



Red Hat

Red Hat Enterprise Linux 8

Managing IdM users, groups, hosts, and access control rules

Configuring users and hosts, managing them in groups, and controlling access via host-based (HBAC) and role-based access control (RBAC) rules

Red Hat Enterprise Linux 8 Managing IdM users, groups, hosts, and access control rules

Configuring users and hosts, managing them in groups, and controlling access via host-based (HBAC) and role-based access control (RBAC) rules

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This documentation collection provides instructions on creating users, groups, and hosts in Identity Management on Red Hat Enterprise Linux 8, and managing access to those hosts via HBAC and RBAC rules.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	9
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	10
CHAPTER 1. MANAGING USER ACCOUNTS USING THE COMMAND LINE	11
1.1. USER LIFE CYCLE	11
1.2. ADDING USERS USING THE COMMAND LINE	12
1.3. ACTIVATING USERS USING THE COMMAND LINE	13
1.4. PRESERVING USERS USING THE COMMAND LINE	14
1.5. DELETING USERS USING THE COMMAND LINE	14
1.6. RESTORING USERS USING THE COMMAND LINE	15
CHAPTER 2. MANAGING USER ACCOUNTS USING THE IDM WEB UI	17
2.1. USER LIFE CYCLE	17
2.2. ADDING USERS IN THE WEB UI	18
2.3. ACTIVATING STAGE USERS IN THE IDM WEB UI	20
2.4. DISABLING USER ACCOUNTS IN THE WEB UI	21
2.5. ENABLING USER ACCOUNTS IN THE WEB UI	22
2.6. PRESERVING ACTIVE USERS IN THE IDM WEB UI	23
2.7. RESTORING USERS IN THE IDM WEB UI	24
2.8. DELETING USERS IN THE IDM WEB UI	25
CHAPTER 3. MANAGING USER ACCOUNTS USING ANSIBLE PLAYBOOKS	27
3.1. USER LIFE CYCLE	27
3.2. ENSURING THE PRESENCE OF AN IDM USER USING AN ANSIBLE PLAYBOOK	28
3.3. ENSURING THE PRESENCE OF MULTIPLE IDM USERS USING ANSIBLE PLAYBOOKS	30
3.4. ENSURING THE PRESENCE OF MULTIPLE IDM USERS FROM A JSON FILE USING ANSIBLE PLAYBOOKS	31
3.5. ENSURING THE ABSENCE OF USERS USING ANSIBLE PLAYBOOKS	33
CHAPTER 4. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT	35
4.1. SUDO ACCESS ON AN IDM CLIENT	35
4.2. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE CLI	35
4.3. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE IDM WEB UI	37
4.4. CREATING A SUDO RULE ON THE CLI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT	40
4.5. CREATING A SUDO RULE IN THE IDM WEBUI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT	42
4.6. ENABLING GSSAPI AUTHENTICATION FOR SUDO ON AN IDM CLIENT	48
4.7. ENABLING GSSAPI AUTHENTICATION AND ENFORCING KERBEROS AUTHENTICATION INDICATORS FOR SUDO ON AN IDM CLIENT	50
4.8. SSSD OPTIONS CONTROLLING GSSAPI AUTHENTICATION FOR PAM SERVICES	52
4.9. TROUBLESHOOTING GSSAPI AUTHENTICATION FOR SUDO	54
4.10. USING AN ANSIBLE PLAYBOOK TO ENSURE SUDO ACCESS FOR AN IDM USER ON AN IDM CLIENT	55
CHAPTER 5. USING LDAPMODIFY TO MANAGE IDM USERS EXTERNALLY	58
5.1. TEMPLATES FOR MANAGING IDM USER ACCOUNTS EXTERNALLY	58
5.2. TEMPLATES FOR MANAGING IDM GROUP ACCOUNTS EXTERNALLY	60
5.3. PRESERVING AN IDM USER WITH LDAPMODIFY	61
CHAPTER 6. CONFIGURING IDM FOR EXTERNAL PROVISIONING OF USERS	64
6.1. PREPARING IDM ACCOUNTS FOR AUTOMATIC ACTIVATION OF STAGE USER ACCOUNTS	64
6.2. CONFIGURING AUTOMATIC ACTIVATION OF IDM STAGE USER ACCOUNTS	66
6.3. ADDING AN IDM STAGE USER DEFINED IN AN LDIF FILE	68

6.4. ADDING AN IDM STAGE USER DIRECTLY FROM THE CLI USING LDAPMODIFY	69
CHAPTER 7. MANAGING SELF-SERVICE RULES IN IDM USING THE CLI	72
7.1. SELF-SERVICE ACCESS CONTROL IN IDM	72
7.2. CREATING SELF-SERVICE RULES USING THE CLI	72
7.3. EDITING SELF-SERVICE RULES USING THE CLI	73
7.4. DELETING SELF-SERVICE RULES USING THE CLI	73
CHAPTER 8. MANAGING SELF-SERVICE RULES USING THE IDM WEB UI	75
8.1. SELF-SERVICE ACCESS CONTROL IN IDM	75
8.2. CREATING SELF-SERVICE RULES USING THE IDM WEB UI	75
8.3. EDITING SELF-SERVICE RULES USING THE IDM WEB UI	77
8.4. DELETING SELF-SERVICE RULES USING THE IDM WEB UI	78
CHAPTER 9. USING ANSIBLE PLAYBOOKS TO MANAGE SELF-SERVICE RULES IN IDM	79
9.1. SELF-SERVICE ACCESS CONTROL IN IDM	79
9.2. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS PRESENT	79
9.3. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS ABSENT	81
9.4. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE HAS SPECIFIC ATTRIBUTES	82
9.5. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE DOES NOT HAVE SPECIFIC ATTRIBUTES	84
CHAPTER 10. MANAGING USER GROUPS IN IDM CLI	86
10.1. THE DIFFERENT GROUP TYPES IN IDM	86
10.2. DIRECT AND INDIRECT GROUP MEMBERS	87
10.3. ADDING A USER GROUP USING IDM CLI	87
10.4. SEARCHING FOR USER GROUPS USING IDM CLI	88
10.5. DELETING A USER GROUP USING IDM CLI	88
10.6. ADDING A MEMBER TO A USER GROUP USING IDM CLI	89
10.7. ADDING USERS WITHOUT A USER PRIVATE GROUP	90
10.7.1. Users without a user private group	90
10.7.2. Adding a user without a user private group when private groups are globally enabled	90
10.7.3. Disabling user private groups globally for all users	91
10.7.4. Adding a user when user private groups are globally disabled	91
10.8. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE IDM CLI	92
10.9. VIEWING GROUP MEMBERS USING IDM CLI	93
10.10. REMOVING A MEMBER FROM A USER GROUP USING IDM CLI	94
10.11. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE IDM CLI	94
CHAPTER 11. MANAGING USER GROUPS IN IDM WEB UI	96
11.1. THE DIFFERENT GROUP TYPES IN IDM	96
11.2. DIRECT AND INDIRECT GROUP MEMBERS	97
11.3. ADDING A USER GROUP USING IDM WEB UI	97
11.4. DELETING A USER GROUP USING IDM WEB UI	98
11.5. ADDING A MEMBER TO A USER GROUP USING IDM WEB UI	99
11.6. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE WEB UI	100
11.7. VIEWING GROUP MEMBERS USING IDM WEB UI	102
11.8. REMOVING A MEMBER FROM A USER GROUP USING IDM WEB UI	102
11.9. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE WEB UI	103
CHAPTER 12. MANAGING USER GROUPS USING ANSIBLE PLAYBOOKS	105

12.1. THE DIFFERENT GROUP TYPES IN IDM	105
12.2. DIRECT AND INDIRECT GROUP MEMBERS	106
12.3. ENSURING THE PRESENCE OF IDM GROUPS AND GROUP MEMBERS USING ANSIBLE PLAYBOOKS	107
12.4. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS	108
12.5. ENSURING THE ABSENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS	109
CHAPTER 13. AUTOMATING GROUP MEMBERSHIP USING IDM CLI	112
13.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP	112
13.2. AUTOMEMBER RULES	112
13.3. ADDING AN AUTOMEMBER RULE USING IDM CLI	113
13.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM CLI	114
13.5. VIEWING EXISTING AUTOMEMBER RULES USING IDM CLI	115
13.6. DELETING AN AUTOMEMBER RULE USING IDM CLI	116
13.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM CLI	116
13.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM CLI	117
13.9. CONFIGURING A DEFAULT AUTOMEMBER GROUP USING IDM CLI	118
CHAPTER 14. AUTOMATING GROUP MEMBERSHIP USING IDM WEB UI	120
14.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP	120
14.2. AUTOMEMBER RULES	120
14.3. ADDING AN AUTOMEMBER RULE USING IDM WEB UI	121
14.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM WEB UI	122
14.5. VIEWING EXISTING AUTOMEMBER RULES AND CONDITIONS USING IDM WEB UI	123
14.6. DELETING AN AUTOMEMBER RULE USING IDM WEB UI	124
14.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM WEB UI	125
14.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM WEB UI	126
14.8.1. Rebuilding automatic membership for all users or hosts	126
14.8.2. Rebuilding automatic membership for a single user or host only	127
14.9. CONFIGURING A DEFAULT USER GROUP USING IDM WEB UI	128
14.10. CONFIGURING A DEFAULT HOST GROUP USING IDM WEB UI	128
CHAPTER 15. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM CLI	130
15.1. DELEGATION RULES	130
15.2. CREATING A DELEGATION RULE USING IDM CLI	130
15.3. VIEWING EXISTING DELEGATION RULES USING IDM CLI	131
15.4. MODIFYING A DELEGATION RULE USING IDM CLI	131
15.5. DELETING A DELEGATION RULE USING IDM CLI	132
CHAPTER 16. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM WEBUI	133
16.1. DELEGATION RULES	133
16.2. CREATING A DELEGATION RULE USING IDM WEBUI	133
16.3. VIEWING EXISTING DELEGATION RULES USING IDM WEBUI	135
16.4. MODIFYING A DELEGATION RULE USING IDM WEBUI	136
16.5. DELETING A DELEGATION RULE USING IDM WEBUI	137
CHAPTER 17. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING ANSIBLE PLAYBOOKS	139
17.1. DELEGATION RULES	139
17.2. CREATING AN ANSIBLE INVENTORY FILE FOR IDM	139
17.3. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS PRESENT	140
17.4. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS ABSENT	142
17.5. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE HAS SPECIFIC ATTRIBUTES	143

17.6. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE DOES NOT HAVE SPECIFIC ATTRIBUTES	145
CHAPTER 18. MANAGING ROLE-BASED ACCESS CONTROLS IN IDM USING THE CLI	147
18.1. ROLE-BASED ACCESS CONTROL IN IDM	147
18.1.1. Permissions in IdM	147
18.1.2. Default managed permissions	148
18.1.3. Privileges in IdM	149
18.1.4. Roles in IdM	150
18.1.5. Predefined roles in Identity Management	150
18.2. MANAGING IDM PERMISSIONS IN THE CLI	151
18.3. COMMAND OPTIONS FOR EXISTING PERMISSIONS	153
18.4. MANAGING IDM PRIVILEGES IN THE CLI	153
18.5. COMMAND OPTIONS FOR EXISTING PRIVILEGES	154
18.6. MANAGING IDM ROLES IN THE CLI	154
18.7. COMMAND OPTIONS FOR EXISTING ROLES	155
CHAPTER 19. MANAGING ROLE-BASED ACCESS CONTROLS USING THE IDM WEB UI	156
19.1. ROLE-BASED ACCESS CONTROL IN IDM	156
19.1.1. Permissions in IdM	156
19.1.2. Default managed permissions	157
19.1.3. Privileges in IdM	158
19.1.4. Roles in IdM	159
19.1.5. Predefined roles in Identity Management	159
19.2. MANAGING PERMISSIONS IN THE IDM WEB UI	160
19.3. MANAGING PRIVILEGES IN THE IDM WEBUI	164
19.4. MANAGING ROLES IN THE IDM WEB UI	167
CHAPTER 20. PREPARING YOUR ENVIRONMENT FOR MANAGING IDM USING ANSIBLE PLAYBOOKS	172
CHAPTER 21. USING ANSIBLE PLAYBOOKS TO MANAGE ROLE-BASED ACCESS CONTROL IN IDM ...	174
21.1. PERMISSIONS IN IDM	174
21.2. DEFAULT MANAGED PERMISSIONS	175
21.3. PRIVILEGES IN IDM	177
21.4. ROLES IN IDM	177
21.5. PREDEFINED ROLES IN IDENTITY MANAGEMENT	177
21.6. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE WITH PRIVILEGES IS PRESENT	178
21.7. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE IS ABSENT	180
21.8. USING ANSIBLE TO ENSURE THAT A GROUP OF USERS IS ASSIGNED TO AN IDM RBAC ROLE	181
21.9. USING ANSIBLE TO ENSURE THAT SPECIFIC USERS ARE NOT ASSIGNED TO AN IDM RBAC ROLE	183
21.10. USING ANSIBLE TO ENSURE A SERVICE IS A MEMBER OF AN IDM RBAC ROLE	184
21.11. USING ANSIBLE TO ENSURE A HOST IS A MEMBER OF AN IDM RBAC ROLE	186
21.12. USING ANSIBLE TO ENSURE A HOST GROUP IS A MEMBER OF AN IDM RBAC ROLE	187
CHAPTER 22. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PRIVILEGES	190
22.1. USING ANSIBLE TO ENSURE A CUSTOM IDM RBAC PRIVILEGE IS PRESENT	190
22.2. USING ANSIBLE TO ENSURE MEMBER PERMISSIONS ARE PRESENT IN A CUSTOM IDM RBAC PRIVILEGE	191
22.3. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE DOES NOT INCLUDE A PERMISSION	193
22.4. USING ANSIBLE TO RENAME A CUSTOM IDM RBAC PRIVILEGE	194
22.5. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE IS ABSENT	196
CHAPTER 23. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PERMISSIONS IN IDM	198
23.1. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS PRESENT	198
23.2. USING ANSIBLE TO ENSURE AN RBAC PERMISSION WITH AN ATTRIBUTE IS PRESENT	200

23.3. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS ABSENT	201
23.4. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS A MEMBER OF AN IDM RBAC PERMISSION	203
23.5. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS NOT A MEMBER OF AN IDM RBAC PERMISSION	204
23.6. USING ANSIBLE TO RENAME AN IDM RBAC PERMISSION	205
23.7. ADDITIONAL RESOURCES	207
CHAPTER 24. USING AN ID VIEW TO OVERRIDE A USER ATTRIBUTE VALUE ON AN IDM CLIENT	208
24.1. ID VIEWS	208
24.2. POTENTIAL NEGATIVE IMPACT OF ID VIEWS ON SSSD PERFORMANCE	209
24.3. ATTRIBUTES AN ID VIEW CAN OVERRIDE	209
24.4. GETTING HELP FOR ID VIEW COMMANDS	210
24.5. USING AN ID VIEW TO OVERRIDE THE LOGIN NAME OF AN IDM USER ON A SPECIFIC HOST	210
24.6. MODIFYING AN IDM ID VIEW	213
24.7. ADDING AN ID VIEW TO OVERRIDE AN IDM USER HOME DIRECTORY ON AN IDM CLIENT	214
24.8. APPLYING AN ID VIEW TO AN IDM HOST GROUP	216
CHAPTER 25. ADJUSTING ID RANGES MANUALLY	219
25.1. ID RANGES	219
25.2. AUTOMATIC ID RANGES ASSIGNMENT	219
25.3. ASSIGNING THE IDM ID RANGE MANUALLY DURING SERVER INSTALLATION	220
25.4. ADDING A NEW IDM ID RANGE	221
25.5. REMOVING AN ID RANGE AFTER REMOVING A TRUST TO AD	222
25.6. DISPLAYING CURRENTLY ASSIGNED DNA ID RANGES	223
25.7. AUTOMATIC DNA ID RANGE EXTENSION	223
25.8. MANUAL DNA ID RANGE ADJUSTMENT	224
25.9. ADJUSTING DNA ID RANGES MANUALLY	225
CHAPTER 26. MANAGING HOSTS IN IDM CLI	226
26.1. HOSTS IN IDM	226
26.2. HOST ENROLLMENT	227
26.2.1. User privileges required for host enrollment	227
User privileges for optionally manually creating a host entry in IdM LDAP	227
User privileges for joining the client to the IdM domain	227
26.2.2. Enrollment and authentication of IdM hosts and users: comparison	228
26.2.3. Alternative authentication options for IdM hosts	229
26.3. HOST OPERATIONS	229
26.4. HOST ENTRY IN IDM LDAP	231
26.4.1. Host entry configuration properties	232
26.5. ADDING IDM HOST ENTRIES FROM IDM CLI	233
26.6. DELETING HOST ENTRIES FROM IDM CLI	234
26.7. RE-ENROLLING AN IDENTITY MANAGEMENT CLIENT	234
26.7.1. Client re-enrollment in IdM	234
26.7.1.1. What happens during client re-enrollment	234
26.7.2. Re-enrolling a client by using user credentials: Interactive re-enrollment	235
26.7.3. Re-enrolling a client by using the client keytab: Non-interactive re-enrollment	235
26.7.4. Testing an Identity Management client after installation	236
26.8. RENAMING IDENTITY MANAGEMENT CLIENT SYSTEMS	236
26.8.1. Preparing an IdM client for its renaming	236
26.8.2. Uninstalling an Identity Management client	237
26.8.3. Renaming the host system	238
26.8.4. Re-installing an Identity Management client	238
26.8.5. Re-adding services, re-generating certificates, and re-adding host groups	238
26.9. DISABLING AND RE-ENABLING HOST ENTRIES	238
26.9.1. Disabling Hosts	238

26.9.2. Re-enabling Hosts	239
CHAPTER 27. ADDING HOST ENTRIES FROM IDM WEB UI	240
27.1. HOSTS IN IDM	240
27.2. HOST ENROLLMENT	240
27.2.1. User privileges required for host enrollment	241
User privileges for optionally manually creating a host entry in IdM LDAP	241
User privileges for joining the client to the IdM domain	241
27.2.2. Enrollment and authentication of IdM hosts and users: comparison	241
27.2.3. Alternative authentication options for IdM hosts	242
27.3. HOST ENTRY IN IDM LDAP	243
27.3.1. Host entry configuration properties	243
27.4. ADDING HOST ENTRIES FROM THE WEB UI	244
CHAPTER 28. MANAGING HOSTS USING ANSIBLE PLAYBOOKS	247
28.1. HOSTS IN IDM	247
28.2. HOST ENROLLMENT	248
28.2.1. User privileges required for host enrollment	248
User privileges for optionally manually creating a host entry in IdM LDAP	248
User privileges for joining the client to the IdM domain	248
28.2.2. Enrollment and authentication of IdM hosts and users: comparison	249
28.2.3. Alternative authentication options for IdM hosts	250
28.3. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH FQDN USING ANSIBLE PLAYBOOKS	250
28.4. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH DNS INFORMATION USING ANSIBLE PLAYBOOKS	252
28.5. ENSURING THE PRESENCE OF MULTIPLE IDM HOST ENTRIES WITH RANDOM PASSWORDS USING ANSIBLE PLAYBOOKS	254
28.6. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH MULTIPLE IP ADDRESSES USING ANSIBLE PLAYBOOKS	255
28.7. ENSURING THE ABSENCE OF AN IDM HOST ENTRY USING ANSIBLE PLAYBOOKS	257
CHAPTER 29. MANAGING HOST GROUPS USING THE IDM CLI	259
29.1. HOST GROUPS IN IDM	259
29.2. VIEWING IDM HOST GROUPS USING THE CLI	259
29.3. CREATING IDM HOST GROUPS USING THE CLI	260
29.4. DELETING IDM HOST GROUPS USING THE CLI	260
29.5. ADDING IDM HOST GROUP MEMBERS USING THE CLI	261
29.6. REMOVING IDM HOST GROUP MEMBERS USING THE CLI	262
29.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE CLI	263
29.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE CLI	264
CHAPTER 30. MANAGING HOST GROUPS USING THE IDM WEB UI	266
30.1. HOST GROUPS IN IDM	266
30.2. VIEWING HOST GROUPS IN THE IDM WEB UI	266
30.3. CREATING HOST GROUPS IN THE IDM WEB UI	267
30.4. DELETING HOST GROUPS IN THE IDM WEB UI	268
30.5. ADDING HOST GROUP MEMBERS IN THE IDM WEB UI	268
30.6. REMOVING HOST GROUP MEMBERS IN THE IDM WEB UI	269
30.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI	270
30.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI	271
CHAPTER 31. MANAGING HOST GROUPS USING ANSIBLE PLAYBOOKS	274
31.1. HOST GROUPS IN IDM	274
31.2. ENSURING THE PRESENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	274
31.3. ENSURING THE PRESENCE OF HOSTS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	276

31.4. NESTING IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	277
31.5. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	279
31.6. ENSURING THE ABSENCE OF HOSTS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	280
31.7. ENSURING THE ABSENCE OF NESTED HOST GROUPS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	282
31.8. ENSURING THE ABSENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	283
31.9. ENSURING THE ABSENCE OF MEMBER MANAGERS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS	284
CHAPTER 32. ENSURING THE PRESENCE OF HOST-BASED ACCESS CONTROL RULES IN IDM USING ANSIBLE PLAYBOOKS	287
32.1. HOST-BASED ACCESS CONTROL RULES IN IDM	287
32.2. ENSURING THE PRESENCE OF AN HBAC RULE IN IDM USING AN ANSIBLE PLAYBOOK	287
CHAPTER 33. CONFIGURING THE DOMAIN RESOLUTION ORDER TO RESOLVE SHORT AD USER NAMES ..	289
33.1. HOW DOMAIN RESOLUTION ORDER WORKS	289
33.2. SETTING THE GLOBAL DOMAIN RESOLUTION ORDER ON AN IDM SERVER	290
33.3. SETTING THE DOMAIN RESOLUTION ORDER FOR AN ID VIEW ON AN IDM SERVER	290
33.4. SETTING THE DOMAIN RESOLUTION ORDER IN SSSD ON AN IDM CLIENT	292
CHAPTER 34. ENABLING AUTHENTICATION USING AD USER PRINCIPAL NAMES IN IDM	294
34.1. USER PRINCIPAL NAMES IN AN AD FOREST TRUSTED BY IDM	294
34.2. ENSURING THAT AD UPNS ARE UP-TO-DATE IN IDM	294
34.3. GATHERING TROUBLESHOOTING DATA FOR AD UPN AUTHENTICATION ISSUES	295
CHAPTER 35. ENABLING AD USERS TO ADMINISTER IDM	297
35.1. ID OVERRIDES FOR AD USERS	297
35.2. USING ID OVERRIDES TO ENABLE AD USERS TO ADMINISTER IDM	297
35.3. MANAGING IDM CLI AS AN AD USER	298

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

In Identity Management, planned terminology replacements include:

- ***block list*** replaces *blacklist*
- ***allow list*** replaces *whitelist*
- ***secondary*** replaces *slave*
- The word *master* is being replaced with more precise language, depending on the context:
 - ***IdM server*** replaces *IdM master*
 - ***CA renewal server*** replaces *CA renewal master*
 - ***CRL publisher server*** replaces *CRL master*
 - ***multi-supplier*** replaces *multi-master*

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
 1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
 2. Use your mouse cursor to highlight the part of text that you want to comment on.
 3. Click the **Add Feedback** pop-up that appears below the highlighted text.
 4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
 1. Go to the [Bugzilla](#) website.
 2. As the Component, use **Documentation**.
 3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
 4. Click **Submit Bug**.

CHAPTER 1. MANAGING USER ACCOUNTS USING THE COMMAND LINE

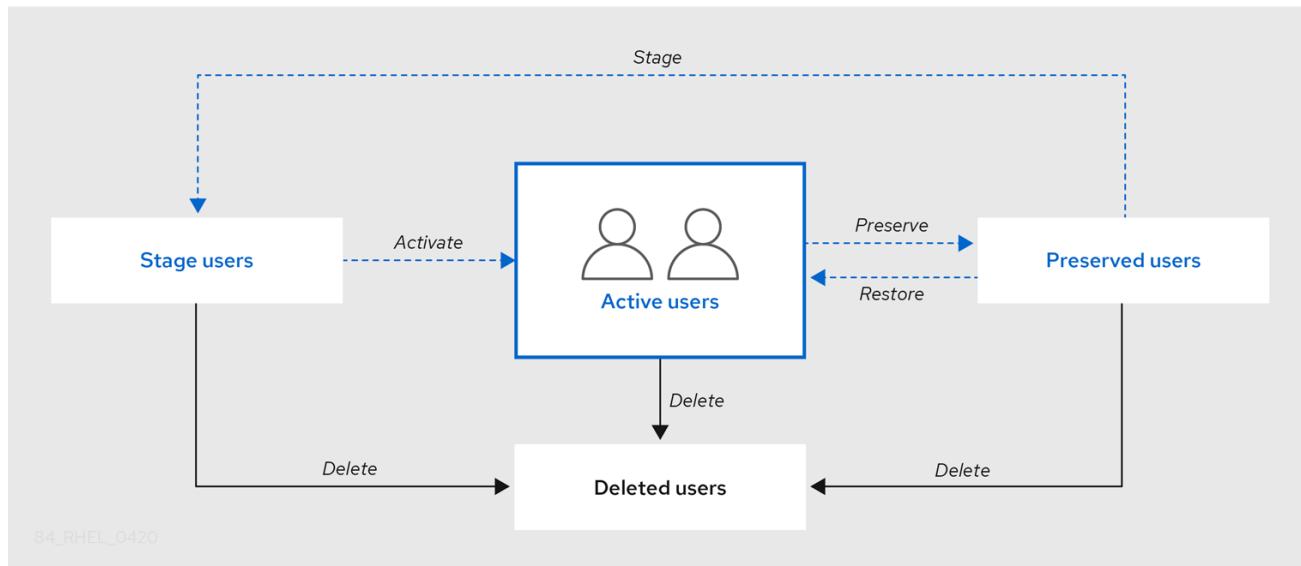
This chapter includes basic description of user life cycle in IdM (Identity Management). The following sections show you how to:

- Create user accounts
- Activate stage user accounts
- Preserve user accounts
- Delete active, stage, or preserved user accounts
- Restore preserved user accounts

1.1. USER LIFE CYCLE

Identity Management (IdM) supports three user account states:

- **Stage** users are not allowed to authenticate. This is an initial state. Some of the user account properties required for active users cannot be set, for example, group membership.
- **Active** users are allowed to authenticate. All required user account properties must be set in this state.
- **Preserved** users are former active users that are considered inactive and cannot authenticate to IdM. Preserved users retain most of the account properties they had as active users, but they are not part of any user groups.



You can delete user entries permanently from the IdM database.



IMPORTANT

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

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



WARNING

Do not delete the **admin** user. As **admin** is a pre-defined user required by IdM, this operation causes problems with certain commands. If you want to define and use an alternative admin user, disable the pre-defined **admin** user with **ipa user-disable admin** after you granted admin permissions to at least one different user.



WARNING

Do not add local users to IdM. The Name Service Switch (NSS) always resolves IdM users and groups before resolving local users and groups. This means that, for example, IdM group membership does not work for local users.

1.2. ADDING USERS USING THE COMMAND LINE

You can add user as:

- **Active** – user accounts which can be actively used by their users.
- **Stage** – users cannot use these accounts. Use it if you want to prepare new user accounts. When users are ready to use their accounts, then you can activate them.

The following procedure describes adding active users to the IdM server with the **ipa user-add** command.

Similarly, you can create stage user accounts with the **ipa stageuser-add** command.



NOTE

IdM automatically assigns a unique user ID (UID) to the new user accounts. You can also do this manually, however, the server does not validate whether the UID number is unique. Due to this, multiple user entries might have the same ID number assigned. Red Hat recommends to prevent having multiple entries with the same UID.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- Obtained a Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Open terminal and connect to the IdM server.
2. Add user login, user's first name, last name and optionally, you can also add their email address.

```
$ ipa user-add user_login --first=first_name --last=last_name --email=email_address
```

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

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



NOTE

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

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

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

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

The **ipa user-add** command includes a lot of parameters. To list them all, use the **ipa help** command:

```
$ ipa help user-add
```

For details about **ipa help** command, see [What is the IPA help](#).

You can verify if the new user account is successfully created by listing all IdM user accounts:

```
$ ipa user-find
```

This command lists all user accounts with details.

1.3. ACTIVATING USERS USING THE COMMAND LINE

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

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- Obtained a Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#) .

Procedure

1. Open terminal and connect to the IdM server.
2. Activate the user account with the following command:

```
$ ipa stageuser-activate user_login
```

```
-----  
Stage user user_login activated  
-----  
...
```

You can verify if the new user account is successfully created by listing all IdM user accounts:

```
$ ipa user-find
```

This command lists all user accounts with details.

