This must include the name of the zone.</td>
</tr>
<tr>
<td><code>--sig-signature=string</code></td>
<td>Contains the digital signature for the record set.</td>
</tr>
</tbody>
</table>

### “SRV” Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--srv-rec=SRVRECORD</code></td>
<td>Passes a comma-separated list of SRV records.</td>
</tr>
<tr>
<td><code>--srv-priority=number</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>--srv-weight=number</code></td>
<td>Sets the weight of the record. This helps determine the order of SRV records with the same priority.</td>
</tr>
<tr>
<td><code>--srv-port=number</code></td>
<td>Gives the port for the service on the target host.</td>
</tr>
<tr>
<td><code>--srv-target=string</code></td>
<td>Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.</td>
</tr>
</tbody>
</table>

### “SSHFP” Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--sshfp-rec=SSHFPRECORD</code></td>
<td>Passes a comma-separated list of SSH fingerprint records.</td>
</tr>
<tr>
<td><code>--sshfp-algorithm=number</code></td>
<td>Contains the algorithm used for the SSH fingerprint.</td>
</tr>
<tr>
<td><code>--sshfp-fp-type=1</code></td>
<td>Identifies the type of fingerprint. This value is always 1, for SHA-1.</td>
</tr>
<tr>
<td><code>--sshfp-fingerprint=string</code></td>
<td>Contains the fingerprint, as calculated from the public key.</td>
</tr>
</tbody>
</table>

### “TXT” Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--txt-rec=TXTRECORD</code></td>
<td>Passes a comma-separated list of TXT records.</td>
</tr>
<tr>
<td><code>--txt-data=string</code></td>
<td>Gives text data for the record.</td>
</tr>
</tbody>
</table>
C.4. ipa Host Commands

This is a collection of subcommands which manage the client, or host, configuration within the FreeIPA domain.

C.4.1. ipa host-add

Adds a new FreeIPA client machine to the domain manually. This is done as part of the enrollment process.

C.4.1.1. Syntax

```
ipa host-add hostname [--desc="description"] [--locality="cityName"] [--location="location"] [--platform="hardware_description"] [--os="OS_version"] [--password="enrollment_password"] [--random] [--certificate="base64_cert"] [--force] [--no-reverse] [--ip-address=IPAddress]
```

C.4.1.2. Options

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>choice=&quot;plain&quot;&gt; hostname</td>
<td>Sets the name of the host entry to create, based on the fully-qualified domain name of the host.</td>
</tr>
<tr>
<td><strong>IMPORTANT</strong></td>
<td>This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.</td>
</tr>
<tr>
<td>--desc=&quot;description&quot;</td>
<td>Gives a text description of the host.</td>
</tr>
<tr>
<td>--locality=&quot;cityName&quot;</td>
<td>Gives the geographical location, usually a city, where the host machine is.</td>
</tr>
<tr>
<td>--location=&quot;location&quot;</td>
<td>Gives the physical location, like a colocation facility, room number, or rack, where the host machine is.</td>
</tr>
<tr>
<td>--platform=&quot;hardware_description&quot;</td>
<td>Gives a hardware description of the host machine.</td>
</tr>
<tr>
<td>--os=&quot;OS_version&quot;</td>
<td>Gives the operating system type, such as &quot;Fedora 16&quot;.</td>
</tr>
<tr>
<td>--password=&quot;enrollment_password&quot;</td>
<td>For a bulk enrollment, manually sets a password to use to proceed with the enrollment.</td>
</tr>
<tr>
<td>--random</td>
<td>For a bulk enrollment, randomly generates a password to use to proceed with the enrollment.</td>
</tr>
<tr>
<td>--certificate=&quot;base64_cert&quot;</td>
<td>Contains the full base 64-encoded blob for a server certificate for the host.</td>
</tr>
</tbody>
</table>
Appendix C. FreeIPA Tools Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--force</td>
<td>Forces the server to create a host entry with the given name, even if the hostname cannot be resolved in DNS.</td>
</tr>
<tr>
<td>--no-reverse</td>
<td>Disables reverse DNS creation through the server installation process. (If a reverse DNS zone is already configured, then that existing reverse DNS zone is used.) If this option is not used, then the default value is true, which assumes that reverse DNS should be configured by the installation script.</td>
</tr>
<tr>
<td>--ip-address=IPaddress</td>
<td>Adds the host machine using its IP address, rather than the hostname.</td>
</tr>
</tbody>
</table>

C.5. Server Scripts

These are scripts that are used to manage the FreeIPA server configuration. These scripts do not manage FreeIPA domain entries; they manage the configuration of the server itself. This means that these scripts are run as system administrative users rather than domain administrative users.

C.5.1. A Quick Summary of Configuration Scripts

There are two basic categories of command-line scripts that FreeIPA has available.

The first is a selection of independent configuration scripts which are used to set up machines that operate within the domain. Most of these scripts are used to add servers and clients to the FreeIPA domain, such as the `ipa-server-install`, `ipa-client-install`, and `ipa-replica-install` scripts.

Some scripts are also available to enable services within the FreeIPA server, such as adding CA services to a replica or enabling DNS. Several services can be configured when a server or replica is installed, but few of those services are required. If a server is configured without a service like DNS, that service can be enabled later using a specific configuration script, such as `ipa-dns-install`.

Then there are scripts that are simply used to manage FreeIPA domain. These scripts can be used to create or delete keytabs and certificates for any service or host within the domain, to set management rules for hosts and services (managed by settings).

Then there are a handful of scripts which are strictly internal, like `ipa-join`, meaning they are called by other management scripts but are not invoked by administrators directly.