1.4. PRESERVING USERS USING THE COMMAND LINE

You can preserve a user account if you want to remove it, but keep the option to restore it later. To preserve a user account, use the **--preserve** option with the **ipa user-del** or **ipa stageuser-del** commands.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- Obtained a Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Open terminal and connect to the IdM server.
2. Preserve the user account with the following command:

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

```
-----  
Deleted user "user_login"  
-----
```



NOTE

Despite the output saying the user account was deleted, it has been preserved.

1.5. DELETING USERS USING THE COMMAND LINE

IdM (Identity Management) enables you to delete users permanently. You can delete:

- Active users with the following command: **ipa user-del**
- Stage users with the following command: **ipa stageuser-del**
- Preserved users with the following command: **ipa user-del**

When deleting multiple users, use the **--continue** option to force the command to continue regardless of errors. A summary of the successful and failed operations is printed to the **stdout** standard output stream when the command completes.

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

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

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- Obtained a Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Open terminal and connect to the IdM server.
2. Delete the user account with the following command:

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

The user account has been permanently deleted from IdM.

1.6. RESTORING USERS USING THE COMMAND LINE

You can restore a preserved users to:

- Active users: **ipa user-undel**
- Stage users: **ipa user-stage**

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

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- Obtained a Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Open terminal and connect to the IdM server.
2. Activate the user account with the following command:

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

Alternatively, you can restore user accounts as staged:

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

Verification steps

- You can verify if the new user account is successfully created by listing all IdM user accounts:

```
$ ipa user-find
```

This command lists all user accounts with details.

CHAPTER 2. MANAGING USER ACCOUNTS USING THE IDM WEB UI

Identity Management (IdM) provides [several stages](#) that can help you to manage various user work life situations:

Creating a user account

[Creating a stage user account](#) before an employee starts their career in your company and be prepared in advance for the day when the employee appears in the office and want to activate the account.

You can omit this step and create the active user account directly. The procedure is similar to creating a stage user account.

Activating a user account

[Activating the account](#) the first working day of the employee.

Disabling a user account

If the user go to a parental leave for couple of months, you will need [to disable the account temporarily](#).

Enabling a user account

When the user returns, you will need [to re-enable the account](#).

Preserving a user account

If the user wants to leave the company, you will need [to delete the account with a possibility to restore it](#) because people can return to the company after some time.

Restoring a user account

Two years later, the user is back and you need [to restore the preserved account](#).

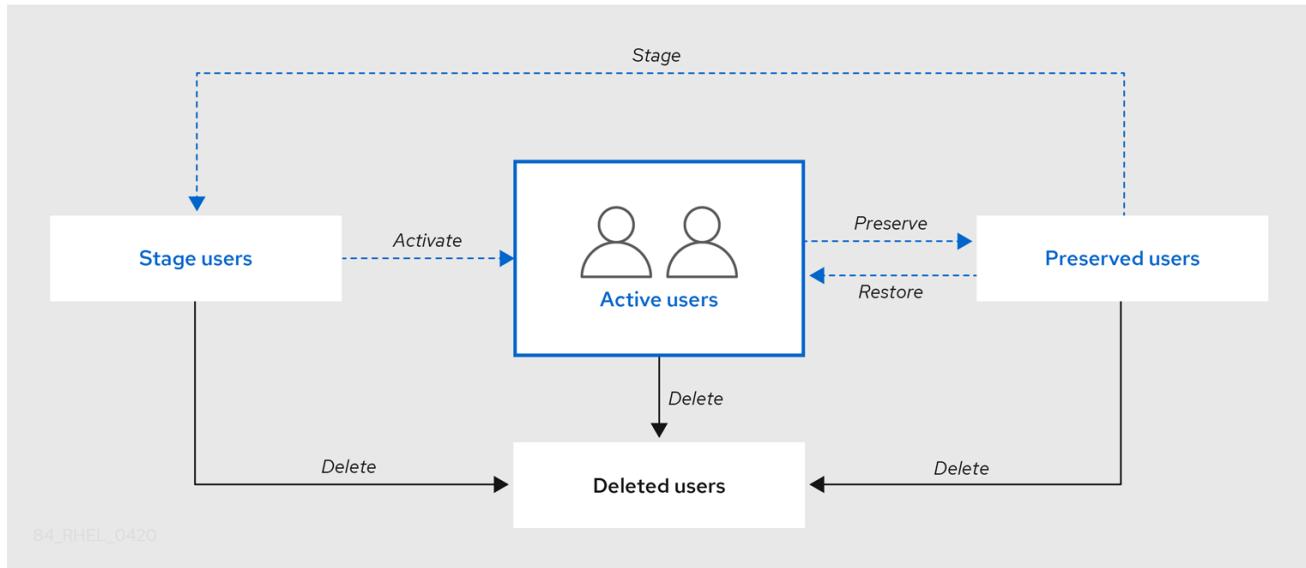
Deleting a user account

If the employee the employee is dismissed you will [delete the account](#) without a backup.

2.1. USER LIFE CYCLE

Identity Management (IdM) supports three user account states:

- **Stage** users are not allowed to authenticate. This is an initial state. Some of the user account properties required for active users cannot be set, for example, group membership.
- **Active** users are allowed to authenticate. All required user account properties must be set in this state.
- **Preserved** users are former active users that are considered inactive and cannot authenticate to IdM. Preserved users retain most of the account properties they had as active users, but they are not part of any user groups.



You can delete user entries permanently from the IdM database.



IMPORTANT

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

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



WARNING

Do not delete the **admin** user. As **admin** is a pre-defined user required by IdM, this operation causes problems with certain commands. If you want to define and use an alternative admin user, disable the pre-defined **admin** user with **ipa user-disable admin** after you granted admin permissions to at least one different user.

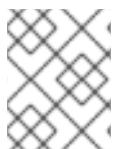


WARNING

Do not add local users to IdM. The Name Service Switch (NSS) always resolves IdM users and groups before resolving local users and groups. This means that, for example, IdM group membership does not work for local users.

2.2. ADDING USERS IN THE WEB UI

Usually, you need to create a new user account before a new employee starts to work. Such a stage account is not accessible and you need to activate it later.



NOTE

Alternatively, you can create an active user account directly. For adding active user, follow the procedure below and add the user account in the **Active users** tab.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.

Procedure

- Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#).
- Go to **Users → Stage Users** tab.
Alternatively, you can add the user account in the **Users → Active users**, however, you cannot add user groups to the account.
- Click the **+ Add** icon.
- In the **Add stage user** dialog box, enter **First name** and **Last name** of the new user.
- [Optional] In the **User login** field, add a login name.
If you leave it empty, the IdM server creates the login name in the following pattern: The first letter of the first name and the surname. The whole login name can have up to 32 characters.
- [Optional] In the **GID** drop down menu, select groups in which the user should be included.
- [Optional] In the **Password** and **Verify password** fields,
- Click on the **Add** button.

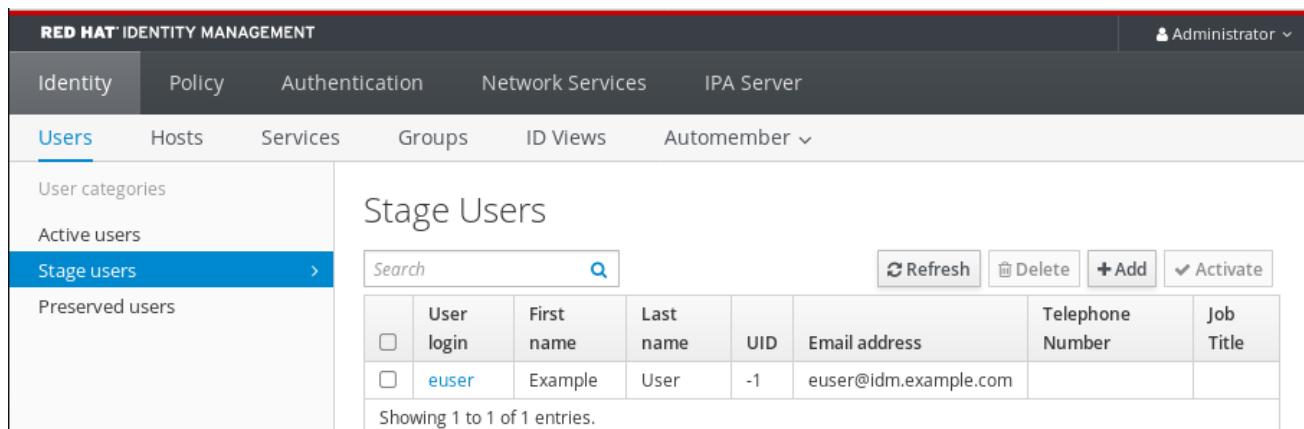
Add stage user

User login	<input type="text"/>
First name *	<input type="text" value="Example"/>
Last name *	<input type="text" value="User"/>
Class	<input type="text"/>
New Password	<input type="password" value="*****"/>
Verify Password	<input type="password" value="*****"/>

* Required field

Add **Add and Add Another** **Add and Edit** **Cancel**

At this point, you can see the user account in the **Stage Users** table.



The screenshot shows the Red Hat Identity Management web interface. The top navigation bar includes links for Identity, Policy, Authentication, Network Services, and IPA Server. The left sidebar has sections for User categories, Active users, Stage users (which is selected and highlighted in blue), and Preserved users. The main content area is titled "Stage Users". It features a search bar with a magnifying glass icon and buttons for Refresh, Delete, Add, and Activate. A table below lists one user entry:

	User login	First name	Last name	UID	Email address	Telephone Number	Job Title
<input type="checkbox"/>	euser	Example	User	-1	euser@idm.example.com		

Showing 1 to 1 of 1 entries.



NOTE

If you click on the user name, you can edit advanced settings, such as adding a phone number, address, or occupation.

2.3. ACTIVATING STAGE USERS IN THE IDM WEB UI

A stage user account must be activated before the user can log in to IdM and before the user can be added to an IdM group. This section describes how to activate stage user accounts.

Prerequisites

- Administrator privileges for managing the IdM Web UI or User Administrator role.
- At least one staged user account in IdM.

Procedure

1. Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#).
2. Go to **Users → Stage users** tab.
3. Click the check-box of the user account you want to activate.
4. Click on the **Activate** button.

User login	First name	Last name	UID	Email address	Telephone Number	Job Title
euser	Example	User	-1	euser@idm.example.com		

Showing 1 to 1 of 1 entries.

5. In the **Confirmation** dialog box, click on the **OK** button.

If the activation is successful, the IdM Web UI displays a green confirmation that the user has been activated and the user account has been moved to **Active users**. The account is active and the user can authenticate to the IdM domain and IdM Web UI. The user is prompted to change their password on the first login.

Active users

User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
admin		Administrator	<input checked="" type="checkbox"/> Enabled	78000000			
euser	Example	User	<input checked="" type="checkbox"/> Enabled	78000006	euser@idm.example.com		
staged.user	Staged	User	<input checked="" type="checkbox"/> Enabled	78000008	staged.user@idm.example.com		

Showing 1 to 3 of 3 entries.



NOTE

At this stage, you can add the active user account to user groups.

2.4. DISABLING USER ACCOUNTS IN THE WEB UI

You can disable active user accounts. Disabling a user account deactivates the account, therefore, user accounts cannot be used to authenticate and using IdM services, such as Kerberos, or perform any tasks.

Disabled user accounts still exist within IdM and all of the associated information remains unchanged. Unlike preserved user accounts, disabled user accounts remain in the active state and can be a member of user groups.



NOTE

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

Prerequisites

- Administrator privileges for managing the IdM Web UI or User Administrator role.

Procedure

- Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#).
- Go to **Users → Active users** tab.
- Click the check-box of the user accounts you want to disable.
- Click on the **Disable** button.

Active users								
Search				Actions				
	User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
<input type="checkbox"/>	admin		Administrator	<input checked="" type="checkbox"/> Enabled	78000000			
<input checked="" type="checkbox"/>	euser	Example	User	<input checked="" type="checkbox"/> Enabled	78000006	euser@idm.example.com		
<input type="checkbox"/>	preserved.user	Preserved	User	<input checked="" type="checkbox"/> Enabled	78000009	preserved.user@idm.example.com		

Showing 1 to 3 of 3 entries.

- In the **Confirmation** dialog box, click on the **OK** button.

If the disabling procedure has been successful, you can verify in the Status column in the **Active users** table.

	User login	First name	Last name	Status	UID	Email address	Telephone Number
<input type="checkbox"/>	admin		Administrator	<input checked="" type="checkbox"/> Enabled	78000000		
<input type="checkbox"/>	euser	Example	User	<input type="checkbox"/> Disabled	78000006	euser@idm.example.com	
<input type="checkbox"/>	preserved.user	Preserved	User	<input checked="" type="checkbox"/> Enabled	78000009	preserved.user@idm.example.com	

2.5. ENABLING USER ACCOUNTS IN THE WEB UI

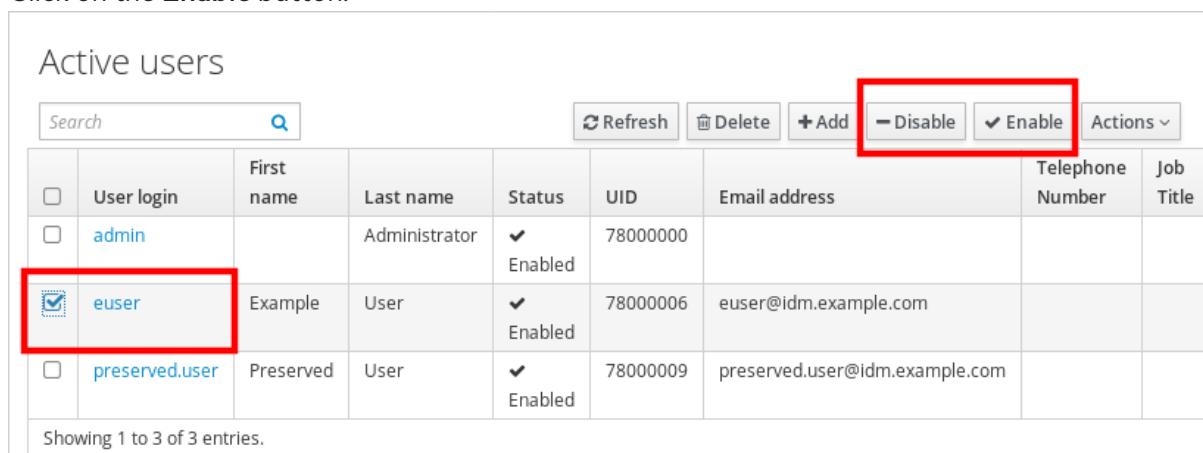
With IdM you can enable disabled active user accounts. Enabling a user account activates the disabled account.

Prerequisites

- Administrator privileges for managing the IdM Web UI or User Administrator role.

Procedure

- Log in to the IdM Web UI.
- Go to **Users → Active users** tab.
- Click the check-box of the user accounts you want to enable.
- Click on the **Enable** button.



Active users							Actions	
	User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
<input type="checkbox"/>	admin			✓ Enabled	78000000			
<input checked="" type="checkbox"/>	euser	Example	User	✓ Enabled	78000006	euser@idm.example.com		
<input type="checkbox"/>	preserved.user	Preserved	User	✓ Enabled	78000009	preserved.user@idm.example.com		

Showing 1 to 3 of 3 entries.

- In the **Confirmation** dialog box, click on the **OK** button.

If the change has been successful, you can verify in the Status column in the **Active users** table.

2.6. PRESERVING ACTIVE USERS IN THE IDM WEB UI

Preserving user accounts enables you to remove accounts from the **Active users** tab, yet keeping these accounts in IdM.

Preserve the user account if the employee leaves the company. If you want to disable user accounts for a couple of weeks or months (parental leave, for example), disable the account. For details, see [Section 2.4, "Disabling user accounts in the Web UI"](#). The preserved accounts are not active and users cannot use them to access your internal network, however, the account stays in the database with all the data.

You can move the restored accounts back to the active mode.



NOTE

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

Prerequisites

- Administrator privileges for managing the IdM (Identity Management) Web UI or User Administrator role.

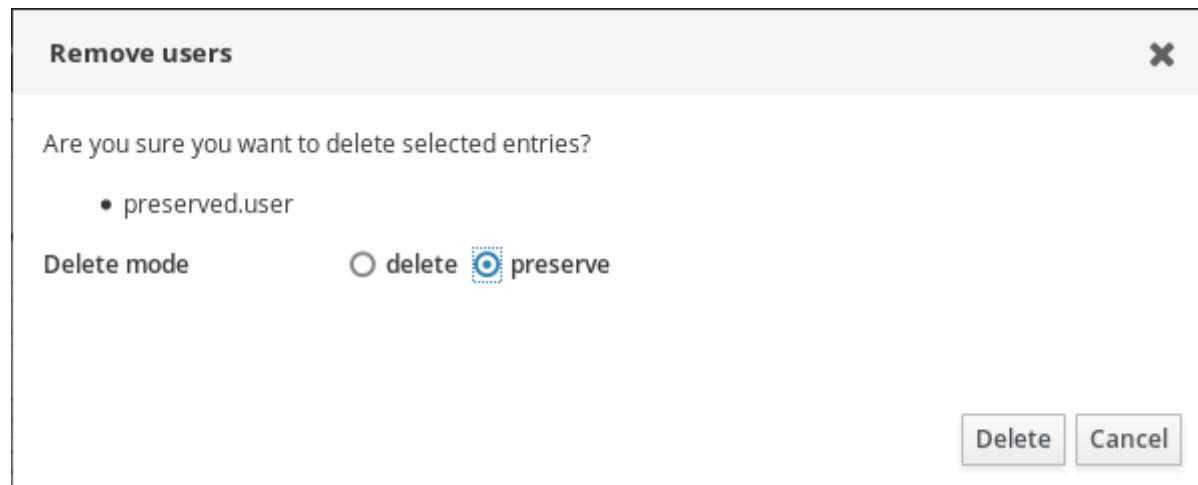
Procedure

1. Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#).
2. Go to **Users → Active users** tab.
3. Click the check-box of the user accounts you want to preserve.
4. Click on the **Delete** button.

User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
admin		Administrator	Enabled	78000000			
euser	Example	User	Enabled	78000006	euser@idm.example.com		
<input checked="" type="checkbox"/> preserved.user	Preserved	User	Enabled	78000009	preserved.user@idm.example.com		

Showing 1 to 3 of 3 entries.

5. In the **Remove users** dialog box, switch the **Delete mode** radio button to **preserve**.
6. Click on the **Delete** button.



As a result, the user account is moved to **Preserved users**.

If you need to restore preserved users, see the [Restoring users in the IdM Web UI](#).

2.7. RESTORING USERS IN THE IDM WEB UI

IdM (Identity Management) enables you to restore preserved user accounts back in the active state.

Prerequisites

- Administrator privileges for managing the IdM Web UI or User Administrator role.

Procedure

1. Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#).

2. Go to **Users → Preserved users** tab.
3. Click the check-box at the user accounts you want to restore.
4. Click on the **Restore** button.



Preserved users							
	User login	First name	Last name	UID	Email address	Telephone Number	Job Title
<input checked="" type="checkbox"/>	preserved.user	Preserved	User	78000009	preserved.user@idm.example.com		
Showing 1 to 1 of 1 entries.							

5. In the **Confirmation** dialog box, click on the **OK** button.

The IdM Web UI displays a green confirmation and moves the user accounts to the **Active users** tab.

2.8. DELETING USERS IN THE IDM WEB UI

Deleting users is an irreversible operation, causing the user accounts to be permanently deleted from the IdM database, including group memberships and passwords. Any external configuration for the user, such as the system account and home directory, is not deleted, but is no longer accessible through IdM.

You can delete:

- Active users – the IdM Web UI offers you with the options:
 - Preserving users temporarily
For details, see the [Preserving active users in the IdM Web UI](#) .
 - Deleting them permanently
- Stage users – you can just delete stage users permanently.
- Preserved users – you can delete preserved users permanently.

The following procedure describes deleting active users. Similarly, you can delete user accounts on:

- The **Stage users** tab
- The **Preserved users** tab

Prerequisites

- Administrator privileges for managing the IdM Web UI or User Administrator role.

Procedure

1. Log in to the IdM Web UI.
For details, see [Accessing the IdM Web UI in a web browser](#) .
2. Go to **Users → Active users** tab.
Alternatively, you can delete the user account in the **Users → Stage users** or **Users → Preserved users**.

3. Click the **Delete** icon.
4. In the **Remove users** dialog box, switch the **Delete mode** radio button to **delete**.
5. Click on the **Delete** button.

The users accounts have been permanently deleted from IdM.

CHAPTER 3. MANAGING USER ACCOUNTS USING ANSIBLE PLAYBOOKS

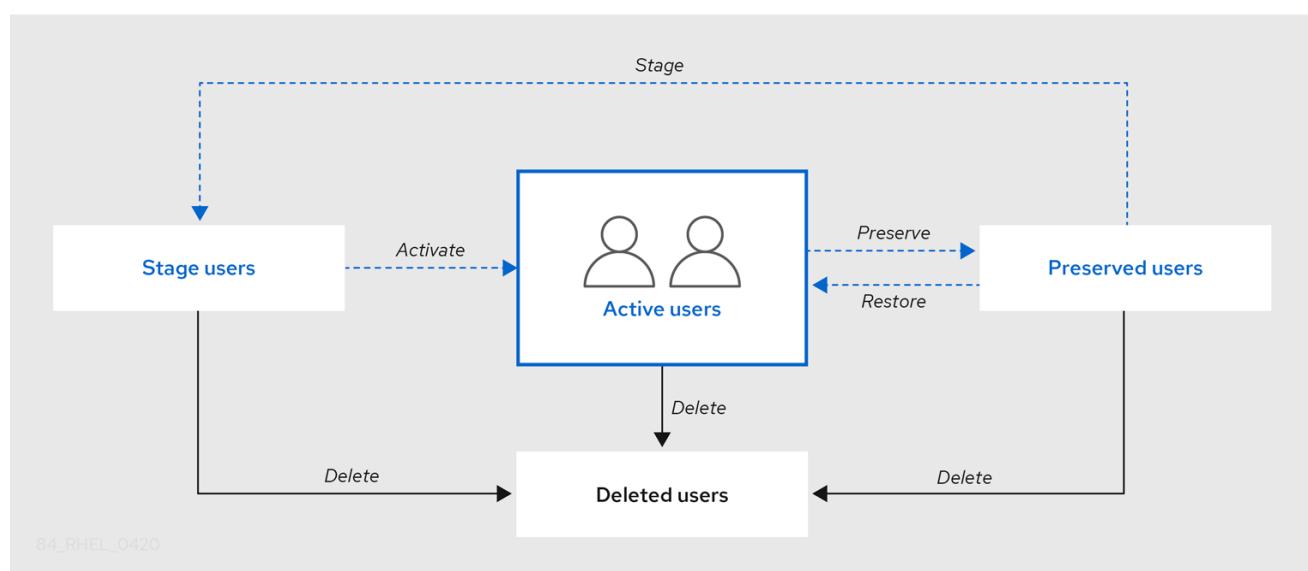
You can manage users in IdM using Ansible playbooks. After presenting the [user life cycle](#), this chapter describes how to use Ansible playbooks for the following operations:

- Ensuring the presence of a single [user](#) listed directly in the [YML](#) file.
- Ensuring the presence of multiple [users](#) listed directly in the [YML](#) file.
- Ensuring the presence of multiple [users](#) listed in a [JSON](#) file that is referenced from the [YML](#) file.
- Ensuring the absence of [users](#) listed directly in the [YML](#) file.

3.1. USER LIFE CYCLE

Identity Management (IdM) supports three user account states:

- **Stage** users are not allowed to authenticate. This is an initial state. Some of the user account properties required for active users cannot be set, for example, group membership.
- **Active** users are allowed to authenticate. All required user account properties must be set in this state.
- **Preserved** users are former active users that are considered inactive and cannot authenticate to IdM. Preserved users retain most of the account properties they had as active users, but they are not part of any user groups.



You can delete user entries permanently from the IdM database.



IMPORTANT

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

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



WARNING

Do not delete the **admin** user. As **admin** is a pre-defined user required by IdM, this operation causes problems with certain commands. If you want to define and use an alternative admin user, disable the pre-defined **admin** user with **ipa user-disable admin** after you granted admin permissions to at least one different user.



WARNING

Do not add local users to IdM. The Name Service Switch (NSS) always resolves IdM users and groups before resolving local users and groups. This means that, for example, IdM group membership does not work for local users.

3.2. ENSURING THE PRESENCE OF AN IDM USER USING AN ANSIBLE PLAYBOOK

The following procedure describes ensuring the presence of a user in IdM using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- The [ansible-freeipa](#) package is installed on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the data of the user whose presence in IdM you want to ensure. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/user/add-user.yml** file. For example, to create user named *idm_user* and add *Password123* as the user password:

```
---
- name: Playbook to handle users
  hosts: ipaserver
  become: true
```

tasks:

```
- name: Create user idm_user
  ipauser:
    ipaadmin_password: MySecret123
    name: idm_user
    first: Alice
    last: Acme
    uid: 1000111
    gid: 10011
    phone: "+555123457"
    email: idm_user@acme.com
    passwordexpiration: "2023-01-19 23:59:59"
    password: "Password123"
    update_password: on_create
```

You must use the following options to add a user:

- **name**: the login name
- **first**: the first name string
- **last**: the last name string

For the full list of available user options, see the [/usr/share/doc/ansible-freeipa/README-user.md](#) Markdown file.



NOTE

If you use the **update_password: on_create** option, Ansible only creates the user password when it creates the user. If the user is already created with a password, Ansible does not generate a new password.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/add-IdM-user.yml
```

Verification steps

- You can verify if the new user account exists in IdM by using the **ipa user-show** command:

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

3. Request information about *idm_user*:

```
$ ipa user-show idm_user
User login: idm_user
First name: Alice
Last name: Acme
....
```

The user named *idm_user* is present in IdM.

3.3. ENSURING THE PRESENCE OF MULTIPLE IDM USERS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the presence of multiple users in IdM using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the data of the users whose presence you want to ensure in IdM. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/user/ensure-users-present.yml** file. For example, to create users *idm_user_1*, *idm_user_2*, and *idm_user_3*, and add *Password123* as the password of *idm_user_1*:

```
---
- name: Playbook to handle users
  hosts: ipaserver
  become: true

  tasks:
    - name: Create user idm_users
      ipauser:
        ipaadmin_password: MySecret123
        users:
          - name: idm_user_1
            first: Alice
            last: Acme
            uid: 10001
            gid: 10011
            phone: "+555123457"
            email: idm_user@acme.com
            password_expiration: "2023-01-19 23:59:59"
            password: "Password123"
          - name: idm_user_2
            first: Bob
            last: Acme
```

```

uid: 100011
gid: 10011
- name: idm_user_3
  first: Eve
  last: Acme
  uid: 100011
  gid: 10011

```

**NOTE**

If you do not specify the **update_password: on_create** option, Ansible re-sets the user password every time the playbook is run: if the user has changed the password since the last time the playbook was run, Ansible re-sets password.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/add-users.yml
```

Verification steps

- You can verify if the user account exists in IdM by using the **ipa user-show** command:
 1. Log into **ipaserver** as administrator:

```
$ ssh administrator@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about *idm_user_1*:

```
$ ipa user-show idm_user_1
User login: idm_user_1
First name: Alice
Last name: Acme
Password: True
....
```

The user named *idm_user_1* is present in IdM.

3.4. ENSURING THE PRESENCE OF MULTIPLE IDM USERS FROM A JSON FILE USING ANSIBLE PLAYBOOKS

The following procedure describes how you can ensure the presence of multiple users in IdM using an Ansible playbook. The users are stored in a **JSON** file.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary tasks. Reference the **JSON** file with the data of the users whose presence you want to ensure. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/ensure-users-present-ymlfile.yml** file:

```
---
- name: Ensure users' presence
  hosts: ipaserver
  become: true

  tasks:
    - name: Include users.json
      include_vars:
        file: users.json

    - name: Users present
      ipauser:
        ipaadmin_password: MySecret123
        users: "{{ users }}"
```

3. Create the **users.json** file, and add the IdM users into it. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/user/users.json** file. For example, to create users *idm_user_1*, *idm_user_2*, and *idm_user_3*, and add *Password123* as the password of *idm_user_1*:

```
{
  "users": [
    {
      "name": "idm_user_1",
      "first": "Alice",
      "last": "Acme",
      "password": "Password123"
    },
    {
      "name": "idm_user_2",
      "first": "Bob",
      "last": "Acme"
    },
    {
      "name": "idm_user_3",
      "first": "Eve",
      "last": "Acme"
    }
  ]
}
```

4. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-users-present-jsonfile.yml
```

Verification steps

- You can verify if the user accounts are present in IdM using the **ipa user-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh administrator@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about *idm_user_1*:

```
$ ipa user-show idm_user_1
User login: idm_user_1
First name: Alice
Last name: Acme
Password: True
....
```

The user named *idm_user_1* is present in IdM.

3.5. ENSURING THE ABSENCE OF USERS USING ANSIBLE PLAYBOOKS

The following procedure describes how you can use an Ansible playbook to ensure that specific users are absent from IdM.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the users whose absence from IdM you want to ensure. To simplify this step, you can copy and modify the example in the [/usr/share/doc/ansible-freeipa/playbooks/user/ensure-users-present.yml](#) file. For example, to delete users *idm_user_1*, *idm_user_2*, and *idm_user_3*:

```
---
- name: Playbook to handle users
  hosts: ipaserver
  become: true
```

tasks:

```
- name: Delete users idm_user_1, idm_user_2, idm_user_3
  ipauser:
    ipaadmin_password: MySecret123
  users:
    - name: idm_user_1
    - name: idm_user_2
    - name: idm_user_3
  state: absent
```

3. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/delete-users.yml
```

Verification steps

You can verify that the user accounts do not exist in IdM by using the **ipa user-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh administrator@server.idm.example.com
Password:
[admin@server /]$
```

2. Request information about *idm_user_1*:

```
$ ipa user-show idm_user_1
ipa: ERROR: idm_user_1: user not found
```

The user named *idm_user_1* does not exist in IdM.

Additional resources

- You can see sample Ansible playbooks for other IdM user-related actions such as preserving, deleting, enabling, disabling, unlocking and undeleting users in the README-user.md Markdown file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of **ipauser** variables.
- You can also see sample Ansible playbooks in the **/usr/share/doc/ansible-freeipa/playbooks/user** directory.

CHAPTER 4. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT

4.1. SUDO ACCESS ON AN IDM CLIENT

System administrators can grant **sudo** access to allow non-root users to execute administrative commands that are normally reserved for the **root** user. Consequently, when users need to perform an administrative command normally reserved for the **root** user, they precede that command with **sudo**. After entering their password, the command is executed as if they were the **root** user. To execute a **sudo** command as another user or group, such as a database service account, you can configure a *RunAs alias* for a **sudo** rule.

If a Red Hat Enterprise Linux (RHEL) 8 host is enrolled as an Identity Management (IdM) client, you can specify **sudo** rules defining which IdM users can perform which commands on the host in the following ways:

- Locally in the **/etc/sudoers** file
- Centrally in IdM

This section describes creating a **central sudo rule** for an IdM client using the command line interface (CLI) and the IdM Web UI.

In RHEL 8.4 and later, you can also configure password-less authentication for **sudo** using the Generic Security Service Application Programming Interface (GSSAPI), the native way for UNIX-based operating systems to access and authenticate Kerberos services. You can use the **pam_sss_gss.so** Pluggable Authentication Module (PAM) to invoke GSSAPI authentication via the SSSD service, allowing users to authenticate to the **sudo** command with a valid Kerberos ticket.

Additional resources

- For details on creating local **sudo** rules on a RHEL host, see [Managing sudo access](#).

4.2. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE CLI

In Identity Management (IdM), you can grant **sudo** access for a specific command to an IdM user account on a specific IdM host. First, add a **sudo** command and then create a **sudo** rule for one or more commands.

For example, complete this procedure to create the **idm_user_reboot sudo** rule to grant the **idm_user** account the permission to run the **/usr/sbin/reboot** command on the **idmclient** machine.

Prerequisites

- You are logged in as IdM administrator.
- You have created a user account for **idm_user** in IdM and unlocked the account by creating a password for the user. For details on adding a new IdM user using the CLI, see [Adding users using the command line](#).
- No local **idm_user** account is present on the **idmclient** host. The **idm_user** user is not listed in the local **/etc/passwd** file.

Procedure

1. Retrieve a Kerberos ticket as the IdM **admin**.

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

2. Add the **/usr/sbin/reboot** command to the IdM database of **sudo** commands:

```
[root@idmclient ~]# ipa sudocmd-add /usr/sbin/reboot
-----
Added Sudo Command "/usr/sbin/reboot"
-----
Sudo Command: /usr/sbin/reboot
```

3. Create a **sudo** rule named **idm_user_reboot**:

```
[root@idmclient ~]# ipa sudorule-add idm_user_reboot
-----
Added Sudo Rule "idm_user_reboot"
-----
Rule name: idm_user_reboot
Enabled: TRUE
```

4. Add the **/usr/sbin/reboot** command to the **idm_user_reboot** rule:

```
[root@idmclient ~]# ipa sudorule-add-allow-command idm_user_reboot --sudocmds
'/usr/sbin/reboot'
Rule name: idm_user_reboot
Enabled: TRUE
Sudo Allow Commands: /usr/sbin/reboot
-----
Number of members added 1
```

5. Apply the **idm_user_reboot** rule to the IdM **idmclient** host:

```
[root@idmclient ~]# ipa sudorule-add-host idm_user_reboot --hosts
idmclient.idm.example.com
Rule name: idm_user_reboot
Enabled: TRUE
Hosts: idmclient.idm.example.com
Sudo Allow Commands: /usr/sbin/reboot
-----
Number of members added 1
```

6. Add the **idm_user** account to the **idm_user_reboot** rule:

```
[root@idmclient ~]# ipa sudorule-add-user idm_user_reboot --users idm_user
Rule name: idm_user_reboot
Enabled: TRUE
Users: idm_user
Hosts: idmclient.idm.example.com
Sudo Allow Commands: /usr/sbin/reboot
```

Number of members added 1



NOTE

Propagating the changes from the server to the client can take a few minutes.

Verification steps

1. Log in to the **idmclient** host as the **idm_user** account.
2. Display which **sudo** rules the **idm_user** account is allowed to perform.

```
[idm_user@idmclient ~]$ sudo -l
Matching Defaults entries for idm_user on idmclient:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY
KRB5CCNAME",
    secure_path=/sbin:/bin:/usr/sbin:/usr/bin
```

User **idm_user** may run the following commands on **idmclient**:
(root) /usr/sbin/reboot

3. Reboot the machine using **sudo**. Enter the password for **idm_user** when prompted:

```
[idm_user@idmclient ~]$ sudo /usr/sbin/reboot
[sudo] password for idm_user:
```

4.3. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE IDM WEB UI

In Identity Management (IdM), you can grant **sudo** access for a specific command to an IdM user account on a specific IdM host. First, add a **sudo** command and then create a **sudo** rule for one or more commands.

Complete this procedure to create the **idm_user_reboot** sudo rule to grant the **idm_user** account the permission to run the **/usr/sbin/reboot** command on the **idmclient** machine.

Prerequisites

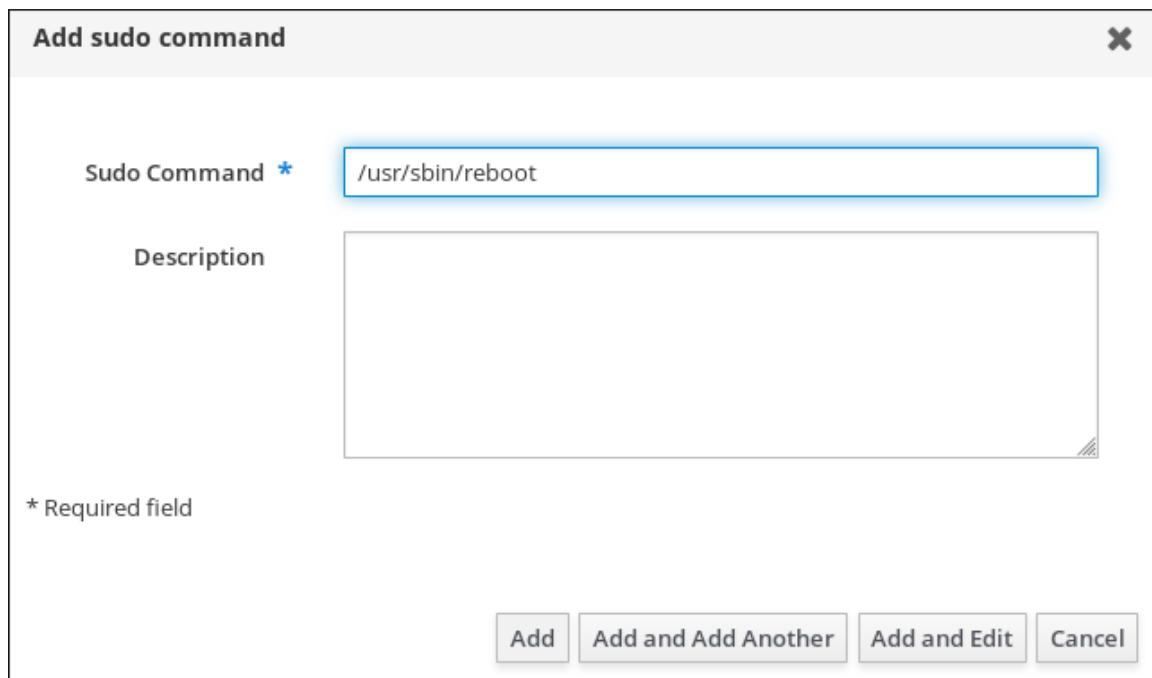
- You are logged in as IdM administrator.
- You have created a user account for **idm_user** in IdM and unlocked the account by creating a password for the user. For details on adding a new IdM user using the command-line interface, see [Adding users using the command line](#).

- No local **idm_user** account is present on the **idmclient** host. The **idm_user** user is not listed in the local **/etc/passwd** file.

Procedure

- Add the **/usr/sbin/reboot** command to the IdM database of **sudo** commands:
 - Navigate to **Policy → Sudo → Sudo Commands**.
 - Click **Add** in the upper right corner to open the **Add sudo command** dialog box.
 - Enter the command you want the user to be able to perform using **sudo: /usr/sbin/reboot**.

Figure 4.1. Adding IdM sudo command



- Click **Add**.
- Use the new **sudo** command entry to create a sudo rule to allow **idm_user** to reboot the **idmclient** machine:
 - Navigate to **Policy → Sudo → Sudo rules**.
 - Click **Add** in the upper right corner to open the **Add sudo rule** dialog box.
 - Enter the name of the **sudo** rule: **idm_user_reboot**.
 - Click **Add and Edit**
 - Specify the user:
 - In the **Who** section, check the **Specified Users and Groups** radio button.
 - In the **User category** the rule applies to subsection, click **Add** to open the **Add users into sudo rule "idm_user_reboot"** dialog box.
 - In the **Add users into sudo rule "idm_user_reboot"** dialog box in the **Available** column, check the **idm_user** checkbox, and move it to the **Prospective** column.

- iv. Click **Add**.
- f. Specify the host:
 - i. In the **Access this host** section, check the **Specified Hosts and Groups** radio button.
 - ii. In the **Host category this rule applies to** subsection, click **Add** to open the **Add hosts into sudo rule "idm_user_reboot"** dialog box.
 - iii. In the **Add hosts into sudo rule "idm_user_reboot"** dialog box in the **Available** column, check the **idmclient.idm.example.com** checkbox, and move it to the **Prospective** column.
 - iv. Click **Add**.
- g. Specify the commands:
 - i. In the **Command category the rule applies to** subsection of the **Run Commands** section, check the **Specified Commands and Groups** radio button.
 - ii. In the **Sudo Allow Commands** subsection, click **Add** to open the **Add allow sudo commands into sudo rule "idm_user_reboot"** dialog box.
 - iii. In the **Add allow sudo commands into sudo rule "idm_user_reboot"** dialog box in the **Available** column, check the **/usr/sbin/reboot** checkbox, and move it to the **Prospective** column.
 - iv. Click **Add** to return to the **idm_sudo_reboot** page.

Adding IdM sudo rule

+ image::IdM-sudo-rule-WebUI.png[A screenshot of an overview of the sudo rule that was added. There is a "Who" section with a table of users the rule applies to. There is an "Access this host" section with a table of hosts that the rule applies to. There is a "Run Commands" section with a table of commands that pertain to the rule.]

- a. Click **Save** in the top left corner.

The new rule is enabled by default.



NOTE

Propagating the changes from the server to the client can take a few minutes.

Verification steps

1. Log in to **idmclient** as **idm_user**.
2. Reboot the machine using **sudo**. Enter the password for **idm_user** when prompted:

```
$ sudo /usr/sbin/reboot
[sudo] password for idm_user:
```

If the **sudo** rule is configured correctly, the machine reboots.

4.4. CREATING A SUDO RULE ON THE CLI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT

In IdM, you can configure a **sudo** rule with a *RunAs alias* to run a **sudo** command as another user or group. For example, you might have an IdM client that hosts a database application, and you need to run commands as the local service account that corresponds to that application.

Use this example to create a **sudo** rule on the command line called **run_third-party-app_report** to allow the **idm_user** account to run the **/opt/third-party-app/bin/report** command as the **thirdpartyapp** service account on the **idmclient** host.

Prerequisites

- You are logged in as IdM administrator.
- You have created a user account for **idm_user** in IdM and unlocked the account by creating a password for the user. For details on adding a new IdM user using the CLI, see [Adding users using the command line](#).
- No local **idm_user** account is present on the **idmclient** host. The **idm_user** user is not listed in the local **/etc/passwd** file.
- You have a custom application named **third-party-app** installed on the **idmclient** host.
- The **report** command for the **third-party-app** application is installed in the **/opt/third-party-app/bin/report** directory.
- You have created a local service account named **thirdpartyapp** to execute commands for the **third-party-app** application.

Procedure

1. Retrieve a Kerberos ticket as the IdM **admin**.

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

2. Add the **/opt/third-party-app/bin/report** command to the IdM database of **sudo** commands:

```
[root@idmclient ~]# ipa sudocmd-add /opt/third-party-app/bin/report
-----
Added Sudo Command "/opt/third-party-app/bin/report"
-----
Sudo Command: /opt/third-party-app/bin/report
```

3. Create a **sudo** rule named **run_third-party-app_report**:

```
[root@idmclient ~]# ipa sudorule-add run_third-party-app_report
-----
Added Sudo Rule "run_third-party-app_report"
-----
Rule name: run_third-party-app_report
Enabled: TRUE
```

4. Use the **--users=<user>** option to specify the RunAs user for the **sudorule-add-runasuser** command:

```
[root@idmclient ~]# ipa sudorule-add-runasuser run_third-party-app_report --  
users=thirdpartyapp  
Rule name: run_third-party-app_report  
Enabled: TRUE  
RunAs External User: thirdpartyapp  
-----  
Number of members added 1  
-----
```

The user (or group specified with the **--groups=*** option) can be external to IdM, such as a local service account or an Active Directory user. Do not add a % prefix for group names.

5. Add the **/opt/third-party-app/bin/report** command to the **idm_user_reboot** rule:

```
[root@idmclient ~]# ipa sudorule-add-allow-command run_third-party-app_report --  
sudocmds '/opt/third-party-app/bin/report'  
Rule name: run_third-party-app_report  
Enabled: TRUE  
Sudo Allow Commands: /opt/third-party-app/bin/report  
RunAs External User: thirdpartyapp  
-----  
Number of members added 1  
-----
```

6. Apply the **run_third-party-app_report** rule to the IdM **idmclient** host:

```
[root@idmclient ~]# ipa sudorule-add-host run_third-party-app_report --hosts  
idmclient.idm.example.com  
Rule name: run_third-party-app_report  
Enabled: TRUE  
Hosts: idmclient.idm.example.com  
Sudo Allow Commands: /opt/third-party-app/bin/report  
RunAs External User: thirdpartyapp  
-----  
Number of members added 1  
-----
```

7. Add the **idm_user** account to the **run_third-party-app_report** rule:

```
[root@idmclient ~]# ipa sudorule-add-user run_third-party-app_report --users idm_user  
Rule name: run_third-party-app_report  
Enabled: TRUE  
Users: idm_user  
Hosts: idmclient.idm.example.com  
Sudo Allow Commands: /opt/third-party-app/bin/report  
RunAs External User: thirdpartyapp  
-----  
Number of members added 1  
-----
```



NOTE

Propagating the changes from the server to the client can take a few minutes.

Verification steps

1. Log in to the **idmclient** host as the **idm_user** account.
2. Test the new sudo rule:
 - a. Display which **sudo** rules the **idm_user** account is allowed to perform.

```
[idm_user@idmclient ~]$ sudo -l
Matching Defaults entries for idm_user@idm.example.com on idmclient:
  !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
  env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
LS_COLORS",
  env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
  env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES",
  env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER
LC_TELEPHONE",
  env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
XAUTHORITY KRB5CCNAME",
  secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin
```

User idm_user@idm.example.com may run the following commands on idmclient:
(thirdpartyapp) /opt/third-party-app/bin/report

- b. Run the **report** command as the **thirdpartyapp** service account.

```
[idm_user@idmclient ~]$ sudo -u thirdpartyapp /opt/third-party-app/bin/report
[sudo] password for idm_user@idm.example.com:
Executing report...
Report successful.
```

4.5. CREATING A SUDO RULE IN THE IDM WEBUI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT

In IdM, you can configure a **sudo** rule with a *RunAs alias* to run a **sudo** command as another user or group. For example, you might have an IdM client that hosts a database application, and you need to run commands as the local service account that corresponds to that application.

Use this example to create a **sudo** rule in the IdM WebUI called **run_third-party-app_report** to allow the **idm_user** account to run the **/opt/third-party-app/bin/report** command as the **thirdpartyapp** service account on the **idmclient** host.

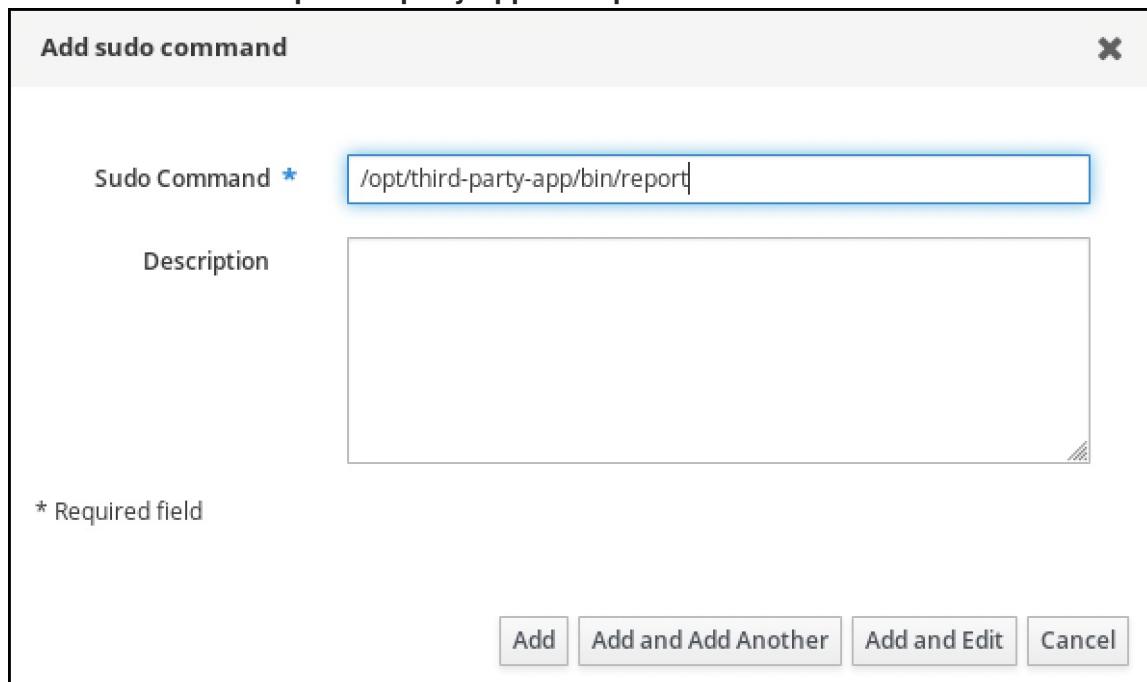
Prerequisites

- You are logged in as IdM administrator.
- You have created a user account for **idm_user** in IdM and unlocked the account by creating a password for the user. For details on adding a new IdM user using the CLI, see [Adding users using the command line](#).

- No local **idm_user** account is present on the **idmclient** host. The **idm_user** user is not listed in the local **/etc/passwd** file.
- You have a custom application named **third-party-app** installed on the **idmclient** host.
- The **report** command for the **third-party-app** application is installed in the **/opt/third-party-app/bin/report** directory.
- You have created a local service account named **thirdpartyapp** to execute commands for the **third-party-app** application.

Procedure

1. Add the **/opt/third-party-app/bin/report** command to the IdM database of **sudo** commands:
 - a. Navigate to **Policy → Sudo → Sudo Commands**.
 - b. Click **Add** in the upper right corner to open the **Add sudo command** dialog box.
 - c. Enter the command: **/opt/third-party-app/bin/report**.



2. Use the new **sudo** command entry to create the new **sudo** rule:
 - a. Navigate to **Policy → Sudo → Sudo rules**.
 - b. Click **Add** in the upper right corner to open the **Add sudo rule** dialog box.
 - c. Enter the name of the **sudo** rule: **run_third-party-app_report**.

Add sudo rule

Rule name *	run_third-party-app_report
-------------	----------------------------

* Required field

Add **Add and Add Another** **Add and Edit** **Cancel**

d. Click **Add and Edit**

e. Specify the user:

- In the **Who** section, check the **Specified Users and Groups** radio button.
- In the **User category** the rule applies to subsection, click **Add** to open the **Add users into sudo rule "run_third-party-app_report"** dialog box.
- In the **Add users into sudo rule "run_third-party-app_report"** dialog box in the **Available** column, check the **idm_user** checkbox, and move it to the **Prospective** column.

Add users into sudo rule 'run_third-party-app_report'

Available		Prospective	
<input type="checkbox"/>	Users	<input type="checkbox"/>	Users
<input type="checkbox"/>	admin	<input checked="" type="checkbox"/>	idmuser

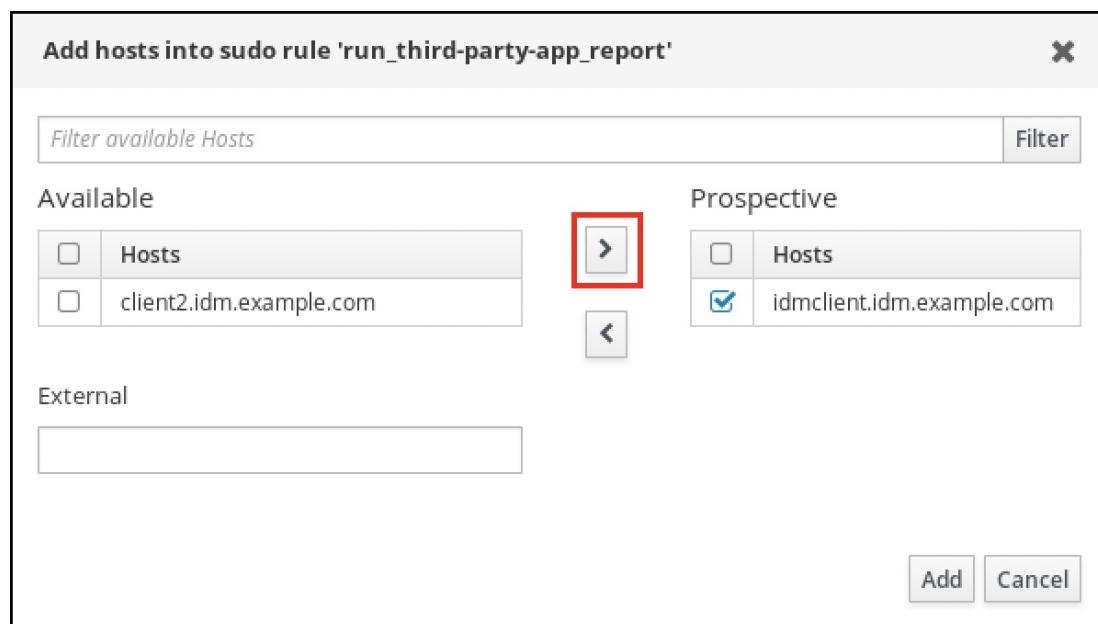
External

Add **Cancel**

iv. Click **Add**.

f. Specify the host:

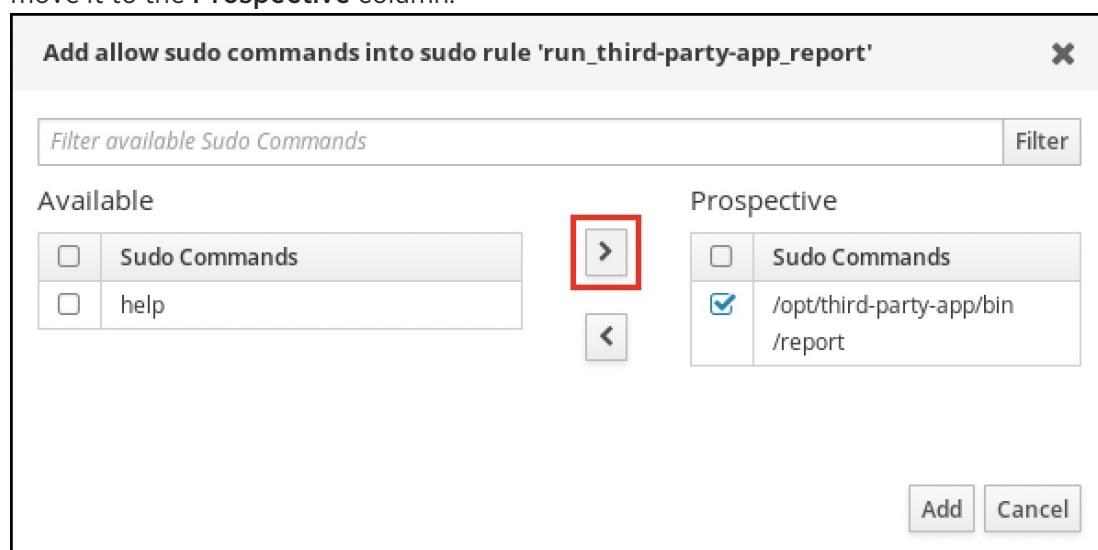
- In the **Access this host** section, check the **Specified Hosts and Groups** radio button.
- In the **Host category** this rule applies to subsection, click **Add** to open the **Add hosts into sudo rule "run_third-party-app_report"** dialog box.
- In the **Add hosts into sudo rule "run_third-party-app_report"** dialog box in the **Available** column, check the **idmclient.idm.example.com** checkbox, and move it to the **Prospective** column.



iv. Click **Add**.

g. Specify the commands:

- i. In the **Command category** the rule applies to subsection of the **Run Commands** section, check the **Specified Commands and Groups** radio button.
- ii. In the **Sudo Allow Commands** subsection, click **Add** to open the **Add allow sudo commands into sudo rule "run_third-party-app_report"** dialog box.
- iii. In the **Add allow sudo commands into sudo rule "run_third-party-app_report"** dialog box in the **Available** column, check the **/opt/third-party-app/bin/report** checkbox, and move it to the **Prospective** column.

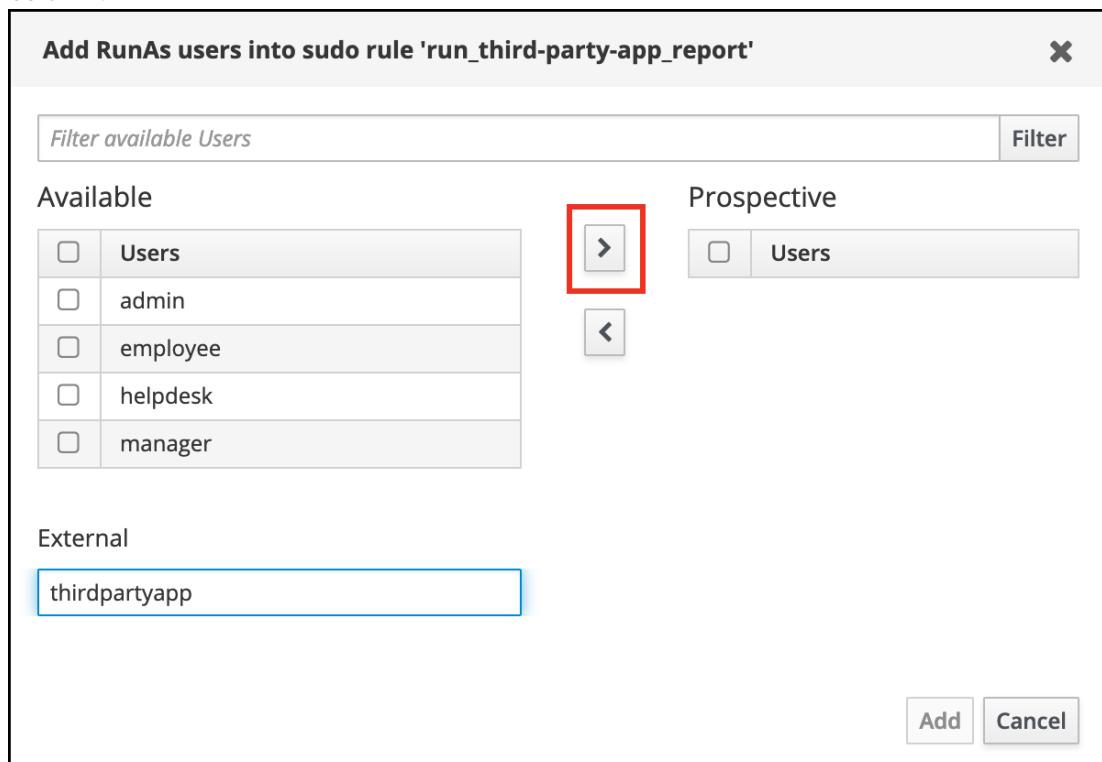


iv. Click **Add** to return to the **run_third-party-app_report** page.

h. Specify the RunAs user:

- i. In the **As Whom** section, check the **Specified Users and Groups** radio button.
- ii. In the **RunAs Users** subsection, click **Add** to open the **Add RunAs users into sudo rule "run_third-party-app_report"** dialog box.