Table C.3. FreeIPA Scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipa-ca-install</td>
<td>Installs a Dogtag CA on a replica which previously did not have a CA installed.</td>
</tr>
<tr>
<td>ipa-join</td>
<td>Adds a client or host machine to the FreeIPA domain. <em>Used by other scripts; not invoked by administrators directly.</em></td>
</tr>
<tr>
<td>ipa-replica-prepare</td>
<td>Pulls the configuration from a FreeIPA server to use to set up a replica.</td>
</tr>
<tr>
<td>ipa-client-install</td>
<td>Sets up the host entry for a Fedora system and configures all associated services.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ipa-ldap-updater</td>
<td>Updates the 389 Directory Server databases used by the FreeIPA server. Used by other scripts; not invoked by administrators directly.</td>
</tr>
<tr>
<td>ipa-rmkeytab</td>
<td>Deletes a keytab generated for a FreeIPA user, service, or host.</td>
</tr>
<tr>
<td>ipa-compat-manage</td>
<td>Enables or disables the compatibility plug-in to use with NIS servers running with RFC 2307 schema.</td>
</tr>
<tr>
<td>ipa-managed-entries</td>
<td>Enables or disables the 389 Directory Server Managed Entries Plug-in which manages user private groups.</td>
</tr>
<tr>
<td>ipa-server-certinstall</td>
<td>Replaces the current SSL certificates used with the FreeIPA server with new certificates. (The certificates must be generated externally, such as using the <code>pk12util</code> tool.)</td>
</tr>
<tr>
<td>ipa-csreplica-manage</td>
<td>Manages the replication agreements between the 389 Directory Server instances used by the certificate authorities for a server or replica.</td>
</tr>
<tr>
<td>ipa-nis-manage</td>
<td>Enables or disables the NIS listener plug-in.</td>
</tr>
<tr>
<td>ipa-server-install</td>
<td>Installs all of the backend servers used by a FreeIPA server and configures associated services and domains.</td>
</tr>
<tr>
<td>ipa-upgradeconfig</td>
<td>Upgrades the Apache instance used by the FreeIPA web UI. Used by other scripts; not invoked by administrators directly.</td>
</tr>
<tr>
<td>ipa-replica-conncheck</td>
<td>Checks the connection between a FreeIPA server and a replica. This is typically run internally when a replica is installed (with <code>ipa-replica-install</code>), but can be run manually to troubleshoot connection issues.</td>
</tr>
<tr>
<td>ipa-dns-install</td>
<td>Installs and configures a DNS server for a FreeIPA server which was initially configured without DNS.</td>
</tr>
<tr>
<td>ipa-replica-install</td>
<td>Installs and configures a FreeIPA replica instance, based on the submitted FreeIPA server configuration which was extracted using <code>ipa-replica-prepare</code>.</td>
</tr>
<tr>
<td>ipa-getcert</td>
<td>Submits requests and manages certificates for user which are issued and tracked using the Certmonger utility.</td>
</tr>
<tr>
<td>ipa-getkeytab</td>
<td>Requests a new keytab for a FreeIPA service, host, or user.</td>
</tr>
<tr>
<td>ipa-replica-manage</td>
<td>Edits the replication agreements between FreeIPA servers and replicas and synchronization agreements between the FreeIPA server and Active Directory domain controller.</td>
</tr>
</tbody>
</table>
C.5.2. ipa-replica-install

Uses a configuration file based on an existing FreeIPA server to create a replica, or copy, of that server. Once the replica is created, it functions as an equal participant and mirror of the original FreeIPA server within the FreeIPA domain. Any changes made on the server or any other replica are automatically propagated over to the other replicas and server.

A replica is created using a file that contains all of the configuration for the FreeIPA server. This initial file is created by running the `ipa-replica-prepare` on the FreeIPA server. Then the file is copied over to the replica machine, and the `ipa-replica-install` script is run.

As with the server and client install scripts, any replica arguments which require a parameter value (such as the Directory Manager password) will be prompted for during installation, unless the argument is passed with the command. Parameters with Boolean values (like configuring DNS) will assume that the default value should be used unless the argument is passed with the command.

C.5.2.1. Location

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool directory</td>
<td>/usr/sbin</td>
</tr>
<tr>
<td>Package</td>
<td>ipa-server</td>
</tr>
</tbody>
</table>

C.5.2.2. Syntax

`ipa-replica-install [options] /path/to/replica_file`

C.5.2.3. Options

<table>
<thead>
<tr>
<th>Short Parameter</th>
<th>Long Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>--setup-ca</td>
<td>Sets up a Dogtag Certificate System certificate authority. Otherwise, no CA is configured for the replica.</td>
</tr>
<tr>
<td>-N</td>
<td>--no-ntp</td>
<td>Does not configure NTP on the replica system.</td>
</tr>
<tr>
<td>--no-sshd</td>
<td></td>
<td>Does not configure the OpenSSH server or client.</td>
</tr>
<tr>
<td>--ssh-trust-dns</td>
<td></td>
<td>Configures the OpenSSH client for the FreeIPA client to trust any domain DNS SSHFP records.</td>
</tr>
<tr>
<td>--no-ui-redirect</td>
<td></td>
<td>Configures the Apache server to not automatically redirect to the FreeIPA web UI, so that</td>
</tr>
<tr>
<td>Short Parameter</td>
<td>Long Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>the web UI has to be accessed through its <code>/ipa/ui</code> path.</td>
<td></td>
</tr>
<tr>
<td>-d</td>
<td>--debug</td>
<td>Prints additional debug information.</td>
</tr>
<tr>
<td>-p</td>
<td>--password</td>
<td>Gives the Directory Manager password for the FreeIPA domain.</td>
</tr>
<tr>
<td>-w</td>
<td>--admin-password</td>
<td>Gives the Kerberos password for the FreeIPA <code>admin</code> user. This is used to check Kerberos and domain connectivity on the replica.</td>
</tr>
<tr>
<td></td>
<td>--setup-dns</td>
<td>Sets up DNS services on the replica machine to connect to the FreeIPA DNS domain. If this is not used, then the default value is false, which does not enable DNS.</td>
</tr>
<tr>
<td></td>
<td>--forwarder</td>
<td>Gives a comma-separated list of IP addresses for DNS forwarders.</td>
</tr>
<tr>
<td></td>
<td>--no-forwarders</td>
<td>Disables DNS forwarder configuration and uses only domain root servers. If this is not used, then the default value is false, which prompts for DNS forwarder information.</td>
</tr>
<tr>
<td></td>
<td>--reverse-zone</td>
<td>Gives a DNS reverse zone to use with the replica DNS services.</td>
</tr>
<tr>
<td></td>
<td>--no-reverse</td>
<td>Disables reverse DNS creation through the server installation process. (If a reverse DNS zone is already configured, then that existing reverse DNS zone is used.) If this option is not used, then the default value is true, which assumes that reverse DNS should be configured by the installation script.</td>
</tr>
<tr>
<td></td>
<td>--no-host-dns</td>
<td>Disables host DNS lookups during the replica installation process. If this is not used, then the default value is true, which performs the host DNS lookups.</td>
</tr>
</tbody>
</table>
### Appendix C. FreeIPA Tools Reference