- iii. In the **Add RunAs users into sudo rule "run_third-party-app_report"** dialog box, enter the **thirdpartyapp** service account in the **External** box and move it to the **Prospective** column.



- iv. Click **Add** to return to the **run_third-party-app_report** page.

- i. Click **Save** in the top left corner.

The new rule is enabled by default.

Figure 4.2. Details of the sudo rule

Who

User category the rule applies to: Anyone Specified Users and Groups

<input type="checkbox"/> Users	External	Delete + Add
<input type="checkbox"/> idm_user		

<input type="checkbox"/> User Groups		Delete + Add
--------------------------------------	--	----------------------------------------------

Access this host

Host category the rule applies to: Any Host Specified Hosts and Groups

<input type="checkbox"/> Hosts	External	Delete + Add
<input type="checkbox"/> idmclient.idm.example.com		

<input type="checkbox"/> Host Groups		Delete + Add
--------------------------------------	--	----------------------------------------------

Run Commands

Command category the rule applies to: Any Command Specified Commands and Groups

Allow

<input type="checkbox"/> Sudo Allow Commands		Delete + Add
<input type="checkbox"/> /opt/third-party-app/bin/report		

<input type="checkbox"/> Sudo Allow Command Groups		Delete + Add
----------------------------------------------------	--	----------------------------------------------

Deny

<input type="checkbox"/> Sudo Deny Commands		Delete + Add
---------------------------------------------	--	----------------------------------------------

<input type="checkbox"/> Sudo Deny Command Groups		Delete + Add
---------------------------------------------------	--	----------------------------------------------

As Whom

RunAs User category the rule applies to: Anyone Specified Users and Groups

<input type="checkbox"/> RunAs Users	External	Delete + Add
<input type="checkbox"/> thirdpartyapp	True	

<input type="checkbox"/> Groups of RunAs Users		Delete + Add
------------------------------------------------	--	----------------------------------------------

RunAs Group category the rule applies to: Any Group Specified Groups

<input type="checkbox"/> RunAs Groups	External	Delete + Add
---------------------------------------	----------	----------------------------------------------

**NOTE**

Propagating the changes from the server to the client can take a few minutes.

Verification steps

1. Log in to the **idmclient** host as the **idm_user** account.
2. Test the new sudo rule:
 - a. Display which **sudo** rules the **idm_user** account is allowed to perform.

```
[idm_user@idmclient ~]$ sudo -l
```

Matching Defaults entries for idm_user@idm.example.com on idmclient:

!visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,

```

env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
LS_COLORS",
env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES",
env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER
LC_TELEPHONE",
env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET
XAUTHORITY KRB5CCNAME",
secure_path=/sbin/:/bin:/usr/sbin:/usr/bin

```

User `idm_user@idm.example.com` may run the following commands on `idmclient`:

(thirdpartyapp) /opt/third-party-app/bin/report

- b. Run the **report** command as the **thirdpartyapp** service account.

```
[idm_user@idmclient ~]$ sudo -u thirdpartyapp /opt/third-party-app/bin/report
[sudo] password for idm_user@idm.example.com:
Executing report...
Report successful.
```

4.6. ENABLING GSSAPI AUTHENTICATION FOR SUDO ON AN IDM CLIENT

The following procedure describes enabling Generic Security Service Application Program Interface (GSSAPI) authentication on an IdM client for the **sudo** and **sudo -i** commands via the **pam_sss_gss.so** PAM module. With this configuration, IdM users can authenticate to the **sudo** command with their Kerberos ticket.

Prerequisites

- You have created a **sudo** rule for an IdM user that applies to an IdM host. For this example, you have created the **idm_user_reboot sudo** rule to grant the **idm_user** account the permission to run the **/usr/sbin/reboot** command on the **idmclient** host.
- The **idmclient** host is running RHEL 8.4 or later.
- You need **root** privileges to modify the **/etc/sssd/sssd.conf** file and PAM files in the **/etc/pam.d/** directory.

Procedure

1. Open the **/etc/sssd/sssd.conf** configuration file.
2. Add the following entry to the **[domain/<domain_name>]** section.

```
[domain/<domain_name>]
pam_gssapi_services = sudo, sudo-i
```

3. Save and close the **/etc/sssd/sssd.conf** file.
4. Restart the SSSD service to load the configuration changes.

```
[root@idmclient ~]# systemctl restart sssd
```

5. Open the **/etc/pam.d/sudo** PAM configuration file.
6. Add the following entry as the first line of the **auth** section in the **/etc/pam.d/sudo** file.

```
#%PAM-1.0
auth sufficient pam_sss_gss.so
auth include system-auth
account include system-auth
password include system-auth
session include system-auth
```

7. Save and close the **/etc/pam.d/sudo** file.
8. Open the **/etc/pam.d/sudo-i** PAM configuration file.
9. Add the following entry as the first line of the **auth** section in the **/etc/pam.d/sudo-i** file.

```
#%PAM-1.0
auth sufficient pam_sss_gss.so
auth include sudo
account include sudo
password include sudo
session optional pam_keyinit.so force revoke
session include sudo
```

10. Save and close the **/etc/pam.d/sudo-i** file.

Verification steps

1. Log into the host as the **idm_user** account.

```
[root@idm-client ~]# ssh -l idm_user@idm.example.com localhost
idm_user@idm.example.com's password:
```

2. Verify that you have a ticket-granting ticket as the **idm_user** account.

```
[idmuser@idmclient ~]$ klist
Ticket cache: KCM:1366201107
Default principal: idm_user@IDM.EXAMPLE.COM

Valid starting     Expires            Service principal
01/08/2021 09:11:48  01/08/2021 19:11:48
krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
renew until 01/15/2021 09:11:44
```

3. (Optional) If you do not have Kerberos credentials for the **idm_user** account, destroy your current Kerberos credentials and request the correct ones.

```
[idm_user@idmclient ~]$ kdestroy -A

[idm_user@idmclient ~]$ kinit idm_user@IDM.EXAMPLE.COM
Password for idm_user@idm.example.com:
```

- Reboot the machine using **sudo**, without specifying a password.

```
[idm_user@idmclient ~]$ sudo /usr/sbin/reboot
```

Additional resources

- The GSSAPI entry in the [IdM terminology](#) listing
- [Granting sudo access to an IdM user on an IdM client using IdM Web UI](#)
- [Granting sudo access to an IdM user on an IdM client using the CLI](#)
- [**pam_sss_gss \(8\)** man page](#)
- [**sssd.conf \(5\)** man page](#)

4.7. ENABLING GSSAPI AUTHENTICATION AND ENFORCING KERBEROS AUTHENTICATION INDICATORS FOR SUDO ON AN IDM CLIENT

The following procedure describes enabling Generic Security Service Application Program Interface (GSSAPI) authentication on an IdM client for the **sudo** and **sudo -i** commands via the **pam_sss_gss.so** PAM module. Additionally, only users who have logged in with a smart card will authenticate to those commands with their Kerberos ticket.



NOTE

You can use this procedure as a template to configure GSSAPI authentication with SSSD for other PAM-aware services, and further restrict access to only those users that have a specific authentication indicator attached to their Kerberos ticket.

Prerequisites

- You have created a **sudo** rule for an IdM user that applies to an IdM host. For this example, you have created the **idm_user_reboot sudo** rule to grant the **idm_user** account the permission to run the **/usr/sbin/reboot** command on the **idmclient** host.
- You have configured smart card authentication for the **idmclient** host.
- The **idmclient** host is running RHEL 8.4 or later.
- You need **root** privileges to modify the **/etc/sssd/sssd.conf** file and PAM files in the **/etc/pam.d** directory.

Procedure

- Open the **/etc/sssd/sssd.conf** configuration file.
- Add the following entries to the **[domain/<domain_name>]** section.

```
[domain/<domain_name>]
pam_gssapi_services = sudo, sudo-i
pam_gssapi_indicators_map = sudo:pkinit, sudo-i:pkinit
```

3. Save and close the **/etc/sssd/sssd.conf** file.
4. Restart the SSSD service to load the configuration changes.

```
[root@idmclient ~]# systemctl restart sssd
```

5. Open the **/etc/pam.d/sudo** PAM configuration file.
6. Add the following entry as the first line of the **auth** section in the **/etc/pam.d/sudo** file.

```
#%PAM-1.0
auth sufficient pam_sss_gss.so
auth include system-auth
account include system-auth
password include system-auth
session include system-auth
```

7. Save and close the **/etc/pam.d/sudo** file.
8. Open the **/etc/pam.d/sudo-i** PAM configuration file.
9. Add the following entry as the first line of the **auth** section in the **/etc/pam.d/sudo-i** file.

```
#%PAM-1.0
auth sufficient pam_sss_gss.so
auth include sudo
account include sudo
password include sudo
session optional pam_keyinit.so force revoke
session include sudo
```

10. Save and close the **/etc/pam.d/sudo-i** file.

Verification steps

1. Log into the host as the **idm_user** account and authenticate with a smart card.

```
[root@idmclient ~]# ssh -l idm_user@idm.example.com localhost
PIN for smart_card
```

2. Verify that you have a ticket-granting ticket as the smart card user.

```
[idm_user@idmclient ~]$ klist
Ticket cache: KEYRING:persistent:1358900015:krb_cache_TObtNMD
Default principal: idm_user@IDM.EXAMPLE.COM

Valid starting     Expires            Service principal
02/15/2021 16:29:48  02/16/2021 02:29:48
krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
renew until 02/22/2021 16:29:44
```

3. Display which **sudo** rules the **idm_user** account is allowed to perform.

```
[idm_user@idmclient ~]$ sudo -l
```

Matching Defaults entries for **idmuser** on **idmclient**:

```
!visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR
LS_COLORS",
env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT
LC_MESSAGES",
env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY
KRB5CCNAME",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin
```

User **idm_user** may run the following commands on **idmclient**:

(root) /usr/sbin/reboot

4. Reboot the machine using **sudo**, without specifying a password.

```
[idm_user@idmclient ~]$ sudo /usr/sbin/reboot
```

Additional resources

- The GSSAPI entry in the [IdM terminology](#) listing
- [Configuring Identity Management for smart card authentication](#)
- [Kerberos authentication indicators](#)
- [Granting sudo access to an IdM user on an IdM client using IdM Web UI](#)
- [Granting sudo access to an IdM user on an IdM client using the CLI](#)
- [**pam_sss_gss \(8\)** man page](#)
- [**sssd.conf \(5\)** man page](#)

4.8. SSSD OPTIONS CONTROLLING GSSAPI AUTHENTICATION FOR PAM SERVICES

You can use the following options for the **/etc/sssd/sssd.conf** configuration file to adjust the GSSAPI configuration within the SSSD service.

pam_gssapi_services

GSSAPI authentication with SSSD is disabled by default. You can use this option to specify a comma-separated list of PAM services that are allowed to try GSSAPI authentication using the **pam_sss_gss.so** PAM module. To explicitly disable GSSAPI authentication, set this option to **-**.

pam_gssapi_indicators_map

This option only applies to Identity Management (IdM) domains. Use this option to list Kerberos authentication indicators that are required to grant PAM access to a service. Pairs must be in the format **<PAM_service>:_<required_authentication_indicator>_**.

Valid authentication indicators are:

- **otp** for two-factor authentication
- **radius** for RADIUS authentication

- **pkinit** for PKINIT, smart card, or certificate authentication
- **hardened** for hardened passwords

pam_gssapi_check_upn

This option is enabled and set to **true** by default. If this option is enabled, the SSSD service requires that the user name matches the Kerberos credentials. If **false**, the **pam_sss_gss.so** PAM module authenticates every user that is able to obtain the required service ticket.

Examples

The following options enable Kerberos authentication for the **sudo** and **sudo-i** services, requires that **sudo** users authenticated with a one-time password, and user names must match the Kerberos principal. Because these settings are in the **[pam]** section, they apply to all domains:

```
[pam]
pam_gssapi_services = sudo, sudo-i
pam_gssapi_indicators_map = sudo:otp
pam_gssapi_check_upn = true
```

You can also set these options in individual **[domain]** sections to overwrite any global values in the **[pam]** section. The following options apply different GSSAPI settings to each domain:

For the **idm.example.com** domain

- Enable GSSAPI authentication for the **sudo** and **sudo -i** services.
- Require certificate or smart card authentication authenticators for the **sudo** command.
- Require one-time password authentication authenticators for the **sudo -i** command.
- Enforce matching user names and Kerberos principals.

For the **ad.example.com** domain

- Enable GSSAPI authentication only for the **sudo** service.
- Do not enforce matching user names and principals.

```
[domain/idm.example.com]
pam_gssapi_services = sudo, sudo-i
pam_gssapi_indicators_map = sudo:pkinit, sudo-i:otp
pam_gssapi_check_upn = true
...
```

```
[domain/ad.example.com]
pam_gssapi_services = sudo
pam_gssapi_check_upn = false
...
```

Additional resources

- [Kerberos authentication indicators](#)

4.9. TROUBLESHOOTING GSSAPI AUTHENTICATION FOR SUDO

If you are unable to authenticate to the **sudo** service with a Kerberos ticket from IdM, use the following scenarios to troubleshoot your configuration.

Prerequisites

- You have enabled GSSAPI authentication for the **sudo** service. See [Enabling GSSAPI authentication for sudo on an IdM client](#).
- You need **root** privileges to modify the **/etc/sssd/sssd.conf** file and PAM files in the **/etc/pam.d** directory.

Procedure

- If you see the following error, the Kerberos service might not be able to resolve the correct realm for the service ticket based on the host name:

Server not found in Kerberos database

In this situation, add the hostname directly to **[domain_realm]** section in the **/etc/krb5.conf** Kerberos configuration file:

```
[idm-user@idm-client ~]$ cat /etc/krb5.conf
...
[domain_realm]
.example.com = EXAMPLE.COM
example.com = EXAMPLE.COM
server.example.com = EXAMPLE.COM
```

- If you see the following error, you do not have any Kerberos credentials:

No Kerberos credentials available

In this situation, retrieve Kerberos credentials with the **kinit** utility or authenticate with SSSD:

```
[idm-user@idm-client ~]$ kinit idm-user@IDM.EXAMPLE.COM
Password for idm-user@idm.example.com:
```

- If you see either of the following errors in the **/var/log/sssd/sssd_pam.log** log file, the Kerberos credentials do not match the username of the user currently logged in:

User with UPN [<UPN>] was not found.

UPN [<UPN>] does not match target user [<username>].

In this situation, verify that you authenticated with SSSD, or consider disabling the **pam_gssapi_check_upn** option in the **/etc/sssd/sssd.conf** file:

```
[idm-user@idm-client ~]$ cat /etc/sssd/sssd.conf
...

```

pam_gssapi_check_upn = false

- For additional troubleshooting, you can enable debugging output for the **pam_sss_gss.so** PAM module.
 - Add the **debug** option at the end of all **pam_sss_gss.so** entries in PAM files, such as **/etc/pam.d/sudo** and **/etc/pam.d/sudo-i**:

```
[root@idm-client ~]# cat /etc/pam.d/sudo
#%PAM-1.0
auth sufficient pam_sss_gss.so debug
auth include system-auth
account include system-auth
password include system-auth
session include system-auth
```

```
[root@idm-client ~]# cat /etc/pam.d/sudo-i
#%PAM-1.0
auth sufficient pam_sss_gss.so debug
auth include sudo
account include sudo
password include sudo
session optional pam_keyinit.so force revoke
session include sudo
```

- Try to authenticate with the **pam_sss_gss.so** module and review the console output. In this example, the user did not have any Kerberos credentials.

```
[idm-user@idm-client ~]$ sudo ls -l /etc/sssd/sssd.conf
pam_sss_gss: Initializing GSSAPI authentication with SSSD
pam_sss_gss: Switching euid from 0 to 1366201107
pam_sss_gss: Trying to establish security context
pam_sss_gss: SSSD User name: idm-user@idm.example.com
pam_sss_gss: User domain: idm.example.com
pam_sss_gss: User principal:
pam_sss_gss: Target name: host@idm.example.com
pam_sss_gss: Using ccache: KCM:
pam_sss_gss: Acquiring credentials, principal name will be derived
pam_sss_gss: Unable to read credentials from [KCM:] [maj:0xd0000, min:0x96c73ac3]
pam_sss_gss: GSSAPI: Unspecified GSS failure. Minor code may provide more
information
pam_sss_gss: GSSAPI: No credentials cache found
pam_sss_gss: Switching euid from 1366200907 to 0
pam_sss_gss: System error [5]: Input/output error
```

4.10. USING AN ANSIBLE PLAYBOOK TO ENSURE SUDO ACCESS FOR AN IDM USER ON AN IDM CLIENT

In Identity Management (IdM), you can ensure **sudo** access to a specific command is granted to an IdM user account on a specific IdM host.

Complete this procedure to ensure a **sudo** rule named **idm_user_reboot** exists. The rule grants **idm_user** the permission to run the **/usr/sbin/reboot** command on the **idmclient** machine.

Prerequisites

- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- You know the IdM administrator password.
- You have [ensured the presence of a user account for `idm_user` in IdM and unlocked the account by creating a password for the user](#). For details on adding a new IdM user using the command-line interface, see [Adding users using the command line](#).
- No local `idm_user` account exists on `idmclient`. The `idm_user` user is not listed in the `/etc/passwd` file on `idmclient`.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaservers** in it:

```
[ipaservers]
server.idm.example.com
```

2. Add one or more **sudo** commands:

- a. Create an **ensure-reboot-sudocmd-is-present.yml** Ansible playbook that ensures the presence of the `/usr/sbin/reboot` command in the IdM database of **sudo** commands. To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/sudocmd/ensure-sudocmd-is-present.yml` file:

```
---
- name: Playbook to manage sudo command
  hosts: ipaserver
  become: true

  tasks:
    # Ensure sudo command is present
    - ipasudocmd:
        ipaadmin_password: MySecret123
        name: /usr/sbin/reboot
        state: present
```

- b. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-reboot-sudocmd-is-present.yml
```

3. Create a **sudo** rule that references the commands:

- a. Create an **ensure-sudorule-for-idmuser-on-idmclient-is-present.yml** Ansible playbook that uses the **sudo** command entry to ensure the presence of a sudo rule. The sudo rule allows `idm_user` to reboot the `idmclient` machine. To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/sudorule/ensure-sudorule-is-present.yml` file:

```
---
- name: Tests
  hosts: ipaserver
```

```
become: true
```

tasks:

```
# Ensure a sudorule is present granting idm_user the permission to run /usr/sbin/reboot
on idmclient
- ipasudorule:
    ipaadmin_password: MySecret123
    name: idm_user_reboot
    description: A test sudo rule.
    allow_sudocmd: /usr/sbin/reboot
    host: idmclient.idm.example.com
    user: idm_user
    state: present
```

b. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-sudorule-for-idmuser-on-idmclient-is-
present.yml
```

Verification steps

Test that the **sudo** rule whose presence you have ensured on the IdM server works on **idmclient** by verifying that **idm_user** can reboot **idmclient** using **sudo**. Note that it can take a few minutes for the changes made on the server to take effect on the client.

1. Log in to **idmclient** as **idm_user**.
2. Reboot the machine using **sudo**. Enter the password for **idm_user** when prompted:

```
$ sudo /usr/sbin/reboot
[sudo] password for idm_user:
```

If **sudo** is configured correctly, the machine reboots.

Additional materials

- For more details on how to apply **sudo** commands, command groups, and rules in IdM using an Ansible playbook including the descriptions of playbook variables, see the README-sudocmd.md, README-sudocmdgroup.md, and README-sudorule.md Markdown files available in the **/usr/share/doc/ansible-freeipa/** directory.

CHAPTER 5. USING LDAPMODIFY TO MANAGE IDM USERS EXTERNALLY

You can modify Identity Management (IdM) LDAP directly from the command-line interface (CLI) using the **ldapmodify** and **ldapdelete** utilities. The utilities provide full functionality for adding, editing, and deleting your directory contents. You can use these utilities to manage both the configuration entries of the server and the data in the user entries. The utilities can also be used to write scripts to perform bulk management of one or more directories.

5.1. TEMPLATES FOR MANAGING IDM USER ACCOUNTS EXTERNALLY

This section describes templates for various user management operations in IdM. The templates show which attributes you must modify using **ldapmodify** to achieve the following goals:

- Adding a new stage user
- Modifying a user’s attribute
- Enabling a user
- Disabling a user
- Preserving a user

The templates are formatted in the LDAP Data Interchange Format (LDIF). LDIF is a standard plain text data interchange format for representing LDAP directory content and update requests.

Using the templates, you can configure the LDAP provider of your provisioning system to manage IdM user accounts.

For detailed example procedures, see the following sections:

- [Adding an IdM stage user defined in an LDIF file](#)
- [Adding an IdM stage user directly from the CLI using ldapmodify](#)
- [Preserving an IdM user with ldapmodify](#)

Templates for adding a new stage user

- A template for adding a user with **UID and GID assigned automatically**. The distinguished name (DN) of the created entry must start with **uid=user_login**:

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

- A template for adding a user with **UID and GID assigned statically**

```

dn: uid=user_login,cn=staged
users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com
changetype: add
objectClass: top
objectClass: person
objectClass: inetorgperson
objectClass: organizationalperson
objectClass: posixaccount
uid: user_login
uidNumber: UID_number
gidNumber: GID_number
sn: surname
givenName: first_name
cn: full_name
homeDirectory: /home/user_login

```

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

Templates for modifying existing users

- **Modifying a user's attribute**

```

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

```

- **Disabling a user**:

```

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

```

- **Enabling a user**:

```

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

```

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

- **Preserving a user**:

```

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

```



NOTE

Before modifying a user, obtain the user's distinguished name (DN) by searching using the user's login. In the following example, the `user_allowed_to_modify_user_entries` user is a user allowed to modify user and group information, for example `activator` or IdM administrator. The password in the example is this user's password:

```
[...]
# ldapsearch -LLL -x -D
"uid=user_allowed_to_modify_user_entries,cn=users,cn=accounts,dc=idm,dc=example,dc=com" -w "Secret123" -H ldap://r8server.idm.example.com -b
"cn=users,cn=accounts,dc=idm,dc=example,dc=com" uid=test_user
dn: uid=test_user,cn=users,cn=accounts,dc=idm,dc=example,dc=com
memberOf: cn=ipausers,cn=groups,cn=accounts,dc=idm,dc=example,dc=com
```

5.2. TEMPLATES FOR MANAGING IDM GROUP ACCOUNTS EXTERNALLY

This section describes templates for various user group management operations in IdM. The templates show which attributes you must modify using `ldapmodify` to achieve the following aims:

- Creating a new group
- Deleting an existing group
- Adding a member to a group
- Removing a member from a group

The templates are formatted in the LDAP Data Interchange Format (LDIF). LDIF is a standard plain text data interchange format for representing LDAP directory content and update requests.

Using the templates, you can configure the LDAP provider of your provisioning system to manage IdM group accounts.

Creating a new group

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

Modifying groups

- Deleting an existing group:

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

- Adding a member to a group

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

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

- Removing a member from a group

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

NOTE

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

```
# ldapsearch -Y GSSAPI -H ldap://server.idm.example.com -b
"cn=groups,cn=accounts,dc=idm,dc=example,dc=com" "cn=group_name"
dn: cn=group_name,cn=groups,cn=accounts,dc=idm,dc=example,dc=com
ipaNTSecurityIdentifier: S-1-5-21-1650388524-2605035987-2578146103-11017
cn: testgroup
objectClass: top
objectClass: groupofnames
objectClass: nestedgroup
objectClass: ipausergroup
objectClass: ipaobject
objectClass: posixgroup
objectClass: ipantgroupattrs
ipaUniqueID: 569bf864-9d45-11ea-bea3-525400f6f085
gidNumber: 1997010017
```

5.3. PRESERVING AN IDM USER WITH LDAPMODIFY

This section describes how to use **ldapmodify** to preserve an IdM user; that is, how to deactivate a user account after the employee has left the company.

Prerequisites

- You can authenticate as an IdM user with a role to preserve users.

Procedure

1. Log in as an IdM user with a role to preserve users:

```
$ kinit admin
```

2. Enter the **ldapmodify** command and specify the Generic Security Services API (GSSAPI) as the Simple Authentication and Security Layer (SASL) mechanism to be used for authentication:

```
# ldapmodify -Y GSSAPI
SASL/GSSAPI authentication started
SASL username: admin@IDM.EXAMPLE.COM
SASL SSF: 256
SASL data security layer installed.
```

3. Enter the **dn** of the user you want to preserve:

```
dn: uid=user1,cn=users,cn=accounts,dc=idm,dc=example,dc=com
```

4. Enter **modrdn** as the type of change you want to perform:

```
changetype: modrdn
```

5. Specify the **newrdn** for the user:

```
newrdn: uid=user1
```

6. Indicate that you want to preserve the user:

```
deleteoldrdn: 0
```

7. Specify the **new superior DN**:

```
newsuperior: cn=deleted users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com
```

Preserving a user moves the entry to a new location in the directory information tree (DIT). For this reason, you must specify the DN of the new parent entry as the new superior DN.

8. Press **Enter** again to confirm that this is the end of the entry:

```
[Enter]
```

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

9. Exit the connection using **Ctrl + C**.

Verification steps

- Verify that the user has been preserved by listing all preserved users:

```
$ ipa user-find --preserved=true
-----
1 user matched
-----
User login: user1
First name: First 1
```

Last name: Last 1
Home directory: /home/user1
Login shell: /bin/sh
Principal name: [user1@IDM.EXAMPLE.COM](#)
Principal alias: [user1@IDM.EXAMPLE.COM](#)
Email address: [user1@idm.example.com](#)
UID: 1997010003
GID: 1997010003
Account disabled: True
Preserved user: True

Number of entries returned 1

CHAPTER 6. CONFIGURING IDM FOR EXTERNAL PROVISIONING OF USERS

As a system administrator, you can configure Identity Management (IdM) to support the provisioning of users by an external solution for managing identities.

Rather than use the **ipa** utility, the administrator of the external provisioning system can access the IdM LDAP using the **ldapmodify** utility. The administrator can add individual stage users [from the CLI using ldapmodify](#) or [using an LDIF file](#).

The assumption is that you, as an IdM administrator, fully trust your external provisioning system to only add validated users. However, at the same time you do not want to assign the administrators of the external provisioning system the IdM role of **User Administrator** to enable them to add new active users directly.

You can [configure a script](#) to automatically move the staged users created by the external provisioning system to active users automatically.

This chapter contains these sections:

1. [Preparing Identity Management \(IdM\)](#) to use an external provisioning system to add stage users to IdM.
2. [Creating a script](#) to move the users added by the external provisioning system from stage to active users.
3. Using an external provisioning system to add an IdM stage user. You can do that in two ways:
 - [Add an IdM stage user using an LDIF file](#)
 - [Add an IdM stage user directly from the CLI using ldapmodify](#)

Additional materials

For examples and templates for using **ldapmodify** as a full IdM administrator to perform user and group management operations that require higher privileges, see [Using ldapmodify to manage IdM users externally](#).

6.1. PREPARING IDM ACCOUNTS FOR AUTOMATIC ACTIVATION OF STAGE USER ACCOUNTS

This procedure shows how to configure two IdM user accounts to be used by an external provisioning system. By adding the accounts to a group with an appropriate password policy, you enable the external provisioning system to manage user provisioning in IdM. In the following, the user account to be used by the external system to add stage users is named **provisionator**. The user account to be used to automatically activate the stage users is named **activator**.

Prerequisites

- The host on which you perform the procedure is enrolled into IdM.

Procedure

1. Log in as IdM administrator:

```
$ kinit admin
```

2. Create a user named **provisionator** with the privileges to add stage users.

- a. Add the provisionator user account:

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

- a. Grant the provisionator user the required privileges.