<table>
<thead>
<tr>
<th>Short Parameter</th>
<th>Long Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--no-dns-sshfp</td>
<td>Configures the client to not configure DNS SSHFP records.</td>
</tr>
<tr>
<td></td>
<td>--skip-conncheck</td>
<td>Disables checks for the replica’s connection to the FreeIPA domain. If this is not used, then the default value is true, which checks that the replica can connect to the Kerberos realm. This can be useful if the replica is unable to reach the Directory Server or the CA used by the original FreeIPA server, such as the server is offline or the server's firewall is blocking access on the required ports (Section 2.1.4.4, &quot;System Ports&quot;).</td>
</tr>
<tr>
<td>-U</td>
<td>--unattended</td>
<td>Disables user prompts so that the replica installation script runs without user interaction.</td>
</tr>
</tbody>
</table>

**C.5.3. ipa-replica-prepare**

Creates a file that can be used to create a copy, or replica, of the FreeIPA server.

Each replica initialization file is unique to the replica machine because the configuration is based, in part, on the IP address and hostname of the replica machine. This host-specific configuration is especially critical for setting up services like Kerberos which use SSL because SSL certificates are created based on the hostname.

When the replica file is created, the prep script requires the hostname and, optionally, accepts the IP address.

Once the configuration file is created on the server using the `ipa-replica-prepare` command, then the replica file is copied over to the replica machine and the replica is configured using the `ipa-replica-install` command.

- **NOTE**

  If DNS is managed by FreeIPA, then use either the `--ip-address` option or configure DNS forwarders and allow reverse DNS lookups.

**C.5.3.1. Location**

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool directory</td>
<td>/usr/sbin</td>
</tr>
<tr>
<td>Package</td>
<td>ipa-server</td>
</tr>
</tbody>
</table>
C.5.3.2. Syntax

ipa-replica-prepare [ --dirsrv_pkcs12=file ] [ --http_pkcs12=file ] [ --dirsrv_pin=pin ] [ --http_pin=pin ] [ --ip-address=ipAddress ] hostname

C.5.3.3. Options

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dirsrv_pkcs12</td>
<td>Gives the full path and filename of a PKCS #12 file (.p12) which contains the Directory Server's SSL certificate.</td>
</tr>
<tr>
<td>--dirsrv_pin</td>
<td>Gives the password to access the Directory Server certificate file.</td>
</tr>
<tr>
<td>--http_pkcs12</td>
<td>Gives the full path and filename of a PKCS #12 file (.p12) which contains the Apache server's SSL certificate.</td>
</tr>
<tr>
<td>--http_pin</td>
<td>Gives the password to access the Apache certificate file.</td>
</tr>
<tr>
<td>--ip-address</td>
<td>Gives the IP address of the replica server. Using this option automatically adds A and PTR records for the replica host to the FreeIPA DNS configuration.</td>
</tr>
</tbody>
</table>

C.5.4. ipa-server-install

Configures all of the services used by the FreeIPA server for the FreeIPA domain:

- Dogtag Certificate System, for issuing server certificates
- 389 Directory Server, for storing all of the FreeIPA information
- The Kerberos KDC, with the LDAP backend
- Apache, for the web-based services
- NTP
- The `ipa_kpasswd` service
- Optionally, DNS

This script can be run interactively, which prompts for many of the server values, or information can be passed directly to the script so that the server can be configured without human intervention.

The FreeIPA server configuration is very flexible. The setup script allows some customization to services like DNS, NTP, certificate issuance, and access control in FreeIPA so that the server can be suited to the network environment.

C.5.4.1. Location

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool directory</td>
<td>/usr/sbin</td>
</tr>
</tbody>
</table>
Appendix C. FreeIPA Tools Reference

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>ipa-server</td>
</tr>
</tbody>
</table>

C.5.4.2. Syntax

```
ipa-server-install -a ipa_admin_password --hostname=hostname -p directory_manager_password -n domain_name -r realm_name [[--external-ca][--external-ca-file=CA_cert_chain_file][--external_cert_file=certificate_file][--selfsign][--subject=subject_DN][--setup-dns][--forwarder=forwarder][--no-forwarders][--no-reverse][--zone-refresh=seconds][--zone-notif][--zonomgr=email_address][--ip-address=ip_address][-P kerberos_master_password][-no-ntp][-idmax=number][-idstart=number][-no_hbac_allow][-no-host-dns][-U][-install][-debug][-help][-version]
```

C.5.4.3. Options

<table>
<thead>
<tr>
<th>Argument</th>
<th>Alternate Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Options²</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-a ipa_admin_password</td>
<td>--admin-password=ipa_admin_password</td>
<td>The password for the FreeIPA administrator. This is used for the admin user to authenticate to the Kerberos realm.</td>
</tr>
<tr>
<td>--hostname=hostname</td>
<td></td>
<td>The fully-qualified domain name of the FreeIPA server machine.</td>
</tr>
<tr>
<td>-n domain_name</td>
<td>--domain=domain_name</td>
<td>The name of the LDAP server domain to use for the FreeIPA domain. This is usually based on the FreeIPA server's hostname.</td>
</tr>
</tbody>
</table>