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

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

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

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

- iii. Add the provisionator user to the role:

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

- iv. Verify that the provisionator exists in IdM:

```
$ ipa user-find provisionator --all --raw
```

```
-----  
1 user matched  
-----
```

```
dn: uid=provisionator,cn=users,cn=accounts,dc=idm,dc=example,dc=com  
uid: provisionator
```

```
[...]
```

3. Create a user, **activator**, with the privileges to manage user accounts.

- a. Add the activator user account:

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

- b. Grant the activator user the required privileges by adding the user to the default **User Administrator** role:

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

4. Create a user group for application accounts:

```
$ ipa group-add application-accounts
```

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



```
$ ipa pwpolicy-add application-accounts --maxlife=10000 --minlife=0 --history=0 --minclasses=4 --minlength=8 --priority=1 --maxfail=0 --failinterval=1 --lockouttime=0
```

6. (Optional) Verify that the password policy exists in IdM:

```
$ ipa pwpolicy-show application-accounts
Group: application-accounts
Max lifetime (days): 10000
Min lifetime (hours): 0
History size: 0
[...]
```

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

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

8. Change the passwords for the user accounts:

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

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

Additional resources:

- For details on adding new users, see [Managing user accounts using the command line](#).
- For details on granting users the privileges required to manage other user accounts, see [Delegating Permissions over Users](#).
- For details on managing IdM password policies, see [Defining IdM Password Policies](#).

6.2. CONFIGURING AUTOMATIC ACTIVATION OF IDM STAGE USER ACCOUNTS

This procedure shows how to create a script for activating stage users. The system runs the script automatically at specified time intervals. This ensures that new user accounts are automatically activated and available for use shortly after they are created.



IMPORTANT

The procedure assumes that the owner of the external provisioning system has already validated the users and that they do not require additional validation on the IdM side before the script adds them to IdM.

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

Prerequisites

- The **provisionator** and **activator** accounts exist in IdM. For details, see [Preparing IdM accounts for automatic activation of stage user accounts](#).
- You have root privileges on the IdM server on which you are running the procedure.

- You are logged in as IdM administrator.
- You trust your external provisioning system.

Procedure

1. Generate a keytab file for the activation account:

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

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

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

```
#!/bin/bash

kinit -k -i activator

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

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

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

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

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

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

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

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

[Timer]
OnBootSec=15min
OnUnitActiveSec=1min

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

6. Reload the new configuration:

```
# systemctl daemon-reload
```

7. Enable **ipa-activate-all.timer**:

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

8. Start **ipa-activate-all.timer**:

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

9. (Optional) Verify that the **ipa-activate-all.timer** daemon is running:

```
# systemctl status ipa-activate-all.timer
● ipa-activate-all.timer - Scan IdM every minute for any stage users that must be activated
  Loaded: loaded (/etc/systemd/system/ipa-activate-all.timer; enabled; vendor preset: disabled)
  Active: active (waiting) since Wed 2020-06-10 16:34:55 CEST; 15s ago
    Trigger: Wed 2020-06-10 16:35:55 CEST; 44s left

Jun 10 16:34:55 server.idm.example.com systemd[1]: Started Scan IdM every minute for any stage users that must be activated.
```

6.3. ADDING AN IDM STAGE USER DEFINED IN AN LDIF FILE

This section describes how an administrator of an external provisioning system can access IdM LDAP and use an LDIF file to add stage users. While the example below shows adding one single user, multiple users can be added in one file in bulk mode.

Prerequisites

- IdM administrator has created the **provisionator** account and a password for it. For details, see [Preparing IdM accounts for automatic activation of stage user accounts](#) .
- You as the external administrator know the password of the **provisionator** account.
- You can SSH to the IdM server from your LDAP server.
- You are able to supply the minimal set of attributes that an IdM stage user must have to allow the correct processing of the user life cycle, namely:
 - The **distinguished name** (dn)
 - The **common name** (cn)
 - The **last name** (sn)
 - The **uid**

Procedure

1. On the external server, create an LDIF file that contains information about the new user:

```
dn: uid=stageidmuser,cn=staged
users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com
changetype: add
objectClass: top
```

```
objectClass: inetorgperson
uid: stageidmuser
sn: surname
givenName: first_name
cn: full_name
```

- Transfer the LDIF file from the external server to the IdM server:

```
$ scp add-stageidmuser.ldif provisionator@server.idm.example.com:/provisionator/
Password:
add-stageidmuser.ldif                                         100% 364
217.6KB/s 00:00
```

- Use the **SSH** protocol to connect to the IdM server as **provisionator**:

```
$ ssh provisionator@server.idm.example.com
Password:
[provisionator@server ~]$
```

- On the IdM server, obtain the Kerberos ticket-granting ticket (TGT) for the provisionator account:

```
[provisionator@server ~]$ kinit provisionator
```

- Enter the **ldapadd** command with the -f option and the name of the LDIF file. Specify the name of the IdM server and the port number:

```
~]$ ldapadd -h server.idm.example.com -p 389 -f add-stageidmuser.ldif
SASL/GSSAPI authentication started
SASL username: provisionator@IDM.EXAMPLE.COM
SASL SSF: 256
SASL data security layer installed.
adding the entry "uid=stageidmuser,cn=staged
users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com"
```

6.4. ADDING AN IDM STAGE USER DIRECTLY FROM THE CLI USING LDAPMODIFY

This section describes how an administrator of an external provisioning system can access the Identity Management (IdM) LDAP and use the **ldapmodify** utility to add a stage user.

Prerequisites

- The IdM administrator has created the **provisionator** account and a password for it. For details, see [Preparing IdM accounts for automatic activation of stage user accounts](#).
- You as the external administrator know the password of the **provisionator** account.
- You can SSH to the IdM server from your LDAP server.
- You are able to supply the minimal set of attributes that an IdM stage user must have to allow the correct processing of the user life cycle, namely:

- The **distinguished name** (dn)
- The **common name** (cn)
- The **last name** (sn)
- The **uid**

Procedure

1. Use the **SSH** protocol to connect to the IdM server using your IdM identity and credentials:

```
$ ssh provisionator@server.idm.example.com  
Password:  
[provisionator@server ~]$
```

2. Obtain the TGT of the **provisionator** account, an IdM user with a role to add new stage users:

```
$ kinit provisionator
```

3. Enter the **ldapmodify** command and specify Generic Security Services API (GSSAPI) as the Simple Authentication and Security Layer (SASL) mechanism to use for authentication. Specify the name of the IdM server and the port:

```
# ldapmodify -h server.idm.example.com -p 389 -Y GSSAPI  
SASL/GSSAPI authentication started  
SASL username: provisionator@IDM.EXAMPLE.COM  
SASL SSF: 56  
SASL data security layer installed.
```

4. Enter the **dn** of the user you are adding:

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

5. Enter **add** as the type of change you are performing:

```
changetype: add
```

6. Specify the LDAP object class categories required to allow the correct processing of the user life cycle:

```
objectClass: top  
objectClass: inetorgperson
```

You can specify additional object classes.

7. Enter the **uid** of the user:

```
uid: stageuser
```

8. Enter the **cn** of the user:

```
cn: Babs Jensen
```

9. Enter the last name of the user:

```
sn: Jensen
```

10. Press **Enter** again to confirm that this is the end of the entry:

```
[Enter]
```

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

11. Exit the connection using **Ctrl + C**.

Verification steps

Verify the contents of the stage entry to make sure your provisioning system added all required POSIX attributes and the stage entry is ready to be activated.

- To display the new stage user's LDAP attributes, enter the **ipa stageuser-show --all --raw** command:

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

1. Note that the user is explicitly disabled by the **nsaccountlock** attribute.

CHAPTER 7. MANAGING SELF-SERVICE RULES IN IDM USING THE CLI

This chapter introduces self-service rules in Identity Management (IdM) and describes how to create and edit self-service access rules in the command-line interface (CLI).

7.1. SELF-SERVICE ACCESS CONTROL IN IDM

Self-service access control rules define which operations an Identity Management (IdM) entity can perform on its IdM Directory Server entry: for example, IdM users have the ability to update their own passwords.

This method of control allows an authenticated IdM entity to edit specific attributes within its LDAP entry, but does not allow **add** or **delete** operations on the entire entry.



WARNING

Be careful when working with self-service access control rules: configuring access control rules improperly can inadvertently elevate an entity's privileges.

7.2. CREATING SELF-SERVICE RULES USING THE CLI

This procedure describes creating self-service access rules in IdM using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

- To add a self-service rule, use the **ipa selfservice-add** command and specify the following two options:

--permissions

sets the **read** and **write** permissions the Access Control Instruction (ACI) grants.

--attrs

sets the complete list of attributes to which this ACI grants permission.

For example, to create a self-service rule allowing users to modify their own name details:

```
$ ipa selfservice-add "Users can manage their own name details" --permissions=write -- attrs=givenname --attrs=displayname --attrs=title --attrs initials  
-----  
Added selfservice "Users can manage their own name details"  
-----
```

Self-service name: Users can manage their own name details

Permissions: write

Attributes: givenname, displayname, title, initials

7.3. EDITING SELF-SERVICE RULES USING THE CLI

This procedure describes editing self-service access rules in IdM using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. *Optional:* Display existing self-service rules with the **ipa selfservice-find** command.
2. *Optional:* Display details for the self-service rule you want to modify with the **ipa selfservice-show** command.
3. Use the **ipa selfservice-mod** command to edit a self-service rule.

For example:

```
$ ipa selfservice-mod "Users can manage their own name details" --attrs=givenname -- attrs=displayname --attrs=title --attrs=initials --attrs=surname
```

```
-----
```

Modified selfservice "Users can manage their own name details"

```
-----
```

Self-service name: Users can manage their own name details

Permissions: write

Attributes: givenname, displayname, title, initials



IMPORTANT

Using the **ipa selfservice-mod** command overwrites the previously defined permissions and attributes, so always include the complete list of existing permissions and attributes along with any new ones you want to define.

Verification steps

- Use the **ipa selfservice-show** command to display the self-service rule you edited.

```
$ ipa selfservice-show "Users can manage their own name details"
```

```
-----
```

Self-service name: Users can manage their own name details

Permissions: write

Attributes: givenname, displayname, title, initials

7.4. DELETING SELF-SERVICE RULES USING THE CLI

This procedure describes deleting self-service access rules in IdM using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

- Use the **ipa selfservice-del** command to delete a self-service rule.

For example:

```
$ ipa selfservice-del "Users can manage their own name details"
-----
Deleted selfservice "Users can manage their own name details"
```

Verification steps

- Use the **ipa selfservice-find** command to display all self-service rules. The rule you just deleted should be missing.

CHAPTER 8. MANAGING SELF-SERVICE RULES USING THE IDM WEB UI

This chapter introduces self-service rules in Identity Management (IdM) and describes how to create and edit self-service access rules in the web interface (IdM Web UI).

8.1. SELF-SERVICE ACCESS CONTROL IN IDM

Self-service access control rules define which operations an Identity Management (IdM) entity can perform on its IdM Directory Server entry: for example, IdM users have the ability to update their own passwords.

This method of control allows an authenticated IdM entity to edit specific attributes within its LDAP entry, but does not allow **add** or **delete** operations on the entire entry.



WARNING

Be careful when working with self-service access control rules: configuring access control rules improperly can inadvertently elevate an entity's privileges.

8.2. CREATING SELF-SERVICE RULES USING THE IDM WEB UI

This procedure describes how to create self-service access rules in IdM using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Self Service Permissions**.
2. Click **Add** at the top-right of the list of the self-service access rules:

The screenshot shows the IPA Server interface with the 'Identity' tab selected. Under 'Role Based Access Control', the 'Self Service Permissions' option is highlighted. On the right, there is a list of permissions with three items: 'Users can manage their own SSH public keys', 'Self can write own password', and 'User Self service'. Below the list is a search bar and buttons for 'Refresh', 'Delete', and '+ Add' (which is highlighted with a red box).

3. The **Add Self Service Permission** window opens. Enter the name of the new self-service rule in the **Self-service name** field. Spaces are allowed:

The dialog box has a title 'Add Self Service Permission' with a close button. It contains two main sections: 'Self-service * name' (containing 'Adding Personal Info') and 'Attributes *'. The 'Attributes' section includes a 'Filter' input, a search icon, and an 'Add' button. A list of attributes with checkboxes includes: audio, carlicense, departmentnumber, homedirectory, homepostaladdress, inetuserstatus, internationalisdnnumber, ipatokenradiusconfiglink, ipauniqueid, jpegphoto (checked), businesscategory, cn, description, homephone, inetuserhttppurl, initials (checked), ipasshpubkey, ipatokenradiususername, ipauserauthtype, and krbcCanonicalname.

4. Select the check boxes next to the attributes you want users to be able to edit.
5. *Optional:* If an attribute you would like to provide access to is not listed, you can add a listing for it:
 - a. Click the **Add** button.
 - b. Enter the attribute name in the **Attribute** text field of the following **Add Custom Attribute** window.
 - c. Click the **OK** button to add the attribute

- d. Verify that the new attribute is selected
- 6. Click the **Add** button at the bottom of the form to save the new self-service rule.
Alternatively, you can save and continue editing the self-service rule by clicking the **Add and Edit** button, or save and add further rules by clicking the **Add and Add another** button.

8.3. EDITING SELF-SERVICE RULES USING THE IDM WEB UI

This procedure describes how to edit self-service access rules in IdM using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Self Service Permissions**.
2. Click on the name of the self-service rule you want to modify.

General																					
Self-service name	User Self service																				
Attributes *	<input type="text" value="Filter"/> <input type="button" value="Add"/> <table border="0"> <tr> <td><input type="checkbox"/> audio</td> <td><input checked="" type="checkbox"/> businesscategory</td> </tr> <tr> <td><input checked="" type="checkbox"/> carlicense</td> <td><input checked="" type="checkbox"/> cn</td> </tr> <tr> <td><input type="checkbox"/> departmentnumber</td> <td><input checked="" type="checkbox"/> description</td> </tr> <tr> <td><input type="checkbox"/> destinationindicator</td> <td><input checked="" type="checkbox"/> displayname</td> </tr> <tr> <td><input type="checkbox"/> employeeonenumber</td> <td><input checked="" type="checkbox"/> employeetype</td> </tr> <tr> <td><input checked="" type="checkbox"/> facsimiletelephonenumber</td> <td><input checked="" type="checkbox"/> gecos</td> </tr> <tr> <td><input type="checkbox"/> gidnumber</td> <td><input checked="" type="checkbox"/> givenname</td> </tr> <tr> <td><input type="checkbox"/> homedirectory</td> <td><input checked="" type="checkbox"/> homephone</td> </tr> <tr> <td><input type="checkbox"/> homepostaladdress</td> <td><input checked="" type="checkbox"/> inetuserhttppurl</td> </tr> <tr> <td><input type="checkbox"/> inetuserstatus</td> <td><input checked="" type="checkbox"/> initials</td> </tr> </table>	<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory	<input checked="" type="checkbox"/> carlicense	<input checked="" type="checkbox"/> cn	<input type="checkbox"/> departmentnumber	<input checked="" type="checkbox"/> description	<input type="checkbox"/> destinationindicator	<input checked="" type="checkbox"/> displayname	<input type="checkbox"/> employeeonenumber	<input checked="" type="checkbox"/> employeetype	<input checked="" type="checkbox"/> facsimiletelephonenumber	<input checked="" type="checkbox"/> gecos	<input type="checkbox"/> gidnumber	<input checked="" type="checkbox"/> givenname	<input type="checkbox"/> homedirectory	<input checked="" type="checkbox"/> homephone	<input type="checkbox"/> homepostaladdress	<input checked="" type="checkbox"/> inetuserhttppurl	<input type="checkbox"/> inetuserstatus	<input checked="" type="checkbox"/> initials
<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory																				
<input checked="" type="checkbox"/> carlicense	<input checked="" type="checkbox"/> cn																				
<input type="checkbox"/> departmentnumber	<input checked="" type="checkbox"/> description																				
<input type="checkbox"/> destinationindicator	<input checked="" type="checkbox"/> displayname																				
<input type="checkbox"/> employeeonenumber	<input checked="" type="checkbox"/> employeetype																				
<input checked="" type="checkbox"/> facsimiletelephonenumber	<input checked="" type="checkbox"/> gecos																				
<input type="checkbox"/> gidnumber	<input checked="" type="checkbox"/> givenname																				
<input type="checkbox"/> homedirectory	<input checked="" type="checkbox"/> homephone																				
<input type="checkbox"/> homepostaladdress	<input checked="" type="checkbox"/> inetuserhttppurl																				
<input type="checkbox"/> inetuserstatus	<input checked="" type="checkbox"/> initials																				

3. The edit page only allows you to edit the list of attributes to you want to add or remove to the self-service rule. Select or deselect the appropriate check boxes.
4. Click the **Save** button to save your changes to the self-service rule.

8.4. DELETING SELF-SERVICE RULES USING THE IDM WEB UI

This procedure describes how to delete self-service access rules in IdM using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Self Service Permissions**.
2. Select the check box next to the rule you want to delete, then click on the **Delete** button on the right of the list.

The screenshot shows the IdM Web UI interface. At the top, there is a navigation bar with tabs: Identity, Policy, Authentication, Network Services, and IPA Server. The IPA Server tab is selected. Below the navigation bar, there is a sub-navigation menu under the Role Based Access Control section, with options: Roles, Privileges, Permissions, Self Service Permissions (which is currently selected and highlighted in blue), and Delegations. To the right of this menu, there is a search bar with the placeholder text "missions". Below the search bar, there are three buttons: Refresh, Delete (which is highlighted with a red box), and Add. Underneath these buttons, there is a table listing three self-service permissions, each with a checkbox column:

	Users can manage their own SSH public keys	Self can write own password	User Self service
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/>			

3. A dialog opens, click on **Delete** to confirm.

CHAPTER 9. USING ANSIBLE PLAYBOOKS TO MANAGE SELF-SERVICE RULES IN IDM

This section introduces self-service rules in Identity Management (IdM) and describes how to create and edit self-service access rules using Ansible playbooks. Self-service access control rules allow an IdM entity to perform specified operations on its IdM Directory Server entry.

This section covers the following topics:

- [Self-service access control in IdM](#)
- [Using Ansible to ensure that a self-service rule is present](#)
- [Using Ansible to ensure that a self-service rule is absent](#)
- [Using Ansible to ensure that a self-service rule has specific attributes](#)
- [Using Ansible to ensure that a self-service rule does not have specific attributes](#)

9.1. SELF-SERVICE ACCESS CONTROL IN IDM

Self-service access control rules define which operations an Identity Management (IdM) entity can perform on its IdM Directory Server entry: for example, IdM users have the ability to update their own passwords.

This method of control allows an authenticated IdM entity to edit specific attributes within its LDAP entry, but does not allow **add** or **delete** operations on the entire entry.



WARNING

Be careful when working with self-service access control rules: configuring access control rules improperly can inadvertently elevate an entity's privileges.

9.2. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS PRESENT

The following procedure describes how to use an Ansible playbook to define self-service rules and ensure their presence on an Identity Management (IdM) server. In this example, the new **Users can manage their own name details** rule grants users the ability to change their own **givenname**, **displayname**, **title** and **initials** attributes. This allows them to, for example, change their display name or initials if they want to.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.

- You have installed the [ansible-freeipa](#) package.
- In the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **selfservice-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/selfservice/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/selfservice/selfservice-present.yml  
selfservice-present-copy.yml
```

3. Open the **selfservice-present-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **ipaselfservice** task section:
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the new self-service rule.
 - Set the **permission** variable to a comma-separated list of permissions to grant: **read** and **write**.
 - Set the **attribute** variable to a list of attributes that users can manage themselves: **givenname**, **displayname**, **title**, and **initials**.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Self-service present  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Ensure self-service rule "Users can manage their own name details" is present  
      ipaselfservice:  
        ipaadmin_password: Secret123  
        name: "Users can manage their own name details"  
        permission: read, write  
        attribute:  
          - givenname  
          - displayname  
          - title  
          - initials
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory selfservice-present-copy.yml
```

Additional resources

- For more information on the concept of self-service rules, see [Self-service access control in IdM](#).
- For more sample Ansible playbooks that use the **ipaselfservice** module, see:
 - The **README-selfservice.md** file available in the **/usr/share/doc/ansible-freeipa** directory. This file also contains the definitions of the **ipaselfservice** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/selfservice** directory.

9.3. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS ABSENT

The following procedure describes how to use an Ansible playbook to ensure a specified self-service rule is absent from your IdM configuration. The example below describes how to make sure the **Users can manage their own name details** self-service rule does not exist in IdM. This will ensure that users cannot, for example, change their own display name or initials.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - In the **~/MyPlaybooks/** directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.

Procedure

1. Navigate to the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **selfservice-absent.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/selfservice/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/selfservice/selfservice-absent.yml
selfservice-absent-copy.yml
```

3. Open the **selfservice-absent-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **ipaselfservice** task section:
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the self-service rule.
 - Set the **state** variable to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Self-service absent
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure self-service rule "Users can manage their own name details" is absent
      ipaselfservice:
        ipaadmin_password: Secret123
        name: "Users can manage their own name details"
        state: absent
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory selfservice-absent-copy.yml
```

Additional resources

- For more information on the concept of self-service rules, see [Self-service access control in IdM](#).
- For more sample Ansible playbooks that use the **ipaselfservice** module, see:
 - The **README-selfservice.md** file available in the **/usr/share/doc/ansible-freeipa** directory. This file also contains the definitions of the **ipaselfservice** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/selfservice** directory.

9.4. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE HAS SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that an already existing self-service rule has specific settings. In the example, you ensure the **Users can manage their own name details** self-service rule also has the **surname** member attribute.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - In the **~/MyPlaybooks/** directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
- The **Users can manage their own name details** self-service rule exists in IdM.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `selfservice-member-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/selfservice/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/selfservice/selfservice-member-present.yml selfservice-member-present-copy.yml
```

3. Open the `selfservice-member-present-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipaselfservice` task section:

- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the self-service rule to modify.
- Set the `attribute` variable to `surname`.
- Set the `action` variable to `member`.

This is the modified Ansible playbook file for the current example:

```
---
- name: Self-service member present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure selfservice "Users can manage their own name details" member attribute
      surname is present
      ipaselfservice:
        ipaadmin_password: Secret123
        name: "Users can manage their own name details"
        attribute:
        - surname
        action: member
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory selfservice-member-present-copy.yml
```

Additional resources

- For more information on the concept of self-service rules, see [Self-service access control in IdM](#).
- For more sample Ansible playbooks that use the `ipaselfservice` module, see:

- The **README-selfservice.md** file available in the **/usr/share/doc/ansible-freeipa** directory. This file also contains the definitions of the **ipaselfservice** variables.
- The **/usr/share/doc/ansible-freeipa/playbooks/selfservice** directory.

9.5. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE DOES NOT HAVE SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that a self-service rule does not have specific settings. You can use this playbook to make sure a self-service rule does not grant undesired access. In the example, you ensure the **Users can manage their own name details** self-service rule does not have the **givenname** and **surname** member attributes.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - In the **~/MyPlaybooks/** directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
- The **Users can manage their own name details** self-service rule exists in IdM.

Procedure

1. Navigate to the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **selfservice-member-absent.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/selfservice/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/selfservice/selfservice-member-absent.yml selfservice-member-absent-copy.yml
```

3. Open the **selfservice-member-absent-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipaselfservice** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the self-service rule you want to modify.
- Set the **attribute** variable to **givenname** and **surname**.
- Set the **action** variable to **member**.
- Set the **state** variable to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
```

```
- name: Self-service member absent
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure selfservice "Users can manage their own name details" member attributes
      givenname and surname are absent
      ipaselfservice:
        ipaadmin_password: Secret123
        name: "Users can manage their own name details"
        attribute:
          - givenname
          - surname
        action: member
        state: absent
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory selfservice-member-absent-copy.yml
```

Additional resources

- For more information on the concept of self-service rules, see [Self-service access control in IdM](#).
- For more sample Ansible playbooks that use the **ipaselfservice** module, see:
 - The **README-selfservice.md** file available in the **/usr/share/doc/ansible-freeipa/** directory. This file also contains the definitions of the **ipaselfservice** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/selfservice** directory.

CHAPTER 10. MANAGING USER GROUPS IN IDM CLI

This chapter introduces user groups management using the IdM CLI.

A user group is a set of users with common privileges, password policies, and other characteristics.

A user group in Identity Management (IdM) can include:

- IdM users
- other IdM user groups
- external users, which are users that exist outside of IdM

10.1. THE DIFFERENT GROUP TYPES IN IDM

IdM supports the following types of groups:

POSIX groups (the default)

POSIX groups support Linux POSIX attributes for their members. Note that groups that interact with Active Directory cannot use POSIX attributes.

POSIX attributes identify users as separate entities. Examples of POSIX attributes relevant to users include **uidNumber**, a user number (UID), and **gidNumber**, a group number (GID).

Non-POSIX groups

Non-POSIX groups do not support POSIX attributes. For example, these groups do not have a GID defined.

All members of this type of group must belong to the IdM domain.

External groups

Use external groups to add group members that exist in an identity store outside of the IdM domain, such as:

- A local system
- An Active Directory domain
- A directory service

External groups do not support POSIX attributes. For example, these groups do not have a GID defined.

Table 10.1. User groups created by default

Group name	Default group members
ipausers	All IdM users
admins	Users with administrative privileges, including the default admin user
editors	This is a legacy group that no longer has any special privileges

Group name	Default group members
trust admins	Users with privileges to manage the Active Directory trusts

When you add a user to a user group, the user gains the privileges and policies associated with the group. For example, to grant administrative privileges to a user, add the user to the **admins** group.



WARNING

Do not delete the **admins** group. As **admins** is a pre-defined group required by IdM, this operation causes problems with certain commands.

In addition, IdM creates *user private groups* by default whenever a new user is created in IdM. For more information about private groups, see [Adding users without a private group](#).

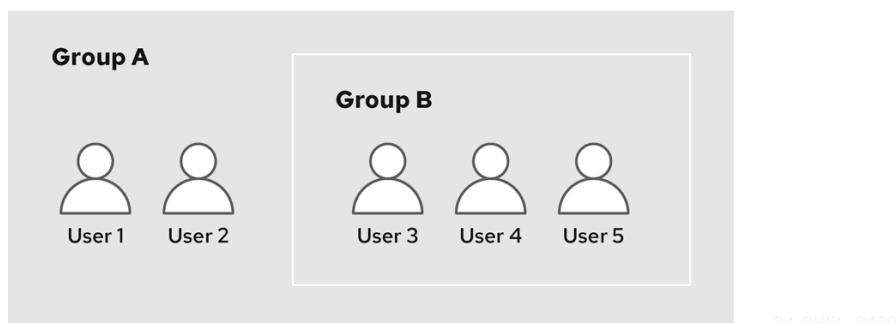
10.2. DIRECT AND INDIRECT GROUP MEMBERS

User group attributes in IdM apply to both direct and indirect members: when group B is a member of group A, all users in group B are considered indirect members of group A.

For example, in the following diagram:

- User 1 and User 2 are *direct members* of group A.
- User 3, User 4, and User 5 are *indirect members* of group A.

Figure 10.1. Direct and Indirect Group Membership



If you set a password policy for user group A, the policy also applies to all users in user group B.

10.3. ADDING A USER GROUP USING IDM CLI

This section describes how to add a user group using IdM CLI.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

Procedure

- Add a user group by using the **ipa group-add *group_name*** command. For example, to create group_a:

```
$ ipa group-add group_a
```

```
-----  
Added group "group_a"  
-----
```

```
Group name: group_a  
GID: 1133400009
```

By default, **ipa group-add** adds a POSIX user group. To specify a different group type, add options to **ipa group-add**:

- **--nonposix** to create a non-POSIX group
- **--external** to create an external group

For details on group types, see [The different group types in IdM](#).

You can specify a custom GID when adding a user group by using the **--gid=custom_GID** option. If you do this, be careful to avoid ID conflicts. If you do not specify a custom GID, IdM automatically assigns a GID from the available ID range.



WARNING

Do not add local groups to IdM. The Name Service Switch (NSS) always resolves IdM users and groups before resolving local users and groups. This means that, for example, IdM group membership does not work for local users.

10.4. SEARCHING FOR USER GROUPS USING IDM CLI

This section describes how to search for existing user groups using IdM CLI.

Procedure

- Display all user groups by using the **ipa group-find** command. To specify a group type, add options to **ipa group-find**:
 - Display all POSIX groups using the **ipa group-find --posix** command.
 - Display all non-POSIX groups using the **ipa group-find --nonposix** command.
 - Display all external groups using the **ipa group-find --external** command.

For more information on different group types, see [The different group types in IdM](#).

10.5. DELETING A USER GROUP USING IDM CLI

This section describes how to delete a user group using IdM CLI. Note that deleting a group does not delete the group members from IdM.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

Procedure

- Delete a user group by using the **ipa group-del *group_name*** command. For example, to delete group_a:

```
$ ipa group-del group_a
-----
Deleted group "group_a"
```

10.6. ADDING A MEMBER TO A USER GROUP USING IDM CLI

This section describes how to add a member to a user group using IdM CLI. You can add both users and user groups as members of a user group. For more information, see [The different group types in IdM](#) and [Direct and indirect group members](#).

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

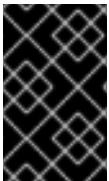
Procedure

- Add a member to a user group by using the **ipa group-add-member** command. Specify the type of member using these options:
 - **--users** adds an IdM user
 - **--external** adds a user that exists outside the IdM domain, in the format of **DOMAIN\user_name** or **user_name@domain**
 - **--groups** adds an IdM user group

For example, to add group_b as a member of group_a:

```
$ ipa group-add-member group_a --groups=group_b
Group name: group_a
GID: 1133400009
Member users: user_a
Member groups: group_b
Indirect Member users: user_b
-----
Number of members added 1
```

Members of group_b are now indirect members of group_a.



IMPORTANT

When adding a group as a member of another group, do not create recursive groups. For example, if Group A is a member of Group B, do not add Group B as a member of Group A. Recursive groups can cause unpredictable behavior.



NOTE

After you add a member to a user group, the update may take some time to spread to all clients in your Identity Management environment. This is because when any given host resolves users, groups and netgroups, the **System Security Services Daemon** (SSSD) first looks into its cache and performs server lookups only for missing or expired records.

10.7. ADDING USERS WITHOUT A USER PRIVATE GROUP

By default, IdM creates user private groups (UPGs) whenever a new user is created in IdM. UPGs are a specific group type:

- The UPG has the same name as the newly created user.
- The user is the only member of the UPG. The UPG cannot contain any other members.
- The GID of the private group matches the UID of the user.

However, it is possible to add users without creating a UPG.

10.7.1. Users without a user private group

If a NIS group or another system group already uses the GID that would be assigned to a user private group, it is necessary to avoid creating a UPG.

You can do this in two ways:

- Add a new user without a UPG, without disabling private groups globally. See [Adding a user without a user private group when private groups are globally enabled](#).
- Disable UPGs globally for all users, then add a new user. See [Disabling user private groups globally for all users](#) and [Adding a user when user private groups are globally disabled](#).

In both cases, IdM will require specifying a GID when adding new users, otherwise the operation will fail. This is because IdM requires a GID for the new user, but the default user group **ipausers** is a non-POSIX group and therefore does not have an associated GID. The GID you specify does not have to correspond to an already existing group.



NOTE

Specifying the GID does not create a new group. It only sets the GID attribute for the new user, because the attribute is required by IdM.

10.7.2. Adding a user without a user private group when private groups are globally enabled

You can add a user without creating a user private group (UPG) even when UPGs are enabled on the system. This requires manually setting a GID for the new user. For details on why this is needed, see [Section 10.7.1, "Users without a user private group"](#).

Procedure

- To prevent IdM from creating a UPG, add the **--noprivate** option to the **ipa user-add** command.

Note that for the command to succeed, you must specify a custom GID. For example, to add a new user with GID 10000:

```
$ ipa user-add jsmith --first=John --last=Smith --noprivate --gid 10000
```

10.7.3. Disabling user private groups globally for all users

You can disable user private groups (UPGs) globally. This prevents the creation of UPGs for all new users. Existing users are unaffected by this change.

Procedure

- Obtain administrator privileges:

```
$ kinit admin
```

- IdM uses the Directory Server Managed Entries Plug-in to manage UPGs. List the instances of the plug-in:

```
$ ipa-managed-entries --list
```

- To ensure IdM does not create UPGs, disable the plug-in instance responsible for managing user private groups:

```
$ ipa-managed-entries -e "UPG Definition" disable  
Disabling Plugin
```



NOTE

To re-enable the **UPG Definition** instance later, use the **ipa-managed-entries -e "UPG Definition" enable** command.

- Restart Directory Server to load the new configuration.

```
$ sudo systemctl restart dirsrv.target
```

To add a user after UPGs have been disabled, you need to specify a GID. For more information, see [Adding a user when user private groups are globally disabled](#)

Verification steps

- To check if UPGs are globally disabled, use the disable command again:

```
$ ipa-managed-entries -e "UPG Definition" disable  
Plugin already disabled
```

10.7.4. Adding a user when user private groups are globally disabled

When user private groups (UPGs) are disabled globally, IdM does not assign a GID to a new user automatically. To successfully add a user, you must assign a GID manually or by using an automember rule. For details on why this is required, see [Section 10.7.1, “Users without a user private group”](#).

Prerequisites

- UPGs must be disabled globally for all users. For more information, see [Disabling user private groups globally for all users](#)

Procedure

- To make sure adding a new user succeeds when creating UPGs is disabled, choose one of the following:
 - Specify a custom GID when adding a new user. The GID does not have to correspond to an already existing user group.
For example, when adding a user from the command line, add the **--gid** option to the **ipa user-add** command.
 - Use an automember rule to add the user to an existing group with a GID.

10.8. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE IDM CLI

This section describes how to add users or groups as member managers to an IdM user group using the IdM CLI. Member managers can add users or groups to IdM user groups but cannot change the attributes of a group.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- You must have the name of the user or group you are adding as member managers and the name of the group you want them to manage.

Procedure

- Add a user as a member manager to an IdM user group by using the **ipa group-add-member-manager** command.
For example, to add the user **test** as a member manager of **group_a**:

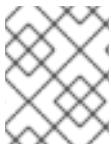
```
$ ipa group-add-member-manager group_a --users=test
Group name: group_a
GID: 1133400009
Membership managed by users: test
-----
Number of members added 1
-----
```

User **test** can now manage members of **group_a**.

- Add a group as a member manager to an IdM user group by using the **ipa group-add-member-manager** command.
For example, to add the group **group_admins** as a member manager of **group_a**:

```
$ ipa group-add-member-manager group_a --groups=group_admins
Group name: group_a
GID: 1133400009
Membership managed by groups: group_admins
Membership managed by users: test
-----
Number of members added 1
-----
```

Group **group_admins** can now manage members of **group_a**.



NOTE

After you add a member manager to a user group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Using the **ipa group-show** command to verify the user and group were added as member managers.

```
$ ipa group-show group_a
Group name: group_a
GID: 1133400009
Membership managed by groups: group_admins
Membership managed by users: test
```

Additional resources

- See **ipa group-add-member-manager --help** for more details.

10.9. VIEWING GROUP MEMBERS USING IDM CLI

This section describes how to view members of a group using IdM CLI. You can view both direct and indirect group members. For more information, see [Direct and indirect group members](#).

Procedure:

- To list members of a group, use the **ipa group-show group_name** command. For example:

```
$ ipa group-show group_a
...
Member users: user_a
Member groups: group_b
Indirect Member users: user_b
```



NOTE

The list of indirect members does not include external users from trusted Active Directory domains. The Active Directory trust user objects are not visible in the Identity Management interface because they do not exist as LDAP objects within Identity Management.

10.10. REMOVING A MEMBER FROM A USER GROUP USING IDM CLI

This section describes how to remove a member from a user group using IdM CLI.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. *Optional.* Use the **ipa group-show** command to confirm that the group includes the member you want to remove.
2. Remove a member from a user group by using the **ipa group-remove-member** command. Specify members to remove using these options:
 - **--users** removes an IdM user
 - **--external** removes a user that exists outside the IdM domain, in the format of **DOMAIN\user_name** or **user_name@domain**
 - **--groups** removes an IdM user group

For example, to remove *user1*, *user2*, and *group1* from a group called *group_name*:

```
$ ipa group-remove-member group_name --users=user1 --users=user2 --groups=group1
```

10.11. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE IDM CLI

This section describes how to remove users or groups as member managers from an IdM user group using the IdM CLI. Member managers can remove users or groups from IdM user groups but cannot change the attributes of a group.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- You must have the name of the existing member manager user or group you are removing and the name of the group they are managing.

Procedure

- Remove a user as a member manager of an IdM user group by using the **ipa group-remove-member-manager** command.

For example, to remove the user **test** as a member manager of **group_a**:

```
$ ipa group-remove-member-manager group_a --users=test
Group name: group_a
GID: 1133400009
Membership managed by groups: group_admins
-----
Number of members removed 1
-----
```

- User **test** can no longer manage members of **group_a**.
- Remove a group as a member manager of an IdM user group by using the **ipa group-remove-member-manager** command.
For example, to remove the group **group_admins** as a member manager of **group_a**:

```
$ ipa group-remove-member-manager group_a --groups=group_admins
Group name: group_a
GID: 1133400009
-----
Number of members removed 1
-----
```

Group **group_admins** can no longer manage members of **group_a**.



NOTE

After you remove a member manager from a user group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Using the **ipa group-show** command to verify the user and group were removed as member managers.

```
$ ipa group-show group_a
Group name: group_a
GID: 1133400009
```

Additional resources

- See **ipa group-remove-member-manager --help** for more details.

CHAPTER 11. MANAGING USER GROUPS IN IDM WEB UI

This chapter introduces user groups management using the IdM web UI.

A user group is a set of users with common privileges, password policies, and other characteristics.

A user group in Identity Management (IdM) can include:

- IdM users
- other IdM user groups
- external users, which are users that exist outside of IdM

11.1. THE DIFFERENT GROUP TYPES IN IDM

IdM supports the following types of groups:

POSIX groups (the default)

POSIX groups support Linux POSIX attributes for their members. Note that groups that interact with Active Directory cannot use POSIX attributes.

POSIX attributes identify users as separate entities. Examples of POSIX attributes relevant to users include **uidNumber**, a user number (UID), and **gidNumber**, a group number (GID).

Non-POSIX groups

Non-POSIX groups do not support POSIX attributes. For example, these groups do not have a GID defined.

All members of this type of group must belong to the IdM domain.

External groups

Use external groups to add group members that exist in an identity store outside of the IdM domain, such as:

- A local system
- An Active Directory domain
- A directory service

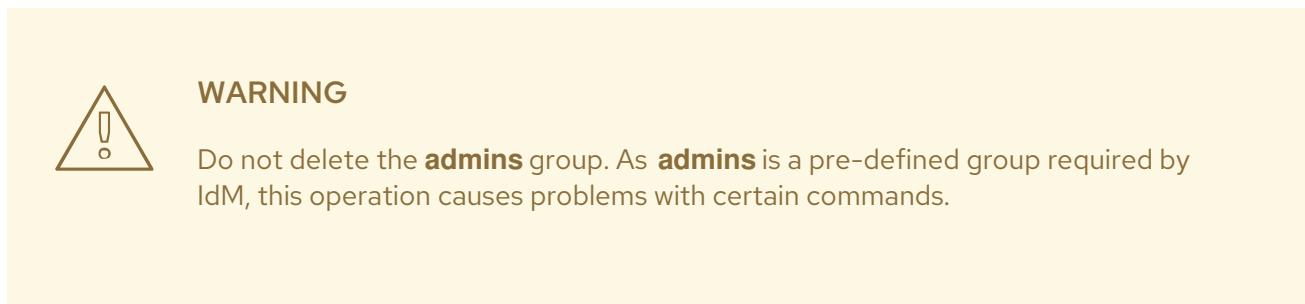
External groups do not support POSIX attributes. For example, these groups do not have a GID defined.

Table 11.1. User groups created by default

Group name	Default group members
ipausers	All IdM users
admins	Users with administrative privileges, including the default admin user
editors	This is a legacy group that no longer has any special privileges

Group name	Default group members
trust admins	Users with privileges to manage the Active Directory trusts

When you add a user to a user group, the user gains the privileges and policies associated with the group. For example, to grant administrative privileges to a user, add the user to the **admins** group.



In addition, IdM creates *user private groups* by default whenever a new user is created in IdM. For more information about private groups, see [Adding users without a private group](#).

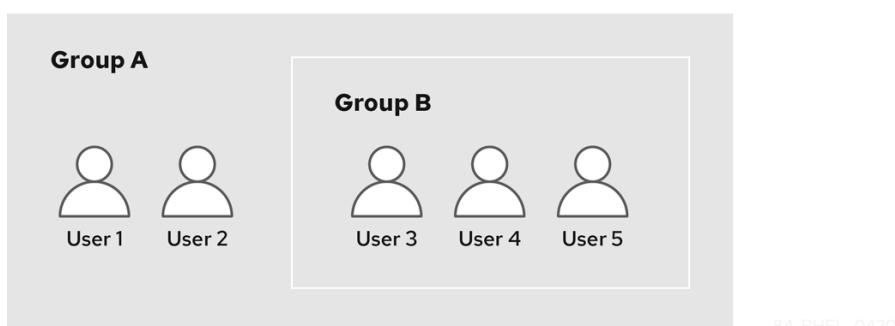
11.2. DIRECT AND INDIRECT GROUP MEMBERS

User group attributes in IdM apply to both direct and indirect members: when group B is a member of group A, all users in group B are considered indirect members of group A.

For example, in the following diagram:

- User 1 and User 2 are *direct members* of group A.
- User 3, User 4, and User 5 are *indirect members* of group A.

Figure 11.1. Direct and Indirect Group Membership



If you set a password policy for user group A, the policy also applies to all users in user group B.

11.3. ADDING A USER GROUP USING IDM WEB UI

This section describes how to add a user group using the IdM Web UI.

Prerequisites

- You are logged in to the IdM Web UI.

Procedure

1. Click **Identity → Groups**, and select **User Groups** in the left sidebar.
2. Click **Add** to start adding the group.
3. Fill out the information about the group. For more information about user group types, see [The different group types in IdM](#).

You can specify a custom GID for the group. If you do this, be careful to avoid ID conflicts. If you do not specify a custom GID, IdM automatically assigns a GID from the available ID range.

The screenshot shows the 'Add user group' dialog box. At the top, it says 'Add user group' and has a close button. Below that, there are fields for 'Group name *' (containing 'group_a'), 'Description' (an empty text area), 'Group Type' (with radio buttons for Non-POSIX, External, and POSIX, where POSIX is selected), and 'GID' (an empty text field). A note at the bottom left says '* Required field'. At the bottom right, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

4. Click **Add** to confirm.

11.4. DELETING A USER GROUP USING IDM WEB UI

This section describes how to delete a user group using the IdM Web UI. Note that deleting a group does not delete the group members from IdM.

Prerequisites

- You are logged in to the IdM Web UI.

Procedure

1. Click **Identity → Groups** and select **User Groups**.
2. Select the group to delete.

3. Click **Delete**.
4. Click **Delete** to confirm.

11.5. ADDING A MEMBER TO A USER GROUP USING IDM WEB UI

You can add both users and user groups as members of a user group. For more information, see [The different group types in IdM](#) and [Direct and indirect group members](#).

Prerequisites

- You are logged in to the IdM Web UI.

Procedure

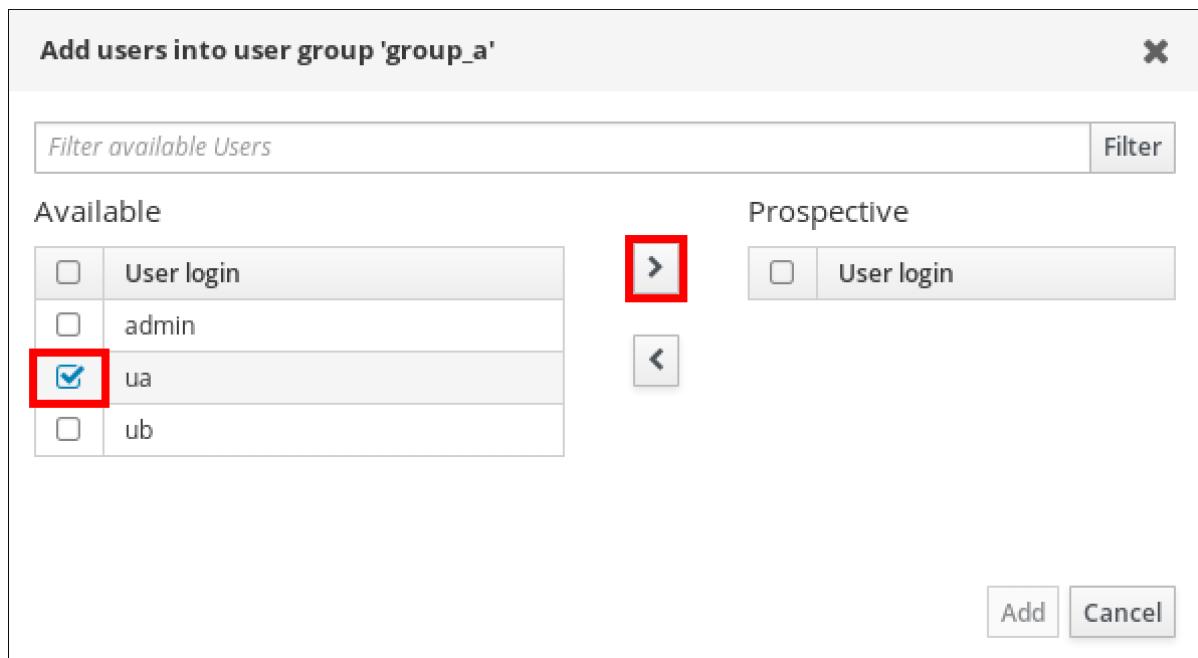
1. Click **Identity → Groups** and select **User Groups** in the left sidebar.
2. Click the name of the group.
3. Select the type of group member you want to add: **Users**, **User Groups**, or **External**.

User Group: group_a

group_a members:

Users	User Groups	Services	External	Settings
<input type="checkbox"/>	User login	UID	Email	

4. Click **Add**.
5. Select the check box next to one or more members you want to add.
6. Click the rightward arrow to move the selected members to the group.



7. Click **Add** to confirm.

11.6. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE WEB UI

This section describes how to add users or groups as member managers to an IdM user group using the Web UI. Member managers can add users or groups to IdM user groups but cannot change the attributes of a group.

Prerequisites

- You are logged in to the IdM Web UI.
- You must have the name of the user or group you are adding as member managers and the name of the group you want them to manage.

Procedure

1. Click **Identity → Groups** and select **User Groups** in the left sidebar.
2. Click the name of the group.
3. Select the type of group member manager you want to add: **Users** or **User Groups**.

User Group: group_a

group_a members:

Users	User Groups	Services	External	User ID overrides
-------	-------------	----------	----------	-------------------

group_a member managers:

User Groups	Users
-------------	-------

Buttons: Refresh, Delete, Add

4. Click **Add**.
5. Select the check box next to one or more members you want to add.
6. Click the rightward arrow to move the selected members to the group.

Add users as member managers for user group 'group_a'

Available

<input type="checkbox"/>	User login
<input type="checkbox"/>	admin
<input checked="" type="checkbox"/>	test1
<input type="checkbox"/>	test2
<input type="checkbox"/>	test_user
<input type="checkbox"/>	test_user2
<input type="checkbox"/>	tuser3

Prospective

<input type="checkbox"/>	User login
--------------------------	------------

Buttons: > (highlighted with a red box), <, Filter, Add, Cancel

7. Click **Add** to confirm.



NOTE

After you add a member manager to a user group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Verify the newly added user or user group has been added to the member manager list of users or user groups:

User Group: project

project members:

Users	User Groups	Services
-------	-------------	----------

project member managers:

User Groups (1)	Users
Refresh	Delete

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

Additional resources

- See **ipa group-add-member-manager --help** for more information.

11.7. VIEWING GROUP MEMBERS USING IDM WEB UI

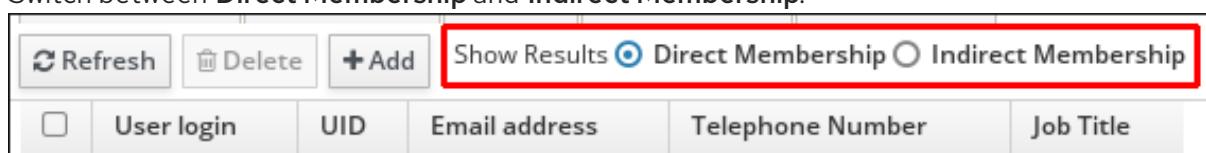
This section describes how to view members of a group using the IdM Web UI. You can view both direct and indirect group members. For more information, see [Direct and indirect group members](#).

Prerequisites

- You are logged in to the IdM Web UI.

Procedure

1. Select Identity → Groups.
2. Select User Groups in the left sidebar.
3. Click the name of the group you want to view.
4. Switch between Direct Membership and Indirect Membership.



11.8. REMOVING A MEMBER FROM A USER GROUP USING IDM WEB UI

This section describes how to remove a member from a user group using the IdM Web UI.

Prerequisites

- You are logged in to the IdM Web UI.

Procedure

1. Click **Identity → Groups** and select **User Groups** in the left sidebar.
2. Click the name of the group.
3. Select the type of group member you want to remove: **Users**, **User Groups**, or **External**.

The screenshot shows the 'User Group: group_a' page. At the top, it says 'group_a members'. Below that is a toolbar with 'Refresh', 'Delete', and 'Add' buttons. The main area has tabs for 'Users' (which is selected and highlighted with a red box), 'User Groups', 'Services', 'External' (also highlighted with a red box), and 'Settings'. At the bottom, there are three columns: 'User login' (with a checkbox icon), 'UID', and 'Email'.

4. Select the check box next to the member you want to remove.
5. Click **Delete**.
6. Click **Delete** to confirm.

11.9. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE WEB UI

This section describes how to remove users or groups as member managers from an IdM user group using the Web UI. Member managers can remove users or groups from IdM user groups but cannot change the attributes of a group.

Prerequisites

- You are logged in to the IdM Web UI.
- You must have the name of the existing member manager user or group you are removing and the name of the group they are managing.

Procedure

1. Click **Identity → Groups** and select **User Groups** in the left sidebar.
2. Click the name of the group.
3. Select the type of member manager you want to remove: **Users** or **User Groups**.

User Group: group_a

group_a members:

Users	User Groups	Services	External	User ID overrides
-------	-------------	----------	----------	-------------------

group_a member managers:

User Groups	Users
-------------	-------

Refresh **Delete** **Add**

4. Select the check box next to the member manager you want to remove.
5. Click **Delete**.
6. Click **Delete** to confirm.



NOTE

After you remove a member manager from a user group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Verify the user or user group has been removed from the member manager list of users or user groups:

User Group: project

project members:

Users	User Groups	Services
-------	-------------	----------

project member managers:

User Groups	Users (1)
-------------	-----------

Refresh **Delete** **Add**

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

No entries.

Additional resources

- See **ipa group-add-member-manager --help** for more details.

CHAPTER 12. MANAGING USER GROUPS USING ANSIBLE PLAYBOOKS

This section introduces user group management using Ansible playbooks.

A user group is a set of users with common privileges, password policies, and other characteristics.

A user group in Identity Management (IdM) can include:

- IdM users
- other IdM user groups
- external users, which are users that exist outside of IdM

The section includes the following topics:

- [The different group types in IdM](#)
- [Direct and indirect group members](#)
- [Ensuring the presence of IdM groups and group members using Ansible playbooks](#)
- [Ensuring the presence of member managers in IDM user groups using Ansible playbooks](#)
- [Ensuring the absence of member managers in IDM user groups using Ansible playbooks](#)

12.1. THE DIFFERENT GROUP TYPES IN IDM

IdM supports the following types of groups:

POSIX groups (the default)

POSIX groups support Linux POSIX attributes for their members. Note that groups that interact with Active Directory cannot use POSIX attributes.

POSIX attributes identify users as separate entities. Examples of POSIX attributes relevant to users include **uidNumber**, a user number (UID), and **gidNumber**, a group number (GID).

Non-POSIX groups

Non-POSIX groups do not support POSIX attributes. For example, these groups do not have a GID defined.

All members of this type of group must belong to the IdM domain.

External groups

Use external groups to add group members that exist in an identity store outside of the IdM domain, such as:

- A local system
- An Active Directory domain
- A directory service

External groups do not support POSIX attributes. For example, these groups do not have a GID defined.

Table 12.1. User groups created by default

Group name	Default group members
ipausers	All IdM users
admins	Users with administrative privileges, including the default admin user
editors	This is a legacy group that no longer has any special privileges
trust admins	Users with privileges to manage the Active Directory trusts

When you add a user to a user group, the user gains the privileges and policies associated with the group. For example, to grant administrative privileges to a user, add the user to the **admins** group.



WARNING

Do not delete the **admins** group. As **admins** is a pre-defined group required by IdM, this operation causes problems with certain commands.

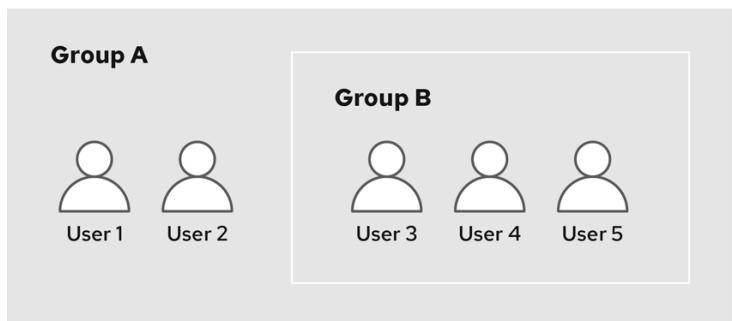
In addition, IdM creates *user private groups* by default whenever a new user is created in IdM. For more information about private groups, see [Adding users without a private group](#).

12.2. DIRECT AND INDIRECT GROUP MEMBERS

User group attributes in IdM apply to both direct and indirect members: when group B is a member of group A, all users in group B are considered indirect members of group A.

For example, in the following diagram:

- User 1 and User 2 are *direct members* of group A.
- User 3, User 4, and User 5 are *indirect members* of group A.

Figure 12.1. Direct and Indirect Group Membership

If you set a password policy for user group A, the policy also applies to all users in user group B.

12.3. ENSURING THE PRESENCE OF IDM GROUPS AND GROUP MEMBERS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the presence of IdM groups and group members - both users and user groups - using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- The users you want to reference in your Ansible playbook exist in IdM. For details on ensuring the presence of users using Ansible, see [Managing user accounts using Ansible playbooks](#).