**IMPORTANT**

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures. Additionally, the hostname must all be lower-case. No capital letters are allowed.
<table>
<thead>
<tr>
<th>Argument</th>
<th>Alternate Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-p</code></td>
<td><code>--ds-password</code></td>
<td>The password for the superuser, <code>cn=Directory Manager</code>, for the LDAP service.</td>
</tr>
<tr>
<td><code>directory_manager_password</code></td>
<td><code>--ds-password=directory_manager_password</code></td>
<td></td>
</tr>
<tr>
<td><code>-r</code></td>
<td><code>--realm</code></td>
<td>The name of the Kerberos realm to create for the FreeIPA domain.</td>
</tr>
<tr>
<td><code>realm_name</code></td>
<td><code>--realm=realm_name</code></td>
<td></td>
</tr>
<tr>
<td><strong>Certificate Authority Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--external-ca</code></td>
<td></td>
<td>Instructs the installation script to generate a certificate request that can be submitted to an external or third-party CA.</td>
</tr>
<tr>
<td><code>external_ca_file=CA_cert_chain_file</code></td>
<td></td>
<td>Points to the PKCS#10 file which contains the CA certificate chain of the external CA. This is required to validate the certificate issued by the CA for the FreeIPA server. If an external CA is used, this is required in a second invocation of <code>ipa-server-install</code> to complete the setup process.</td>
</tr>
<tr>
<td><code>external_cert_file=certificate_file</code></td>
<td></td>
<td>Points to the PKCS#10 file which contains the certificate that was generated by an external CA. If an external CA is used, this is required in a second invocation of <code>ipa-server-install</code> to complete the setup process.</td>
</tr>
<tr>
<td><code>--subject</code></td>
<td><code>subject_DN</code></td>
<td>Sets the base element for the subject DN of the issued certificates. This defaults to <code>O=realm</code>.</td>
</tr>
<tr>
<td><strong>DNS Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--forwarder</code></td>
<td><code>forwarder</code></td>
<td>Gives a comma-separated list of DNS forwarders to use with the DNS service.</td>
</tr>
<tr>
<td><code>no-forwarders</code></td>
<td></td>
<td>Uses root servers with the DNS service instead of forwarders.</td>
</tr>
<tr>
<td><code>--no-reverse</code></td>
<td></td>
<td>Disables reverse DNS creation through the server installation process. (If a reverse DNS zone is already configured, then that existing reverse DNS zone is used.) If this option is not used, then the default value is true, which assumes that reverse DNS should be</td>
</tr>
<tr>
<td>Argument</td>
<td>Alternate Argument</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>configured by the installation script.</td>
</tr>
<tr>
<td>--setup-dns</td>
<td></td>
<td>Tells the installation script to set up a DNS service within the FreeIPA domain. Using an integrated DNS service is optional, so if this option is not passed with the installation script, then no DNS is configured.</td>
</tr>
<tr>
<td>--zone-refresh=seconds</td>
<td></td>
<td>Sets whether the FreeIPA server should periodically check to see when new DNS zones are added and update its DNS server accordingly. The polling interval is set in seconds.</td>
</tr>
<tr>
<td>--zone-notif</td>
<td></td>
<td>Opens a persistent search with its Directory Server and captures any new zone changes immediately.</td>
</tr>
<tr>
<td>--zonemgr=email_address</td>
<td></td>
<td>Gives the email address to use for the DNS zone manager. If none is given, this defaults to root.</td>
</tr>
<tr>
<td><strong>Kerberos Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--ip-address=ip_address</td>
<td></td>
<td>Gives the IP address of the Kerberos master KDC. This can be used if there are multiple FreeIPA servers in the same realm.</td>
</tr>
<tr>
<td>-P kerberos_master_password</td>
<td>--master-password=kerberos_master_password</td>
<td>The password for the KDC. This is randomly generated if no value is given.</td>
</tr>
<tr>
<td><strong>NTP Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-N, --no-ntp</td>
<td></td>
<td>Does not configure the NTP service for the FreeIPA server. This is normally done by default.</td>
</tr>
<tr>
<td>Argument</td>
<td>Alternate Argument</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>NOTE</strong></td>
<td></td>
<td>If the FreeIPA server is running as a virtual guest, it should not run an NTP service.</td>
</tr>
<tr>
<td><strong>FreeIPA Server Configuration Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--idmax=number</code></td>
<td></td>
<td>Sets the upper bound for IDs which can be assigned by the FreeIPA server. The default value is the ID start value plus 199999.</td>
</tr>
<tr>
<td><code>--idstart=number</code></td>
<td></td>
<td>Sets the lower bound (starting value) for IDs which can be assigned by the FreeIPA server. The default value is randomly selected.</td>
</tr>
<tr>
<td><code>--no_hbac_allow</code></td>
<td></td>
<td>Disables the <code>allow_all</code> rule for host-based access control in the FreeIPA domain.</td>
</tr>
<tr>
<td><strong>Other Setup Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>--no-host-dns</code></td>
<td></td>
<td>Does not use DNS to look up the hostname of the FreeIPA server machine during the installation process.</td>
</tr>
<tr>
<td><code>-U</code></td>
<td><code>--unattended</code></td>
<td>Runs the <code>ipa-server-install</code> command without any interactive prompts.</td>
</tr>
<tr>
<td><code>--uninstall</code></td>
<td></td>
<td>Uninstalls an existing FreeIPA server.</td>
</tr>
<tr>
<td><strong>General Tool Options</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>-d</code></td>
<td><code>--debug</code></td>
<td>Runs the <code>ipa-server-install</code> command in debug mode and outputs debugging information.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td><code>--help</code></td>
<td>Prints the help information for the <code>ipa-server-install</code> command.</td>
</tr>
<tr>
<td><code>--version</code></td>
<td></td>
<td>Prints the version number of the <code>ipa-server-install</code> command.</td>
</tr>
</tbody>
</table>

1 The installation script will prompt for these options if they are not passed with the script.
C.6. Client Scripts

These tools are used to manage client machines.

C.6.1. ipa-client-install

Configures a client machine. This script uses the local SSSD service to connect to the FreeIPA server during the setup process. It is also possible to connect to the server through PAM/NSS using LDAP.

This script is also used to uninstall clients, which removes them from the FreeIPA domain and removes all FreeIPA-related configuration.

This script is only available for Fedora platforms.

C.6.1.1. Location

<table>
<thead>
<tr>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool directory</td>
<td>/usr/sbin/</td>
</tr>
<tr>
<td>Package</td>
<td>ipa-client</td>
</tr>
</tbody>
</table>

C.6.1.2. Syntax

```
```

C.6.1.3. Options

<table>
<thead>
<tr>
<th>Long Parameter</th>
<th>Short Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--domain=domainName</td>
<td></td>
<td>Gives the domain name for the FreeIPA domain.</td>
</tr>
<tr>
<td>--enable-dns-updates</td>
<td></td>
<td>Tells SSSD to update DNS with the IP address of this client.</td>
</tr>
<tr>
<td>--force</td>
<td>-f</td>
<td>Forces the script to apply the settings even if errors occur.</td>
</tr>
<tr>
<td>--hostname=clientHostname</td>
<td></td>
<td>Sets the fully-qualified domain name of the client server. If this is not given, the script uses the nodename given in uname.</td>
</tr>
<tr>
<td>Long Parameter</td>
<td>Short Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--mkhomedir</td>
<td></td>
<td>Configures PAM to create a user’s home directory if it does not exist.</td>
</tr>
<tr>
<td>--no-dns-sshfp</td>
<td></td>
<td>Configures the client to not configure DNS SSHFP records.</td>
</tr>
<tr>
<td>--no-ntp</td>
<td>-N</td>
<td>Does not configure or enable NTP.</td>
</tr>
<tr>
<td>--no-krb5-offline-passwords</td>
<td></td>
<td>Prevents the SSSD services from storing Kerberos passwords in the SSSD cache. The cache is useful because a user may log into a system when a machine is offline and then attempt to access domain services after the machine is brought online. Using the cache stores the password, which can be referenced when the domain is accessed.</td>
</tr>
<tr>
<td>--no-sshd</td>
<td></td>
<td>Does not configure the OpenSSH server or client.</td>
</tr>
<tr>
<td>--ntp-server=NTPD_server</td>
<td></td>
<td>Configures the local NTPd service to use the FreeIPA NTP server.</td>
</tr>
<tr>
<td>--on-master</td>
<td></td>
<td>Indicates the client is being configured on a FreeIPA server. This is not for a normal invocation of the setup script; this option is used by ipa-</td>
</tr>
</tbody>
</table>

**IMPORTANT**

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

Additionally, the hostname must all be lower-case. No capital letters are allowed.
## Appendix C. FreeIPA Tools Reference

<table>
<thead>
<tr>
<th>Long Parameter</th>
<th>Short Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>server-install</code></td>
<td>--principal</td>
<td>when a server is configured.</td>
</tr>
<tr>
<td><code>--principal</code></td>
<td>-p</td>
<td>Passes an authorized Kerberos principal to use to join the FreeIPA realm. This is used during an automated deployment, such as a kickstart process.</td>
</tr>
<tr>
<td><code>--permit</code></td>
<td></td>
<td>Configures SSSD to permit all access. If this is not set, then access to the client is controlled by the host-based access controls on the FreeIPA server.</td>
</tr>
<tr>
<td><code>--preserve-sssd</code></td>
<td></td>
<td>Preserves the original SSSD configuration rather than updating it for FreeIPA-specific changes, such as SSH key caching.</td>
</tr>
<tr>
<td><code>--realm=realmName</code></td>
<td></td>
<td>Gives the FreeIPA realm name.</td>
</tr>
<tr>
<td><code>--no-sssd</code></td>
<td>-S</td>
<td>Tells the client to use nss_ldap for authentication instead of SSSD.</td>
</tr>
<tr>
<td><code>--server=IPA_server_fqdn</code></td>
<td></td>
<td>Gives the name of the FreeIPA server to connect to. This must be a fully-qualified domain name.</td>
</tr>
<tr>
<td>--ssh-trust-dns</td>
<td></td>
<td>Configures the OpenSSH client for the FreeIPA client to trust any domain DNS SSHFP records.</td>
</tr>
</tbody>
</table>

**IMPORTANT**

This must be a valid DNS name, which means only numbers, alphabetic characters, and hyphens (-) are allowed. Other characters, like underscores, in the hostname will cause DNS failures.
<table>
<thead>
<tr>
<th>Long Parameter</th>
<th>Short Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--unattended</td>
<td>-U</td>
<td>Performs an unattended installation, with no user prompts.</td>
</tr>
<tr>
<td>--uninstall</td>
<td></td>
<td>Removes the FreeIPA client software and configuration to restore the machine to a pre-FreeIPA state.</td>
</tr>
<tr>
<td>--password=password</td>
<td>-w password</td>
<td>Gives the Kerberos password to use to access the FreeIPA realm and join the machine. If only the password parameter is used, the script assumes this is a bulk enrollment and uses the machine name as the Kerberos principal. If the principal is given, the script binds as an FreeIPA user.</td>
</tr>
<tr>
<td></td>
<td>-W</td>
<td>Prompts for the password.</td>
</tr>
</tbody>
</table>
Glossary

A

access control instruction  See ACI.

access control list  See ACL.

access rights  In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

account inactivation  Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

ACI  An instruction that grants or denies permissions to entries in the directory.

See Also access control instruction.

ACL  The mechanism for controlling access to your directory.

See Also access control list.

All IDs Threshold  Replaced with the ID list scan limit in Directory Server version 7.1. A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

See Also ID list scan limit.

All IDs token  A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access  When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index  Allows for efficient approximate or "sounds-like" searches.

attribute  Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list  A list of required and optional attributes for a given entry type or object class.

authenticating directory server  In pass-through authentication (PTA), the authenticating Directory Server is the Directory Server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.

authentication  (1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding
password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

<table>
<thead>
<tr>
<th>term</th>
<th>definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication certificate</td>
<td>Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.</td>
</tr>
<tr>
<td>base distinguished name</td>
<td>See base DN.</td>
</tr>
<tr>
<td>base DN</td>
<td>Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.</td>
</tr>
<tr>
<td>bind distinguished name</td>
<td>See bind DN.</td>
</tr>
<tr>
<td>bind DN</td>
<td>Distinguished name used to authenticate to Directory Server when performing an operation.</td>
</tr>
<tr>
<td>bind rule</td>
<td>In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.</td>
</tr>
<tr>
<td>branch entry</td>
<td>An entry that represents the top of a subtree in the directory.</td>
</tr>
<tr>
<td>browser</td>
<td>Software, such as Mozilla Firefox, used to request and view World Wide Web material stored as HTML files. The browser uses the HTTP protocol to communicate with the host server.</td>
</tr>
<tr>
<td>browsing index</td>
<td>Speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branch point in the directory tree to improve display performance. See Also virtual list view index.</td>
</tr>
<tr>
<td>CA</td>
<td>See Certificate Authority.</td>
</tr>
<tr>
<td>cascading replication</td>
<td>In a cascading replication scenario, one server, often called the hub supplier, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. It receives updates from the supplier server that holds the master copy of the data and in turn supplies those updates to the consumer.</td>
</tr>
<tr>
<td>certificate</td>
<td>A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in the directory as user object attributes.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Certificate Authority</td>
<td>Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a CA.</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts and can be written in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.</td>
</tr>
<tr>
<td>chaining</td>
<td>A method for relaying requests to another server. Results for the request are collected, compiled, and then returned to the client.</td>
</tr>
<tr>
<td>changelog</td>
<td>A changelog is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on replica servers or on other masters, in the case of multi-master replication.</td>
</tr>
<tr>
<td>character type</td>
<td>Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.</td>
</tr>
<tr>
<td>ciphertext</td>
<td>Encrypted information that cannot be read by anyone without the proper key to decrypt the information.</td>
</tr>
<tr>
<td>class definition</td>
<td>Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.</td>
</tr>
<tr>
<td>class of service</td>
<td>See CoS.</td>
</tr>
<tr>
<td>classic CoS</td>
<td>A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.</td>
</tr>
<tr>
<td>client</td>
<td>See LDAP client.</td>
</tr>
<tr>
<td>code page</td>
<td>An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.</td>
</tr>
<tr>
<td>collation order</td>
<td>Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.</td>
</tr>
<tr>
<td>consumer</td>
<td>Server containing replicated directory trees or subtrees from a supplier server.</td>
</tr>
<tr>
<td>consumer server</td>
<td>In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.</td>
</tr>
<tr>
<td>CoS</td>
<td>A method for sharing attributes between entries in a way that is invisible to applications.</td>
</tr>
<tr>
<td>CoS definition entry</td>
<td>Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.</td>
</tr>
<tr>
<td>CoS template entry</td>
<td>Contains a list of the shared attribute values.</td>
</tr>
</tbody>
</table>
Glossary

See Also template entry.

D

daemon A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.

DAP Directory Access Protocol. The ISO X.500 standard protocol that provides client access to the directory.

data master The server that is the master source of a particular piece of data.

database link An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.

default index One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.

definition entry See CoS definition entry.

Directory Access Protocol See DAP.

Directory Manager The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the Directory Manager.

directory service A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

directory tree The logical representation of the information stored in the directory. It mirrors the tree model used by most filesystems, with the tree’s root point appearing at the top of the hierarchy. Also known as DIT.

distinguished name String representation of an entry’s name and location in an LDAP directory.

DIT See directory tree.

DM See Directory Manager.

DN See distinguished name.

DNS Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with hostnames (such as www.example.com). Machines normally get the IP address for a hostname from a DNS server, or they look it up in tables maintained on their systems.

DNS alias A DNS alias is a hostname that the DNS server knows points to a different host—specifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as www.yourdomain.domain might point to a real machine called realthing.yourdomain.domain where the server currently exists.
entry A group of lines in the LDIF file that contains information about an object.

entry distribution Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index Allows you to search efficiently for entries containing a specific attribute value.

file extension The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename \texttt{index.html} the file extension is \texttt{html}.

file type The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter A constraint applied to a directory query that restricts the information returned.

filtered role Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

general access When granted, indicates that all authenticated users can access directory information.

GSS-API Generic Security Services. The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

hostname A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, \texttt{www.example.com} is the machine \texttt{www} in the subdomain \texttt{example} and \texttt{com} domain.

HTML Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Mozilla Firefox...
<table>
<thead>
<tr>
<th>Glossary</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.</td>
</tr>
<tr>
<td>HTTPD</td>
<td>An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.</td>
</tr>
<tr>
<td>hub</td>
<td>In the context of replication, a server that holds a replica that is copied from a different server, and, in turn, replicates it to a third server. See Also <em>cascading replication</em>.</td>
</tr>
<tr>
<td>ID list scan limit</td>
<td>A size limit which is globally applied to any indexed search operation. When the size of an individual ID list reaches this limit, the server replaces that ID list with an all IDs token.</td>
</tr>
<tr>
<td>index key</td>
<td>Each index that the directory uses is composed of a table of index keys and matching entry ID lists.</td>
</tr>
<tr>
<td>indirect CoS</td>
<td>An indirect CoS identifies the template entry using the value of one of the target entry's attributes.</td>
</tr>
<tr>
<td>international index</td>
<td>Speeds up searches for information in international directories.</td>
</tr>
<tr>
<td>International Standards Organization</td>
<td>See <em>ISO</em>.</td>
</tr>
<tr>
<td>IP address</td>
<td><em>Also Internet Protocol address</em>. A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10). Directory Server supports both IPv4 and IPv6 IP addresses.</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization.</td>
</tr>
<tr>
<td>knowledge reference</td>
<td>Pointers to directory information stored in different databases.</td>
</tr>
<tr>
<td>LDAP client</td>
<td>Software used to request and view LDAP entries from an LDAP Directory Server. See Also <em>browser</em>.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LDAP Data Interchange Format</td>
<td>See <a href="#">LDAP Data Interchange Format</a>.</td>
</tr>
<tr>
<td>LDAP URL</td>
<td>Provides the means of locating Directory Servers using DNS and then completing the query via LDAP. A sample LDAP URL is <code>ldap:// ldap.example.com</code>.</td>
</tr>
<tr>
<td>LDAPv3</td>
<td>Version 3 of the LDAP protocol, upon which Directory Server bases its schema format.</td>
</tr>
<tr>
<td>LDBM database</td>
<td>A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.</td>
</tr>
<tr>
<td>LDIF</td>
<td>LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.</td>
</tr>
<tr>
<td>leaf entry</td>
<td>An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.</td>
</tr>
<tr>
<td>Lightweight Directory Access Protocol</td>
<td>See <a href="#">LDAP</a>.</td>
</tr>
<tr>
<td>locale</td>
<td>Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.</td>
</tr>
<tr>
<td>managed object</td>
<td>A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.</td>
</tr>
<tr>
<td>managed role</td>
<td>Allows creation of an explicit enumerated list of members.</td>
</tr>
<tr>
<td>management information base</td>
<td>See <a href="#">MIB</a>.</td>
</tr>
<tr>
<td>mapping tree</td>
<td>A data structure that associates the names of suffixes (subtrees) with databases.</td>
</tr>
<tr>
<td>master</td>
<td>See <a href="#">supplier</a>.</td>
</tr>
<tr>
<td>master agent</td>
<td>See <a href="#">SNMP master agent</a>.</td>
</tr>
<tr>
<td>matching rule</td>
<td>Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.</td>
</tr>
<tr>
<td>MD5</td>
<td>A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data that is unique with high probability and is mathematically extremely hard to produce; a piece of data that will produce the same message digest.</td>
</tr>
</tbody>
</table>
Glossary

MD5 signature  
A message digest produced by the MD5 algorithm.

MIB  
Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree-like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace  
Management Information Base namespace. The means for directory data to be named and referenced. Also called the directory tree.

monetary format  
Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.

multi-master replication  
An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a changelog for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor  
The server containing the database link that communicates with the remote server.

N

n + 1 directory problem  
The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions  
Multiple entries with the same distinguished name.

nested role  
Allows the creation of roles that contain other roles.

network management application  
Network Management Station component that graphically displays information about SNMP managed devices, such as which device is up or down and which and how many error messages were received.

network management station  
See NMS.

NIS  
Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, filesystems, and network parameters throughout a network of computers.

NMS  
Powerful workstation with one or more network management applications installed. Also network management station.

ns-slapd  
Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server.

See Also slapd.
### Object Class

Object class Defines an entry type in the directory by defining which attributes are contained in the entry.

### Object Identifier

Object identifier A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

See Also **OID**.

### OID

OID See **object identifier**.

### Operational Attribute

Operational attribute Contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

### Parent Access

Parent access When granted, indicates that users have access to entries below their own in the directory tree if the bind DN is the parent of the targeted entry.

### Pass-Through Authentication

Pass-through authentication See **PTA**.

### Pass-Through Subtree

Pass-through subtree In pass-through authentication, the **PTA directory server** will pass through bind requests to the **authenticating directory server** from all clients whose DN is contained in this subtree.

### Password File

Password file A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as /etc/passwd because of where it is kept.

### Password Policy

Password policy A set of rules that governs how passwords are used in a given directory.

### PDU

PDU Encoded messages which form the basis of data exchanges between SNMP devices. Also **protocol data unit**.

### Permission

Permission In the context of access control, permission states whether access to the directory information is granted or denied and the level of access that is granted or denied.

See Also **access rights**.

### Pointer CoS

Pointer CoS A pointer CoS identifies the template entry using the template DN only.

### Presence Index

Presence index Allows searches for entries that contain a specific indexed attribute.

### Protocol

Protocol A set of rules that describes how devices on a network exchange information.

### Protocol Data Unit

Protocol data unit See **PDU**.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxy authentication</td>
<td>A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.</td>
</tr>
<tr>
<td>proxy DN</td>
<td>Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.</td>
</tr>
<tr>
<td>PTA</td>
<td>Mechanism by which one Directory Server consults another to check bind credentials. Also <em>pass-through authentication</em>.</td>
</tr>
<tr>
<td>PTA directory server</td>
<td>In pass-through authentication (<em>PTA</em>), the PTA Directory Server is the server that sends (passes through) bind requests it receives to the <em>authenticating directory server</em>.</td>
</tr>
<tr>
<td>PTA LDAP URL</td>
<td>In pass-through authentication, the URL that defines the <em>authenticating directory server</em>, pass-through subtree(s), and optional parameters.</td>
</tr>
<tr>
<td>RAM</td>
<td>Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.</td>
</tr>
<tr>
<td>rc.local</td>
<td>A file on Unix machines that describes programs that are run when the machine starts. It is also called <em>/etc/rc.local</em> because of its location.</td>
</tr>
<tr>
<td>RDN</td>
<td>The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name. Also <em>relative distinguished name</em>.</td>
</tr>
<tr>
<td>read-only replica</td>
<td>A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.</td>
</tr>
<tr>
<td>read-write replica</td>
<td>A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.</td>
</tr>
<tr>
<td>referential integrity</td>
<td>Mechanism that ensures that relationships between related entries are maintained within the directory.</td>
</tr>
<tr>
<td>referral</td>
<td>(1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.</td>
</tr>
<tr>
<td></td>
<td>(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.</td>
</tr>
<tr>
<td>relative distinguished name</td>
<td>See <em>RDN</em>.</td>
</tr>
<tr>
<td>replica</td>
<td>A database that participates in replication.</td>
</tr>
<tr>
<td>replica-initiated replication</td>
<td>Replication configuration where replica servers, either hub or consumer servers, pull directory data from supplier servers. This method is available only for legacy replication.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>replication</td>
<td>Act of copying directory trees or subtrees from supplier servers to replica servers.</td>
</tr>
<tr>
<td>replication agreement</td>
<td>Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the replica servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.</td>
</tr>
<tr>
<td>role</td>
<td>An entry grouping mechanism. Each role has members, which are the entries that possess the role.</td>
</tr>
<tr>
<td>role-based attributes</td>
<td>Attributes that appear on an entry because it possesses a particular role within an associated CoS template.</td>
</tr>
<tr>
<td>root</td>
<td>The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.</td>
</tr>
<tr>
<td>root suffix</td>
<td>The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.</td>
</tr>
<tr>
<td>SASL</td>
<td>An authentication framework for clients as they attempt to bind to a directory. Also Simple Authentication and Security Layer.</td>
</tr>
<tr>
<td>schema</td>
<td>Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.</td>
</tr>
<tr>
<td>schema checking</td>
<td>Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default, and users will receive an error if they try to save an entry that does not conform to the schema.</td>
</tr>
<tr>
<td>Secure Sockets Layer</td>
<td>See SSL.</td>
</tr>
<tr>
<td>self access</td>
<td>When granted, indicates that users have access to their own entries if the bind DN matches the targeted entry.</td>
</tr>
<tr>
<td>Server Console</td>
<td>Java-based application that allows you to perform administrative management of your Directory Server from a GUI.</td>
</tr>
<tr>
<td>server daemon</td>
<td>The server daemon is a process that, once running, listens for and accepts requests from clients.</td>
</tr>
<tr>
<td>Server Selector</td>
<td>Interface that allows you select and configure servers using a browser.</td>
</tr>
<tr>
<td>server service</td>
<td>A process on Windows that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.</td>
</tr>
</tbody>
</table>
service
A background process on a Windows machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE
Server Instance Entry. The ID assigned to an instance of Directory Server during installation.

Simple Authentication and Security Layer
See SASL.

Simple Network Management Protocol
See SNMP.

single-master replication
The most basic replication scenario in which multiple servers, up to four, each hold a copy of the same read-write replicas to replica servers. In a single-master replication scenario, the supplier server maintains a changelog.

SIR
See supplier-initiated replication.

slapd
LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication.
See Also ns-slapd.

SNMP
Used to monitor and manage application processes running on the servers by exchanging data about network activity. Also Simple Network Management Protocol.

SNMP master agent
Software that exchanges information between the various subagents and the NMS.

SNMP subagent
Software that gathers information about the managed device and passes the information to the master agent. Also called a subagent.

SOA
See start of authority (SOA).

SSL
A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP. Also called Secure Sockets Layer.

standard index
An index maintained by default.

start of authority (SOA)
A record which contains the core information about a DNS zone.

sub suffix
A branch underneath a root suffix.

subagent
See SNMP subagent.

substring index
Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.

suffix
The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

superuser
The most privileged user available on Unix machines. The superuser has complete access privileges to all files on the machine. Also called root.
supplier Server containing the master copy of directory trees or subtrees that are replicated to replica servers.

supplier server In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication Replication configuration where supplier servers replicate directory data to any replica servers.

symmetric encryption Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index Cannot be deleted or modified as it is essential to Directory Server operations.

T
target In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry The entries within the scope of a CoS.

TCP/IP Transmission Control Protocol/Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.

template entry See CoS template entry.

time/date format Indicates the customary formatting for times and dates in a specific region.

TLS The new standard for secure socket layers; a public key based protocol. Also Transport Layer Security.

topology The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security See TLS.

U
uid A unique number associated with each user on a Unix system.

URL Uniform Resource Locator. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is protocol://machine:port/document. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

V
virtual list view index Speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branch point in the directory tree to improve display performance.

See Also browsing index.
| **X** | **X.500 standard** | The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation. |
Index

A
access control
read permission, 340, 347, 352
Active Directory
schema differences between FreeIPA, 184, 187
attributes
setting multi-valued, 100

B
bind
DNS and LDAP, 201

C
client
troubleshooting
installation, 60
uninstalling, 61

D
DHCP, 142
DNS
adding zone records, 212
adding zones, 203
bind-dyndb-ldap and Directory Server, 201
disabling zones, 210
dynamic updates, 225
hosts with DHCP, 142
DNS zone records, 212
deleting, 218
format for adding, 215
IPv4 example, 217
IPv6 example, 217
PTR example, 218
SRV example, 217
types of records, 213

G
glue entries, 377

H
hosts
creating
with DHCP, 142
disabling, 153

I
installing clients
disabling OpenSSH, 159

K
Kerberos, 267
separate credentials cache, 74
SSSD password cache, 272
ticket policies, 268
global, 268
user-level, 270
troubleshooting Windows problems, 44

L
logging in
separate credentials cache, 74

N
naming conflicts
in replication, 376

P
password expiration, 261
password policies
expiration, 261
port forwarding
for the UI, 81
proxy servers
for the UI, 81

S
schema
differences between FreeIPA and Active
Directory, 184, 187
cn, 184
initials, 185
sn, 185
street and streetAddress, 184
services
disabling, 153
SSH
disabling at client install, 159
SSSD
and Kerberos passwords, 272
disabling cache, 272

ticket policies, 268
troubleshooting
client installation, 60
Kerberos on Windows, 44
Kerberos, unknown server error, 61
resolving hostnames on client, 61

U
uninstalling
Index

clients, 61
users
  multi-valued attributes, 100
  password expiration, 261
  separate credentials cache, 74

W
web UI
  port forwarding, 81
  proxy servers, 81
Windows
  troubleshooting Kerberos problems, 44
WinSync
  schema differences, 187

Z
zone records, 212
  deleting, 218
  format for adding, 215
  IPv4 example, 217
  IPv6 example, 217
  PTR example, 218
  SRV example, 217
  types, 213