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary user and group information:

```
---
- name: Playbook to handle groups
  hosts: ipaserver
  become: true

  tasks:
    - name: Create group ops with gid 1234
      ipagroup:
        ipaadmin_password: MySecret123
        name: ops
        gidnumber: 1234

    - name: Create group sysops
      ipagroup:
        ipaadmin_password: MySecret123
        name: sysops
        user:
          - idm_user

    - name: Create group appops
      ipagroup:
        ipaadmin_password: MySecret123
        name: appops

    - name: Add group members sysops and appops to group ops
      ipagroup:
        ipaadmin_password: MySecret123
        name: ops
        group:
          - sysops
          - appops
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file  
path_to_playbooks_directory/add-group-members.yml
```

Verification steps

You can verify if the **ops** group contains **sysops** and **appops** as direct members and **idm_user** as an indirect member by using the **ipa group-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh admin@server.idm.example.com  
Password:  
[admin@server /]$
```

2. Display information about **ops**:

```
ipaserver]$ ipa group-show ops  
Group name: ops  
GID: 1234  
Member groups: sysops, appops  
Indirect Member users: idm_user
```

The **appops** and **sysops** groups - the latter including the **idm_user** user - exist in IdM.

Additional resources

- For more information about ensuring the presence of user groups using Ansible, see the [/usr/share/doc/ansible-freeipa/README-group.md](#) Markdown file.

12.4. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the presence of IdM member managers - both users and user groups - using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- You must have the name of the user or group you are adding as member managers and the name of the group you want them to manage.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]  
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary user and group member management information:

```
---
- name: Playbook to handle membership management
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure user test is present for group_a
      ipagroup:
        ipaadmin_password: MySecret123
        name: group_a
        membermanager_user: test

    - name: Ensure group_admins is present for group_a
      ipagroup:
        ipaadmin_password: MySecret123
        name: group_a
        membermanager_group: group_admins
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/add-member-managers-user-groups.yml
```

Verification steps

You can verify if the `group_a` group contains `test` as a member manager and `group_admins` is a member manager of `group_a` by using the `ipa group-show` command:

1. Log into `ipaserver` as administrator:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about `managergroup1`:

```
[ipaserver]$ ipa group-show group_a
Group name: group_a
GID: 1133400009
Membership managed by groups: group_admins
Membership managed by users: test
```

Additional resources

- See `ipa host-add-member-manager --help`.
- See the `ipa` man page.

12.5. ENSURING THE ABSENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the absence of IdM member managers - both users and user groups - using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- You must have the name of the existing member manager user or group you are removing and the name of the group they are managing.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary user and group member management information:

```
---
- name: Playbook to handle membership management
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure member manager user and group members are absent for group_a
      ipaadmin_password: MySecret123
      name: group_a
      membermanager_user: test
      membermanager_group: group_admins
      action: member
      state: absent
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-member-managers-are-absent.yml
```

Verification steps

You can verify if the **group_a** group does not contain **test** as a member manager and **group_admins** as a member manager of **group_a** by using the **ipa group-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about **group_a**:

```
■
```

```
| ipaserver]$ ipa group-show group_a  
| Group name: group_a  
| GID: 1133400009
```

Additional resources

- See **ipa host-remove-member-manager --help**.
- See the **ipa** man page.

CHAPTER 13. AUTOMATING GROUP MEMBERSHIP USING IDM CLI

Using automatic group membership allows you to assign users and hosts to groups automatically based on their attributes. For example, you can:

- Divide employees' user entries into groups based on the employees' manager, location, or any other attribute.
- Divide hosts based on their class, location, or any other attribute.
- Add all users or all hosts to a single global group.

This chapter covers the following topics:

- Benefits of automatic group membership
- Automember rules
- Adding an automember rule using IdM CLI
- Adding a condition to an automember rule using IdM CLI
- Viewing existing automember rules using IdM CLI
- Deleting an automember rule using IdM CLI
- Removing a condition from an automember rule using IdM CLI
- Applying automember rules to existing entries using IdM CLI
- Configuring a default automember group using IdM CLI

13.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP

Using automatic membership for users allows you to:

- **Reduce the overhead of manually managing group memberships**
You no longer have to assign every user and host to groups manually.
- **Improve consistency in user and host management**
Users and hosts are assigned to groups based on strictly defined and automatically evaluated criteria.
- **Simplify the management of group-based settings**
Various settings are defined for groups and then applied to individual group members, for example **sudo** rules, automount, or access control. Adding users and hosts to groups automatically makes managing these settings easier.

13.2. AUTOMEMBER RULES

When configuring automatic group membership, the administrator defines automember rules. An automember rule applies to a specific user or host target group. It cannot apply to more than one group at a time.

After creating a rule, the administrator adds conditions to it. These specify which users or hosts get included or excluded from the target group:

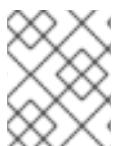
- **Inclusive conditions**

When a user or host entry meets an inclusive condition, it will be included in the target group.

- **Exclusive conditions**

When a user or host entry meets an exclusive condition, it will not be included in the target group.

The conditions are specified as regular expressions in the Perl-compatible regular expressions (PCRE) format. For more information on PCRE, see the [pcresyntax\(3\)](#) man page.



NOTE

IdM evaluates exclusive conditions before inclusive conditions. In case of a conflict, exclusive conditions take precedence over inclusive conditions.

An automember rule applies to every entry created in the future. These entries will be automatically added to the specified target group. If an entry meets the conditions specified in multiple automember rules, it will be added to all the corresponding groups.

Existing entries are **not** affected by the new rule. If you want to change existing entries, see [Applying automember rules to existing entries using IdM CLI](#).

13.3. ADDING AN AUTOMEMBER RULE USING IDM CLI

This section describes adding an automember rule using the IdM CLI. For information about automember rules, see [Automember rules](#).

After adding an automember rule, you can add conditions to it using the procedure described in [Adding a condition to an automember rule](#).



NOTE

Existing entries are **not** affected by the new rule. If you want to change existing entries, see [Applying automember rules to existing entries using IdM CLI](#).

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- The target group of the new rule must exist in IdM.

Procedure

1. Enter the **ipa automember-add** command to add an automember rule.
2. When prompted, specify:
 - **Automember rule.** This is the target group name.
 - **Grouping Type.** This specifies whether the rule targets a user group or a host group. To target a user group, enter **group**. To target a host group, enter **hostgroup**.

For example, to add an automember rule for a user group named **user_group**:

```
$ ipa automember-add
Automember Rule: user_group
Grouping Type: group
-----
Added automember rule "user_group"
-----
Automember Rule: user_group
```

Verification steps

- You can display existing automember rules and conditions in IdM using [Viewing existing automember rules using IdM CLI](#).

13.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM CLI

This section describes how to add a condition to an automember rule using the IdM CLI. For information about automember rules, see [Automember rules](#).

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- The target rule must exist in IdM. For details, see [Adding an automember rule using IdM CLI](#).

Procedure

1. Define one or more inclusive or exclusive conditions using the **ipa automember-add-condition** command.
2. When prompted, specify:
 - **Automember rule**. This is the target rule name. See [Automember rules](#) for details.
 - **Attribute Key**. This specifies the entry attribute to which the filter will apply. For example, **uid** for users.
 - **Grouping Type**. This specifies whether the rule targets a user group or a host group. To target a user group, enter **group**. To target a host group, enter **hostgroup**.
 - **Inclusive regex** and **Exclusive regex**. These specify one or more conditions as regular expressions. If you only want to specify one condition, press **Enter** when prompted for the other.

For example, the following condition targets all users with any value (.*) in their user login attribute (**uid**).

```
$ ipa automember-add-condition
Automember Rule: user_group
Attribute Key: uid
Grouping Type: group
[Inclusive Regex]: .*
```

[Exclusive Regex]:

Added condition(s) to "user_group"

Automember Rule: user_group

Inclusive Regex: uid=.*

Number of conditions added 1

As another example, you can use an automembership rule to target all Windows users synchronized from Active Directory (AD). To achieve this, create a condition that targets all users with **ntUser** in their **objectClass** attribute, which is shared by all AD users:

\$ **ipa automember-add-condition**

Automember Rule: **ad_users**

Attribute Key: **objectclass**

Grouping Type: **group**

[Inclusive Regex]: **ntUser**

[Exclusive Regex]:

Added condition(s) to "ad_users"

Automember Rule: ad_users

Inclusive Regex: objectclass=ntUser

Number of conditions added 1

Verification steps

- You can display existing automember rules and conditions in IdM using [Viewing existing automember rules using IdM CLI](#).

13.5. VIEWING EXISTING AUTOMEMBER RULES USING IDM CLI

This section describes how to view existing automember rules using the IdM CLI.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#) .

Procedure

1. Enter the **ipa automember-find** command.
2. When prompted, specify the **Grouping type**:
 - To target a user group, enter **group**.
 - To target a host group, enter **hostgroup**.
For example:

\$ ipa automember-find

```
Grouping Type: group
-----
1 rules matched
-----
Automember Rule: user_group
Inclusive Regex: uid=.*
-----
Number of entries returned 1
-----
```

13.6. DELETING AN AUTOMEMBER RULE USING IDM CLI

This section describes how to delete an automember rule using the IdM CLI.

Deleting an automember rule also deletes all conditions associated with the rule. To remove only specific conditions from a rule, see [Removing a condition from an automember rule using IdM CLI](#).

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Enter the **ipa automember-del** command.
2. When prompted, specify:
 - **Automember rule.** This is the rule you want to delete.
 - **Grouping rule.** This specifies whether the rule you want to delete is for a user group or a host group. Enter **group** or **hostgroup**.

13.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM CLI

This section describes how to remove a specific condition from an automember rule.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Enter the **ipa automember-remove-condition** command.
2. When prompted, specify:
 - **Automember rule.** This is the name of the rule from which you want to remove a condition.
 - **Attribute Key.** This is the target entry attribute. For example, **uid** for users.
 - **Grouping Type.** This specifies whether the condition you want to delete is for a user group or a host group. Enter **group** or **hostgroup**.

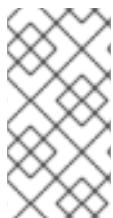
- **Inclusive regex** and **Exclusive regex** These specify the conditions you want to remove. If you only want to specify one condition, press **Enter** when prompted for the other. For example:

```
$ ipa automember-remove-condition
Automember Rule: user_group
Attribute Key: uid
Grouping Type: group
[Inclusive Regex]: .*
[Exclusive Regex]:
-----
Removed condition(s) from "user_group"
-----
Automember Rule: user_group
-----
Number of conditions removed 1
-----
```

13.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM CLI

Automember rules apply automatically to user and host entries created after the rules were added. They are not applied retroactively to entries that existed before the rules were added.

To apply automember rules to previously added entries, you have to manually rebuild automatic membership. Rebuilding automatic membership re-evaluates all existing automember rules and applies them either to all user or hosts entries, or to specific entries.



NOTE

Rebuilding automatic membership **does not** remove user or host entries from groups, even if the entries no longer match the group's inclusive conditions. To remove them manually, see [Removing a member from a user group using IdM CLI](#) or [Removing IdM host group members using the CLI](#).

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#) .

Procedure

- To rebuild automatic membership, enter the **ipa automember-rebuild** command. Use the following options to specify the entries to target:
 - To rebuild automatic membership for all users, use the **--type=group** option:

```
$ ipa automember-rebuild --type=group
-----
```

```
Automember rebuild task finished. Processed (9) entries.
-----
```

- To rebuild automatic membership for all hosts, use the **--type=hostgroup** option.

- To rebuild automatic membership for a specified user or users, use the **--users=*target_user*** option:

```
$ ipa automember-rebuild --users=target_user1 --users=target_user2
```

Automember rebuild task finished. Processed (2) entries.

- To rebuild automatic membership for a specified host or hosts, use the **--hosts=*client.idm.example.com*** option.

13.9. CONFIGURING A DEFAULT AUTOMEMBER GROUP USING IDM CLI

When you configure a default automember group, new user or host entries that do not match any automember rule are automatically added to this default group.

Prerequisites

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- The target group you want to set as default exists in IdM.

Procedure

1. Enter the **ipa automember-default-group-set** command to configure a default automember group.
 2. When prompted, specify:
 - **Default (fallback) Group**, which specifies the target group name.
 - **Grouping Type**, which specifies whether the target is a user group or a host group. To target a user group, enter **group**. To target a host group, enter **hostgroup**.
- For example:

```
$ ipa automember-default-group-set
Default (fallback) Group: default_user_group
Grouping Type: group
```

Set default (fallback) group for automember "default_user_group"

```
Default (fallback) Group:
cn=default_user_group,cn=groups,cn=accounts,dc=example,dc=com
```



NOTE

To remove the current default automember group, enter the **ipa automember-default-group-remove** command.

Verification steps

- To verify that the group is set correctly, enter the **ipa automember-default-group-show** command. The command displays the current default automember group. For example:

```
$ ipa automember-default-group-show
Grouping Type: group
  Default (fallback) Group:
    cn=default_user_group,cn=groups,cn=accounts,dc=example,dc=com
```

CHAPTER 14. AUTOMATING GROUP MEMBERSHIP USING IDM WEB UI

Using automatic group membership enables you to assign users and hosts to groups automatically based on their attributes. For example, you can:

- Divide employees' user entries into groups based on the employees' manager, location, or any other attribute.
- Divide hosts based on their class, location, or any other attribute.
- Add all users or all hosts to a single global group.

This chapter covers the following topics:

- Benefits of automatic group membership
- Automember rules
- Adding an automember rule using IdM Web UI
- Adding a condition to an automember rule using IdM Web UI
- Viewing existing automember rules and conditions using IdM Web UI
- Deleting an automember rule using IdM Web UI
- Removing a condition from an automember rule using IdM Web UI
- Applying automember rules to existing entries using IdM Web UI
- Configuring a default user group using IdM Web UI
- Configuring a default host group using IdM Web UI

14.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP

Using automatic membership for users allows you to:

- **Reduce the overhead of manually managing group memberships**
You no longer have to assign every user and host to groups manually.
- **Improve consistency in user and host management**
Users and hosts are assigned to groups based on strictly defined and automatically evaluated criteria.
- **Simplify the management of group-based settings**
Various settings are defined for groups and then applied to individual group members, for example **sudo** rules, automount, or access control. Adding users and hosts to groups automatically makes managing these settings easier.

14.2. AUTOMEMBER RULES

When configuring automatic group membership, the administrator defines automember rules. An automember rule applies to a specific user or host target group. It cannot apply to more than one group at a time.

After creating a rule, the administrator adds conditions to it. These specify which users or hosts get included or excluded from the target group:

- **Inclusive conditions**

When a user or host entry meets an inclusive condition, it will be included in the target group.

- **Exclusive conditions**

When a user or host entry meets an exclusive condition, it will not be included in the target group.

The conditions are specified as regular expressions in the Perl-compatible regular expressions (PCRE) format. For more information on PCRE, see the [pcresyntax\(3\)](#) man page.



NOTE

IdM evaluates exclusive conditions before inclusive conditions. In case of a conflict, exclusive conditions take precedence over inclusive conditions.

An automember rule applies to every entry created in the future. These entries will be automatically added to the specified target group. If an entry meets the conditions specified in multiple automember rules, it will be added to all the corresponding groups.

Existing entries are **not** affected by the new rule. If you want to change existing entries, see [Applying automember rules to existing entries using IdM Web UI](#).

14.3. ADDING AN AUTOMEMBER RULE USING IDM WEB UI

This section describes adding an automember rule using the IdM Web UI. For information about automember rules, see [Automember rules](#).



NOTE

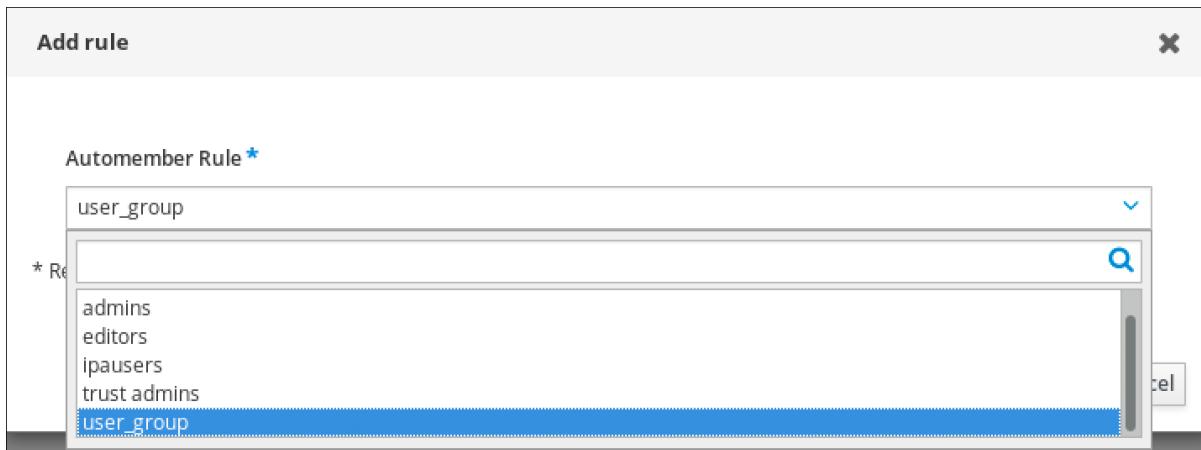
Existing entries are **not** affected by the new rule. If you want to change existing entries, see [Applying automember rules to existing entries using IdM Web UI](#).

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.
- The target group of the new rule exists in IdM.

Procedure

1. Click **Identity → Automember**, and select either **User group rules** or **Host group rules**.
2. Click **Add**.
3. In the **Automember rule** field, select the group to which the rule will apply. This is the target group name.



4. Click **Add** to confirm.
5. Optional: You can add conditions to the new rule using the procedure described in [Adding a condition to an automember rule using IdM Web UI](#).

14.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM WEB UI

This section describes how to add a condition to an automember rule using the IdM Web UI. For information about automember rules, see [Automember rules](#).

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.
- The target rule exists in IdM.

Procedure

1. Click **Identity** → **Automember**, and select either **User group rules** or **Host group rules**.
2. Click on the rule to which you want to add a condition.
3. In the **Inclusive** or **Exclusive** sections, click **Add**.

User group rule: user_group

Refresh Revert Save

General

Automember Rule
user_group

Description

Inclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete	<input style="background-color: red; color: white; border: 2px solid red; padding: 2px 5px;" type="button"/> Add
<input type="checkbox"/>	uid	.*		

Exclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete	<input style="background-color: red; color: white; border: 2px solid red; padding: 2px 5px;" type="button"/> Add
--------------------------	-----------	------------	-------------------------------	------------------------------------------------------------------------------------------------------------------

4. In the **Attribute** field, select the required attribute, for example `uid`.
5. In the **Expression** field, define a regular expression.
6. Click **Add**.
For example, the following condition targets all users with any value (.) in their user ID (uid) attribute.

Add Condition into automember

Attribute	<input type="text" value="uid"/>
Expression *	<input type="text" value=".*"/>
* Required field	
<input type="button" value="Add"/> <input type="button" value="Add and Add Another"/> <input type="button" value="Cancel"/>	

14.5. VIEWING EXISTING AUTOMEMBER RULES AND CONDITIONS USING IDM WEB UI

This section describes how to view existing automember rules and conditions using the IdM Web UI.

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

Procedure

1. Click **Identity → Automember**, and select either **User group rules** or **Host group rules** to view the respective automember rules.
2. Optional: Click on a rule to see the conditions for that rule in the **Inclusive** or **Exclusive** sections.

User group rule: user_group

Refresh Revert Save

General

Automember Rule
user_group

Description

Inclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete	<input type="button"/> Add
<input type="checkbox"/>	uid	.*		

Exclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete	<input type="button"/> Add

14.6. DELETING AN AUTOMEMBER RULE USING IDM WEB UI

This section describes how to delete an automember rule using the IdM Web UI.

Deleting an automember rule also deletes all conditions associated with the rule. To remove only specific conditions from a rule, see [Removing a condition from an automember rule using IdM Web UI](#).

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

Procedure

1. Click **Identity → Automember**, and select either **User group rules** or **Host group rules** to view the respective automember rules.
2. Select the check box next to the rule you want to remove.
3. Click **Delete**.

User group rules

<input type="checkbox"/>	Automember Rule	Description
<input checked="" type="checkbox"/>	user_group	

1 rules matched

4. Click **Delete** to confirm.

14.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM WEB UI

This section describes how to remove a specific condition from an automember rule using the IdM Web UI.

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

Procedure

1. Click **Identity → Automember**, and select either **User group rules** or **Host group rules** to view the respective automember rules.
2. Click on a rule to see the conditions for that rule in the **Inclusive** or **Exclusive** sections.
3. Select the check box next to the conditions you want to remove.
4. Click **Delete**.

User group rule: user_group

Refresh Revert Save

General

Automember Rule
user_group

Description

Inclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete <input type="button"/> Add
<input checked="" type="checkbox"/>	uid	.*	<input type="button"/> Delete <input type="button"/> Add

Exclusive

<input type="checkbox"/>	Attribute	Expression	<input type="button"/> Delete <input type="button"/> Add
			<input type="button"/> Delete <input type="button"/> Add

- Click **Delete** to confirm.

14.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM WEB UI

Automember rules apply automatically to user and host entries created after the rules were added. They are not applied retroactively to entries that existed before the rules were added.

To apply automember rules to previously added entries, you have to manually rebuild automatic membership. Rebuilding automatic membership re-evaluates all existing automember rules and applies them either to all user or hosts entries, or to specific entries.



NOTE

Rebuilding automatic membership **does not** remove user or host entries from groups, even if the entries no longer match the group's inclusive conditions. To remove them manually, see [Removing a member from a user group using IdM Web UI](#) or [Removing host group members in the IdM Web UI](#).

14.8.1. Rebuilding automatic membership for all users or hosts

This section describes how to rebuild automatic membership for all user or host entries.

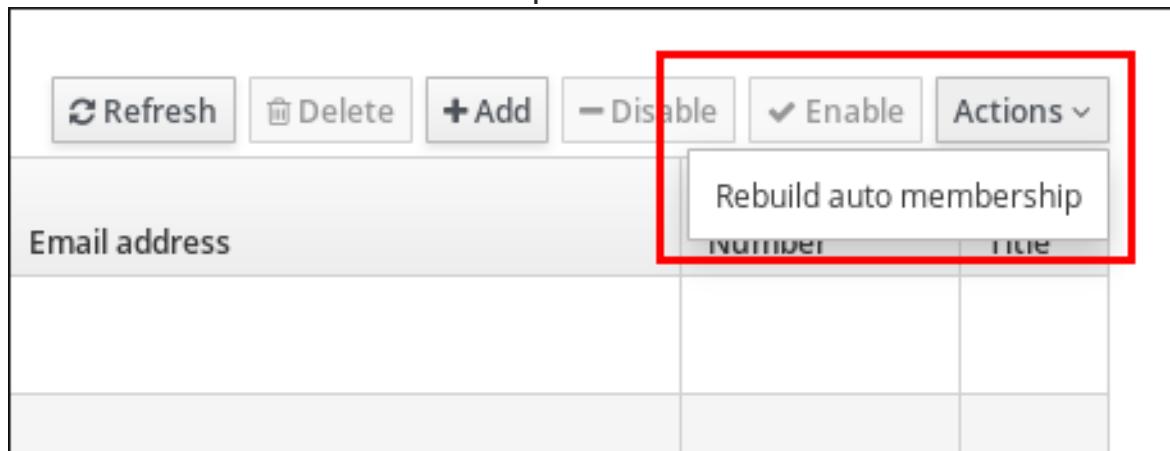
Prerequisites

- You are logged in to the IdM Web UI.

- You must be a member of the **admins** group.

Procedure

1. Select Identity → Users or Hosts.
2. Click Actions → Rebuild auto membership.



14.8.2. Rebuilding automatic membership for a single user or host only

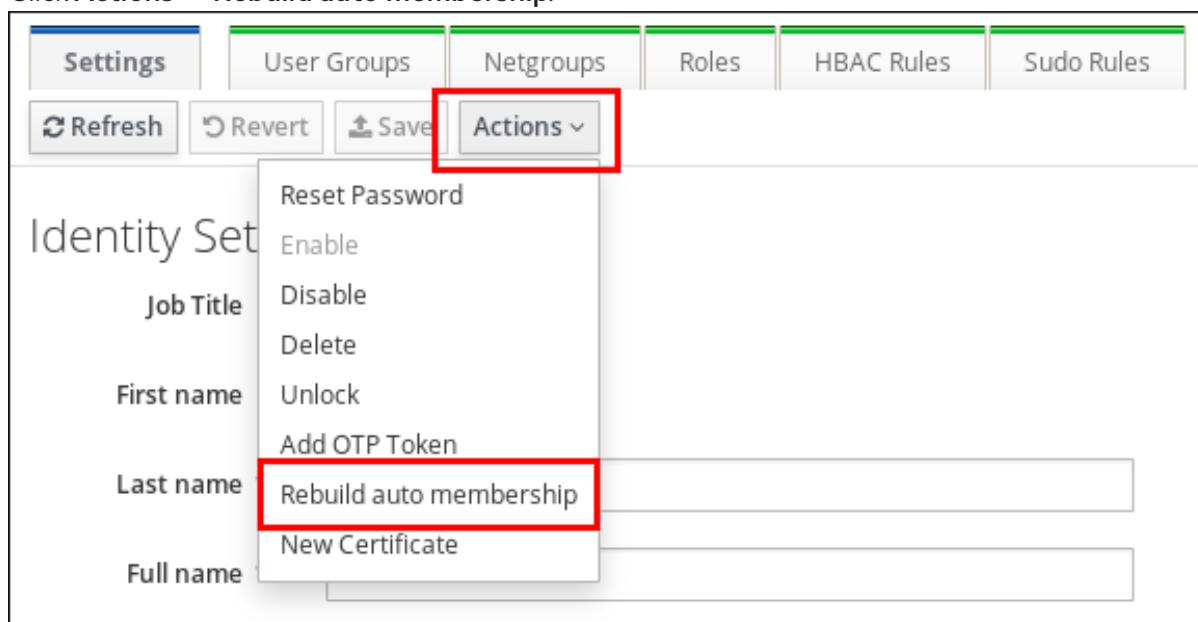
This section describes how to rebuild automatic membership for a specific user or host entry.

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

Procedure

1. Select Identity → Users or Hosts.
2. Click on the required user or host name.
3. Click Actions → Rebuild auto membership.



14.9. CONFIGURING A DEFAULT USER GROUP USING IDM WEB UI

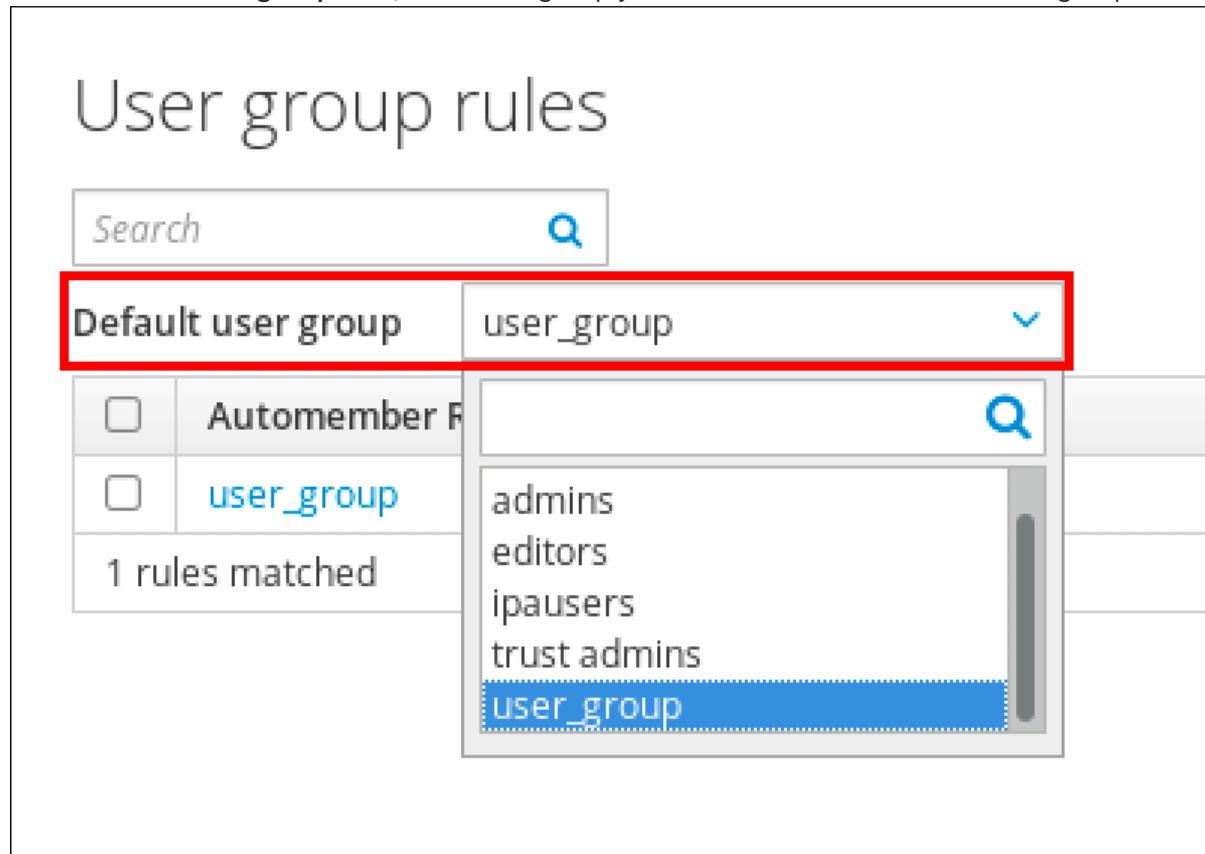
When you configure a default user group, new user entries that do not match any automember rule are automatically added to this default group.

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.
- The target user group you want to set as default exists in IdM.

Procedure

1. Click **Identity → Automember**, and select **User group rules**.
2. In the **Default user group** field, select the group you want to set as the default user group.



14.10. CONFIGURING A DEFAULT HOST GROUP USING IDM WEB UI

When you configure a default host group, new host entries that do not match any automember rule are automatically added to this default group.

Prerequisites

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.
- The target host group you want to set as default exists in IdM.

Procedure

1. Click **Identity** → **Automember**, and select **Host group rules**.
2. In the **Default host group** field, select the group you want to set as the default host group.

The screenshot shows the 'Host group rules' configuration page. At the top, there is a search bar with a magnifying glass icon. Below it, a dropdown menu is open under the heading 'Default host group'. The dropdown contains two items: 'host_group' and 'ipaservers'. The item 'host_group' is highlighted with a blue selection bar. To the left of the dropdown, there is a checkbox labeled 'Automember R' and a message '0 rules matched'.

CHAPTER 15. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM CLI

Delegation is one of the access control methods in IdM, along with self-service rules and role-based access control (RBAC). You can use delegation to assign permissions to one group of users to manage entries for another group of users.

This section covers the following topics:

- [Delegation rules](#)
- [Creating a delegation rule using IdM CLI](#)
- [Viewing existing delegation rules using IdM CLI](#)
- [Modifying a delegation rule using IdM CLI](#)
- [Deleting a delegation rule using IdM CLI](#)

15.1. DELEGATION RULES

You can delegate permissions to user groups to manage users by creating **delegation rules**.

Delegation rules allow a specific user group to perform write (edit) operations on specific attributes for users in another user group. This form of access control rule is limited to editing the values of a subset of attributes you specify in a delegation rule; it does not grant the ability to add or remove whole entries or control over unspecified attributes.

Delegation rules grant permissions to existing user groups in IdM. You can use delegation to, for example, allow the **managers** user group to manage selected attributes of users in the **employees** user group.

15.2. CREATING A DELEGATION RULE USING IDM CLI

This section describes how to create a delegation rule using the IdM CLI.

Prerequisites

- You are logged in as a member of the **admins** group.

Procedure

- Enter the **ipa delegation-add** command. Specify the following options:
 - **--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.
 - **--permissions**: whether users will have the right to view the given attributes (*read*) and add or change the given attributes (*write*). If you do not specify permissions, only the *write* permission will be added.
 - **--attrs**: the attributes which users in the member group are allowed to view or edit.

For example:

```
$ ipa delegation-add "basic manager attributes" --permissions=read --permissions=write --
attrs=businesscategory --attrs=departmentnumber --attrs=employeetype --
attrs=employeeNumber --group=managers --membergroup=employees
-----
Added delegation "basic manager attributes"
-----
Delegation name: basic manager attributes
Permissions: read, write
Attributes: businesscategory, departmentnumber, employeetype, employeeNumber
Member user group: employees
User group: managers
```

15.3. VIEWING EXISTING DELEGATION RULES USING IDM CLI

This section describes how to view existing delegation rules using the IdM CLI.

Prerequisites

- You are logged in as a member of the **admins** group.

Procedure

- Enter the **ipa delegation-find** command:

```
$ ipa delegation-find
-----
1 delegation matched
-----
Delegation name: basic manager attributes
Permissions: read, write
Attributes: businesscategory, departmentnumber, employeeNumber, employeetype
Member user group: employees
User group: managers
-----
Number of entries returned 1
```

15.4. MODIFYING A DELEGATION RULE USING IDM CLI

This section describes how to modify an existing delegation rule using the IdM CLI.



IMPORTANT

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. This also applies to the **--permissions** option.

Prerequisites

- You are logged in as a member of the **admins** group.

Procedure

- Enter the **ipa delegation-mod** command with the desired changes. For example, to add the **displayname** attribute to the **basic manager attributes** example rule:

```
$ ipa delegation-mod "basic manager attributes" --attrs=businesscategory --
--attrs=departmentnumber --attrs=employeetype --attrs=employeeNumber --
--attrs=displayname
```

```
-----  
Modified delegation "basic manager attributes"
```

```
Delegation name: basic manager attributes  
Permissions: read, write  
Attributes: businesscategory, departmentnumber, employeetype, employeeNumber,  
displayname  
Member user group: employees  
User group: managers
```

15.5. DELETING A DELEGATION RULE USING IDM CLI

This section describes how to delete an existing delegation rule using the IdM CLI.

Prerequisites

- You are logged in as a member of the **admins** group.

Procedure

- Enter the **ipa delegation-del** command.
- When prompted, enter the name of the delegation rule you want to delete:

```
$ ipa delegation-del  
Delegation name: basic manager attributes
```

```
-----  
Deleted delegation "basic manager attributes"
```

CHAPTER 16. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM WEBUI

Delegation is one of the access control methods in IdM, along with self-service rules and role-based access control (RBAC). You can use delegation to assign permissions to one group of users to manage entries for another group of users.

This section covers the following topics:

- [Delegation rules](#)
- [Creating a delegation rule using IdM WebUI](#)
- [Viewing existing delegation rules using IdM WebUI](#)
- [Modifying a delegation rule using IdM WebUI](#)
- [Deleting a delegation rule using IdM WebUI](#)

16.1. DELEGATION RULES

You can delegate permissions to user groups to manage users by creating **delegation rules**.

Delegation rules allow a specific user group to perform write (edit) operations on specific attributes for users in another user group. This form of access control rule is limited to editing the values of a subset of attributes you specify in a delegation rule; it does not grant the ability to add or remove whole entries or control over unspecified attributes.

Delegation rules grant permissions to existing user groups in IdM. You can use delegation to, for example, allow the **managers** user group to manage selected attributes of users in the **employees** user group.

16.2. CREATING A DELEGATION RULE USING IDM WEBUI

This section describes how to create a delegation rule using the IdM WebUI.

Prerequisites

- You are logged in to the IdM Web UI as a member of the **admins** group.

Procedure

1. From the **IPA Server** menu, click **Role-Based Access Control → Delegations**.
2. Click **Add**.

The screenshot shows the IPA Server interface with the 'Identity' tab selected. Under 'Identity', the 'Role-Based Access Control' section is active, with 'Delegations' highlighted. In the top right corner of the main content area, there are three buttons: 'Refresh', 'Delete', and '+ Add'. The '+ Add' button is specifically highlighted with a red box.

3. In the **Add delegation** window, do the following:

- a. Name the new delegation rule.
- b. Set the permissions by selecting the check boxes that indicate whether users will have the right to view the given attributes (*read*) and add or change the given attributes (*write*).
- c. In the User group drop-down menu, select the group *who is being granted permissions* to view or edit the entries of users in the member group.
- d. In the **Member user group** drop-down menu, select the group *whose entries can be edited* by members of the delegation group.
- e. In the attributes box, select the check boxes by the attributes to which you want to grant permissions.

Add delegation

Delegation name *	basic manager attributes																																						
Permissions	<input checked="" type="checkbox"/> read <input checked="" type="checkbox"/> write																																						
User group *	managers																																						
Member user group	employees																																						
Attributes *	<input type="button" value="Filter"/> <input type="button" value="Search"/> <input type="button" value="Add"/> <table border="1"> <tbody> <tr><td><input type="checkbox"/> audio</td><td><input checked="" type="checkbox"/> businesscategory</td></tr> <tr><td><input type="checkbox"/> carlicense</td><td><input type="checkbox"/> cn</td></tr> <tr><td><input checked="" type="checkbox"/> departmentnumber</td><td><input type="checkbox"/> description</td></tr> <tr><td><input type="checkbox"/> destinationindicator</td><td><input type="checkbox"/> displayname</td></tr> <tr><td><input checked="" type="checkbox"/> employeenumber</td><td><input checked="" type="checkbox"/> employeetype</td></tr> <tr><td><input type="checkbox"/> facsimiletelephonenumber</td><td><input type="checkbox"/> gecos</td></tr> <tr><td><input type="checkbox"/> gidnumber</td><td><input type="checkbox"/> givenname</td></tr> <tr><td><input type="checkbox"/> homedirectory</td><td><input type="checkbox"/> homephone</td></tr> <tr><td><input type="checkbox"/> homepostaladdress</td><td><input type="checkbox"/> inetuserhttpurl</td></tr> <tr><td><input type="checkbox"/> inetuserstatus</td><td><input type="checkbox"/> initials</td></tr> <tr><td><input type="checkbox"/> internationalisdnnumber</td><td><input type="checkbox"/> ipacertmapdata</td></tr> <tr><td><input type="checkbox"/> ipakrbauthzdata</td><td><input type="checkbox"/> ipanhash</td></tr> <tr><td><input type="checkbox"/> ipanthomedirectory</td><td><input type="checkbox"/> ipanthomedirectorydrive</td></tr> <tr><td><input type="checkbox"/> ipantlogonscript</td><td><input type="checkbox"/> ipantprofilepath</td></tr> <tr><td><input type="checkbox"/> ipantsecurityidentifier</td><td><input type="checkbox"/> ipasshpubkey</td></tr> <tr><td><input type="checkbox"/> ipatokenradiusconfiglink</td><td><input type="checkbox"/> ipatokenradiususername</td></tr> <tr><td><input type="checkbox"/> ipauniqueid</td><td><input type="checkbox"/> ipauserauthtype</td></tr> <tr><td><input type="checkbox"/> jpegphoto</td><td><input type="checkbox"/> krballowedtodelegate</td></tr> <tr><td><input type="checkbox"/> krbcanonicalname</td><td><input type="checkbox"/> krbextradata</td></tr> </tbody> </table>	<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory	<input type="checkbox"/> carlicense	<input type="checkbox"/> cn	<input checked="" type="checkbox"/> departmentnumber	<input type="checkbox"/> description	<input type="checkbox"/> destinationindicator	<input type="checkbox"/> displayname	<input checked="" type="checkbox"/> employeenumber	<input checked="" type="checkbox"/> employeetype	<input type="checkbox"/> facsimiletelephonenumber	<input type="checkbox"/> gecos	<input type="checkbox"/> gidnumber	<input type="checkbox"/> givenname	<input type="checkbox"/> homedirectory	<input type="checkbox"/> homephone	<input type="checkbox"/> homepostaladdress	<input type="checkbox"/> inetuserhttpurl	<input type="checkbox"/> inetuserstatus	<input type="checkbox"/> initials	<input type="checkbox"/> internationalisdnnumber	<input type="checkbox"/> ipacertmapdata	<input type="checkbox"/> ipakrbauthzdata	<input type="checkbox"/> ipanhash	<input type="checkbox"/> ipanthomedirectory	<input type="checkbox"/> ipanthomedirectorydrive	<input type="checkbox"/> ipantlogonscript	<input type="checkbox"/> ipantprofilepath	<input type="checkbox"/> ipantsecurityidentifier	<input type="checkbox"/> ipasshpubkey	<input type="checkbox"/> ipatokenradiusconfiglink	<input type="checkbox"/> ipatokenradiususername	<input type="checkbox"/> ipauniqueid	<input type="checkbox"/> ipauserauthtype	<input type="checkbox"/> jpegphoto	<input type="checkbox"/> krballowedtodelegate	<input type="checkbox"/> krbcanonicalname	<input type="checkbox"/> krbextradata
<input type="checkbox"/> audio	<input checked="" type="checkbox"/> businesscategory																																						
<input type="checkbox"/> carlicense	<input type="checkbox"/> cn																																						
<input checked="" type="checkbox"/> departmentnumber	<input type="checkbox"/> description																																						
<input type="checkbox"/> destinationindicator	<input type="checkbox"/> displayname																																						
<input checked="" type="checkbox"/> employeenumber	<input checked="" type="checkbox"/> employeetype																																						
<input type="checkbox"/> facsimiletelephonenumber	<input type="checkbox"/> gecos																																						
<input type="checkbox"/> gidnumber	<input type="checkbox"/> givenname																																						
<input type="checkbox"/> homedirectory	<input type="checkbox"/> homephone																																						
<input type="checkbox"/> homepostaladdress	<input type="checkbox"/> inetuserhttpurl																																						
<input type="checkbox"/> inetuserstatus	<input type="checkbox"/> initials																																						
<input type="checkbox"/> internationalisdnnumber	<input type="checkbox"/> ipacertmapdata																																						
<input type="checkbox"/> ipakrbauthzdata	<input type="checkbox"/> ipanhash																																						
<input type="checkbox"/> ipanthomedirectory	<input type="checkbox"/> ipanthomedirectorydrive																																						
<input type="checkbox"/> ipantlogonscript	<input type="checkbox"/> ipantprofilepath																																						
<input type="checkbox"/> ipantsecurityidentifier	<input type="checkbox"/> ipasshpubkey																																						
<input type="checkbox"/> ipatokenradiusconfiglink	<input type="checkbox"/> ipatokenradiususername																																						
<input type="checkbox"/> ipauniqueid	<input type="checkbox"/> ipauserauthtype																																						
<input type="checkbox"/> jpegphoto	<input type="checkbox"/> krballowedtodelegate																																						
<input type="checkbox"/> krbcanonicalname	<input type="checkbox"/> krbextradata																																						

* Required field

f. Click the **Add** button to save the new delegation rule.

16.3. VIEWING EXISTING DELEGATION RULES USING IDM WEBUI

This section describes how to view existing delegation rules using the IdM WebUI.

Prerequisites

- You are logged in to the IdM Web UI as a member of the **admins** group.

Procedure

- From the **IPA Server** menu, click **Role-Based Access Control → Delegations**.

The screenshot shows the 'Delegations' page in the IdM WebUI. The 'Role-Based Access Control' tab is selected. A table lists a single delegation rule named 'basic manager attributes'. The 'Delegation name' column contains a checkbox and the text 'Delegation name'. The 'basic manager attributes' column also contains a checkbox and the text 'basic manager attributes'. Below the table, it says '1 delegation matched'. At the top right of the table area are three buttons: 'Refresh', 'Delete', and '+ Add'.

16.4. MODIFYING A DELEGATION RULE USING IDM WEBUI

This section describes how to modify an existing delegation rule using the IdM WebUI.

Prerequisites

- You are logged in to the IdM Web UI as a member of the **admins** group.

Procedure

1. From the **IPA Server** menu, click **Role-Based Access Control → Delegations**.

The screenshot shows the 'Delegations' page in the IdM WebUI. The 'Role-Based Access Control' tab is selected. A table lists a single delegation rule named 'basic manager attributes'. The 'Delegation name' column contains a checkbox and the text 'Delegation name'. The 'basic manager attributes' column also contains a checkbox and the text 'basic manager attributes'. Below the table, it says '1 delegation matched'. At the top right of the table area are three buttons: 'Refresh', 'Delete', and '+ Add'.

2. Click on the rule you want to modify.

3. Make the desired changes:

- Change the name of the rule.
- Change granted permissions by selecting the check boxes that indicate whether users will have the right to view the given attributes (*read*) and add or change the given attributes (*write*).
- In the User group drop-down menu, select the group *who is being granted permissions* to view or edit the entries of users in the member group.
- In the **Member user group** drop-down menu, select the group *whose entries can be edited* by members of the delegation group.

- In the attributes box, select the check boxes by the attributes to which you want to grant permissions. To remove permissions to an attribute, uncheck the relevant check box.

The screenshot shows the 'Role-Based Access Control' section under 'Delegations'. The 'basic manager attributes' delegation is selected. The 'General' tab is active. Under 'Permissions', 'read' and 'write' are checked. The 'User group' dropdown is set to 'managers'. The 'Member user group' dropdown is set to 'employees'. In the 'Attributes' section, several checkboxes are checked, including 'businesscategory', 'departmentnumber', 'displayname', 'employeetype', 'facsimiletelephonenumber', 'givenname', 'homepostaladdress', 'initials', 'ipakrbauthzdata', 'ipanhomeDirectory', 'ipantprofilepath', 'employeeNumber', 'gecos', 'homeDirectory', 'inetUserHTTPURL', 'internationalISDNNumber', 'ipanhash', 'ipantLogonScript', and 'ipassPubKey'. The 'Save' button at the top right is highlighted with a red box.

- Click the **Save** button to save the changes.

16.5. DELETING A DELEGATION RULE USING IDM WEBUI

This section describes how to delete an existing delegation rule using the IdM WebUI.

Prerequisites

- You are logged in to the IdM Web UI as a member of the **admins** group.

Procedure

- From the **IPA Server** menu, click **Role-Based Access Control → Delegations**.
- Select the check box next to the rule you want to remove.
- Click **Delete**.

The screenshot shows the 'Role-Based Access Control' section under 'Delegations'. The 'basic manager attributes' delegation is listed with a checked checkbox. The 'Delete' button to its right is highlighted with a red box.

4. Click **Delete** to confirm.

CHAPTER 17. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING ANSIBLE PLAYBOOKS

Delegation is one of the access control methods in IdM, along with self-service rules and role-based access control (RBAC). You can use delegation to assign permissions to one group of users to manage entries for another group of users.

This section covers the following topics:

- Delegation rules
- Creating the Ansible inventory file for IdM
- Using Ansible to ensure that a delegation rule is present
- Using Ansible to ensure that a delegation rule is absent
- Using Ansible to ensure that a delegation rule has specific attributes
- Using Ansible to ensure that a delegation rule does not have specific attributes

17.1. DELEGATION RULES

You can delegate permissions to user groups to manage users by creating **delegation rules**.

Delegation rules allow a specific user group to perform write (edit) operations on specific attributes for users in another user group. This form of access control rule is limited to editing the values of a subset of attributes you specify in a delegation rule; it does not grant the ability to add or remove whole entries or control over unspecified attributes.

Delegation rules grant permissions to existing user groups in IdM. You can use delegation to, for example, allow the **managers** user group to manage selected attributes of users in the **employees** user group.

17.2. CREATING AN ANSIBLE INVENTORY FILE FOR IDM

When working with Ansible, it is good practice to create, in your home directory, a subdirectory dedicated to Ansible playbooks that you copy and adapt from the **/usr/share/doc/ansible-freeipa/*** and **/usr/share/doc/rhel-system-roles/*** subdirectories. This practice has the following advantages:

- You can find all your playbooks in one place.
- You can run your playbooks without invoking **root** privileges.

Procedure

1. Create a directory for your Ansible configuration and playbooks in your home directory:

```
$ mkdir ~/MyPlaybooks/
```

2. Change into the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks/
```

3. Create the `~/MyPlaybooks/ansible.cfg` file with the following content:

```
[defaults]
inventory = /home/<username>/MyPlaybooks/inventory

[privilege_escalation]
become=True
```

4. Create the `~/MyPlaybooks/inventory` file with the following content:

```
[eu]
server.idm.example.com

[us]
replica.idm.example.com

[ipaserver:children]
eu
us
```

This configuration defines two host groups, **eu** and **us**, for hosts in these locations. Additionally, this configuration defines the **ipaserver** host group, which contains all hosts from the **eu** and **us** groups.

17.3. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS PRESENT

The following procedure describes how to use an Ansible playbook to define privileges for a new IdM delegation rule and ensure its presence. In the example, the new **basic manager attributes** delegation rule grants the **managers** group the ability to read and write the following attributes for members of the **employees** group:

- **businesscategory**
- **departmentnumber**
- **employeenumber**
- **employeetype**

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
 - Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `delegation-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/delegation/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/delegation/delegation-present.yml  
delegation-present-copy.yml
```

3. Open the `delegation-present-copy.yml` Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the `ipadelegation` task section:
 - Set the `ipaadmin_password` variable to the password of the IdM administrator.
 - Set the `name` variable to the name of the new delegation rule.
 - Set the `permission` variable to a comma-separated list of permissions to grant: `read` and `write`.
 - Set the `attribute` variable to a list of attributes the delegated user group can manage: `businesscategory`, `departmentnumber`, `employeenumber`, and `employeetype`.
 - Set the `group` variable to the name of the group that is being given access to view or modify attributes.
 - Set the `membergroup` variable to the name of the group whose attributes can be viewed or modified.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Playbook to manage a delegation rule  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Ensure delegation "basic manager attributes" is present  
      ipadelegation:  
        ipaadmin_password: Secret123  
        name: "basic manager attributes"  
        permission: read, write  
        attribute:  
          - businesscategory  
          - departmentnumber  
          - employeenumber  
          - employeetype  
        group: managers  
        membergroup: employees
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-present-copy.yml
```

Additional resources

- For more information on the concept of a delegation rule, see [Delegation rules](#).
- For more sample Ansible playbooks that use the **ipadelegation** module, see:
 - The **README-delegation.md** file available in the **/usr/share/doc/ansible-freeipa/** directory. This file also contains the definitions of the **ipadelegation** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/ipadelegation** directory.

17.4. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS ABSENT

The following procedure describes how to use an Ansible playbook to ensure a specified delegation rule is absent from your IdM configuration. The example below describes how to make sure the custom **basic manager attributes** delegation rule does not exist in IdM.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
 - Your Ansible inventory file is located in the **~/MyPlaybooks/** directory.

Procedure

1. Navigate to the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks>
```

2. Make a copy of the **delegation-absent.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/delegation/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/delegation/delegation-present.yml  
delegation-absent-copy.yml
```

3. Open the **delegation-absent-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipadelegation** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the delegation rule.

- Set the **state** variable to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Delegation absent
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure delegation "basic manager attributes" is absent
      ipadelegation:
        ipaadmin_password: Secret123
        name: "basic manager attributes"
        state: absent
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-absent-copy.yml
```

Additional resources

- For more information on the concept of a delegation rule, see [Delegation rules](#).
- For more sample Ansible playbooks that use the **ipadelegation** module, see:
 - The **README-delegation.md** file available in the **/usr/share/doc/ansible-freeipa/** directory. This file also contains the definitions of the **ipadelegation** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/ipadelegation** directory.

17.5. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE HAS SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that a delegation rule has specific settings. You can use this playbook to modify a delegation role you have previously created. In the example, you ensure the **basic manager attributes** delegation rule only has the **departmentnumber** member attribute.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
 - Your Ansible inventory file is located in the **~/MyPlaybooks/** directory.

- The **basic manager attributes** delegation rule exists in IdM.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **delegation-member-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/delegation/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/delegation/delegation-member-present.yml delegation-member-present-copy.yml
```

3. Open the **delegation-member-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipadelegation** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the delegation rule to modify.
- Set the **attribute** variable to **departmentnumber**.
- Set the **action** variable to **member**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Delegation member present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure delegation "basic manager attributes" member attribute departmentnumber
      is present
        ipadelegation:
          ipaadmin_password: Secret123
          name: "basic manager attributes"
          attribute:
            - departmentnumber
          action: member
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-member-present-copy.yml
```

Additional resources

- For more information on the concept of a delegation rule, see [Delegation rules](#).

- For more sample Ansible playbooks that use the **ipadelegation** module, see:
 - The **README-delegation.md** file available in the **/usr/share/doc/ansible-freeipa** directory. This file also contains the definitions of the **ipadelegation** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/ipadelegation** directory.

17.6. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE DOES NOT HAVE SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that a delegation rule does not have specific settings. You can use this playbook to make sure a delegation role does not grant undesired access. In the example, you ensure the **basic manager attributes** delegation rule does not have the **employeenumber** and **employeetype** member attributes.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
 - You are using Ansible version 2.8 or later.
 - You have installed the [ansible-freeipa](#) package.
 - You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
 - Your Ansible inventory file is located in the **~/MyPlaybooks/** directory.
- The **basic manager attributes** delegation rule exists in IdM.

Procedure

1. Navigate to the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **delegation-member-absent.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/delegation/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/delegation/delegation-member-absent.yml delegation-member-absent-copy.yml
```

3. Open the **delegation-member-absent-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipadelegation** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the delegation rule to modify.
- Set the **attribute** variable to **employeenumber** and **employeetype**.
- Set the **action** variable to **member**.

- Set the **state** variable to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Delegation member absent
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure delegation "basic manager attributes" member attributes employeeNumber
      and employeeType are absent
        ipadelegation:
          ipaadmin_password: Secret123
          name: "basic manager attributes"
          attribute:
            - employeeNumber
            - employeeType
          action: member
          state: absent
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-member-absent-copy.yml
```

Additional resources

- For more information on the concept of a delegation rule, see [Delegation rules](#).
- For more sample Ansible playbooks that use the **ipadelegation** module, see:
 - The **README-delegation.md** file available in the **/usr/share/doc/ansible-freeipa/** directory. This file also contains the definitions of the **ipadelegation** variables.
 - The **/usr/share/doc/ansible-freeipa/playbooks/ipadelegation** directory.

CHAPTER 18. MANAGING ROLE-BASED ACCESS CONTROLS IN IDM USING THE CLI

This chapter introduces role-based access control in Identity Management (IdM) and describes the following operations in the command-line interface (CLI):

- Managing permissions
- Managing privileges
- Managing roles

18.1. ROLE-BASED ACCESS CONTROL IN IDM

Role-based access control (RBAC) in IdM grants a very different kind of authority to users compared to self-service and delegation access controls.

Role-based access control is composed of three parts:

- **Permissions** grant the right to perform a specific task such as adding or deleting users, modifying a group, enabling read-access, etc.
- **Privileges** combine permissions, for example all the permissions needed to add a new user.
- **Roles** grant a set of privileges to users, user groups, hosts or host groups.

18.1.1. Permissions in IdM

Permissions are the lowest level unit of role-based access control, they define operations together with the LDAP entries to which those operations apply. Comparable to building blocks, permissions can be assigned to as many privileges as needed.

One or more **rights** define what operations are allowed:

- **write**
- **read**
- **search**
- **compare**
- **add**
- **delete**
- **all**

These operations apply to three basic **targets**:

- **subtree**: a domain name (DN); the subtree under this DN
- **target filter**: an LDAP filter
- **target**: DN with possible wildcards to specify entries

Additionally, the following convenience options set the corresponding attribute(s):

- **type**: a type of object (user, group, etc); sets **subtree** and **target filter**
- **memberof**: members of a group; sets a **target filter**
- **targetgroup**: grants access to modify a specific group (such as granting the rights to manage group membership); sets a **target**

With IdM permissions, you can control which users have access to which objects and even which attributes of these objects. IdM enables you to allow or block individual attributes or change the entire visibility of a specific IdM function, such as users, groups, or sudo, to all anonymous users, all authenticated users, or just a certain group of privileged users.

For example, the flexibility of this approach to permissions is useful for an administrator who wants to limit access of users or groups only to the specific sections these users or groups need to access and to make the other sections completely hidden to them.



NOTE

A permission cannot contain other permissions.

18.1.2. Default managed permissions

Managed permissions are permissions that come by default with IdM. They behave like other permissions created by the user, with the following differences:

- You cannot delete them or modify their name, location, and target attributes.
- They have three sets of attributes:
 - **Default** attributes, the user cannot modify them, as they are managed by IdM
 - **Included** attributes, which are additional attributes added by the user
 - **Excluded** attributes, which are attributes removed by the user

A managed permission applies to all attributes that appear in the default and included attribute sets but not in the excluded set.



NOTE

While you cannot delete a managed permission, setting its bind type to permission and removing the managed permission from all privileges effectively disables it.

Names of all managed permissions start with **System**: for example **System: Add Sudo rule** or **System: Modify Services**. Earlier versions of IdM used a different scheme for default permissions. For example, the user could not delete them and was only able to assign them to privileges. Most of these default permissions have been turned into managed permissions, however, the following permissions still use the previous scheme:

- Add Automember Rebuild Membership Task
- Add Configuration Sub-Entries
- Add Replication Agreements
- Certificate Remove Hold

- Get Certificates status from the CA
- Read DNA Range
- Modify DNA Range
- Read PassSync Managers Configuration
- Modify PassSync Managers Configuration
- Read Replication Agreements
- Modify Replication Agreements
- Remove Replication Agreements
- Read LDBM Database Configuration
- Request Certificate
- Request Certificate ignoring CA ACLs
- Request Certificates from a different host
- Retrieve Certificates from the CA
- Revoke Certificate
- Write IPA Configuration



NOTE

If you attempt to modify a managed permission from the command line, the system does not allow you to change the attributes that you cannot modify, the command fails. If you attempt to modify a managed permission from the Web UI, the attributes that you cannot modify are disabled.

18.1.3. Privileges in IdM

A privilege is a group of permissions applicable to a role.

While a permission provides the rights to do a single operation, there are certain IdM tasks that require multiple permissions to succeed. Therefore, a privilege combines the different permissions required to perform a specific task.

For example, setting up an account for a new IdM user requires the following permissions:

- Creating a new user entry
- Resetting a user password
- Adding the new user to the default IPA users group

Combining these three low-level tasks into a higher level task in the form of a custom privilege named, for example, **Add User** makes it easier for a system administrator to manage roles. IdM already contains several default privileges. Apart from users and user groups, privileges are also assigned to hosts and host groups, as well as network services. This practice permits a fine-grained control of operations by a set of users on a set of hosts using specific network services.

**NOTE**

A privilege may not contain other privileges.

18.1.4. Roles in IdM

A role is a list of privileges that users specified for the role possess.

In effect, permissions grant the ability to perform given low-level tasks (create a user entry, add an entry to a group, etc.), privileges combine one or more of these permissions needed for a higher-level task (such as creating a new user in a given group). Roles gather privileges together as needed: for example, a User Administrator role would be able to add, modify, and delete users.

**IMPORTANT**

Roles are used to classify permitted actions. They are not used as a tool to implement privilege separation or to protect from privilege escalation.

**NOTE**

Roles can not contain other roles.

18.1.5. Predefined roles in Identity Management

Red Hat Identity Management provides the following range of pre-defined roles:

Table 18.1. Predefined Roles in Identity Management

Role	Privilege	Description
Helpdesk	Modify Users and Reset passwords, Modify Group membership	Responsible for performing simple user administration tasks
IT Security Specialist	Netgroups Administrators, HBAC Administrator, Sudo Administrator	Responsible for managing security policy such as host-based access controls, sudo rules
IT Specialist	Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators	Responsible for managing hosts
Security Architect	Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator	Responsible for managing the Identity Management environment, creating trusts, creating replication agreements
User Administrator	User Administrators, Group Administrators, Stage User Administrators	Responsible for creating users and groups

18.2. MANAGING IDM PERMISSIONS IN THE CLI

This section describes how to manage Identity Management (IdM) permissions using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

- Create new permission entries with the **ipa permission-add** command.
For example, to add a permission named *dns admin*:

```
$ ipa permission-add "dns admin"
```

- Specify the properties of the permission with the following options:

- bindtype** specifies the bind rule type. This option accepts the **all**, **anonymous**, and **permission** arguments. The **permission** bindtype means that only the users who are granted this permission via a role can exercise it.

For example:

```
$ ipa permission-add "dns admin" --bindtype=all
```

If you do not specify **--bindtype**, then **permission** is the default value.



NOTE

It is not possible to add permissions with a non-default bind rule type to privileges. You also cannot set a permission that is already present in a privilege to a non-default bind rule type.

- right** lists the rights granted by the permission, it replaces the deprecated **--permissions** option. The available values are **add**, **delete**, **read**, **search**, **compare**, **write**, **all**.

You can set multiple attributes by using multiple **--right** options or with a comma-separated list inside curly braces. For example:

```
$ ipa permission-add "dns admin" --right=read --right=write
```

```
$ ipa permission-add "dns admin" --right={read,write}
```



NOTE

add and **delete** are entry-level operations (for example deleting a user, adding a group, etc.) while **read**, **search**, **compare** and **write** are more attribute-level: you can write to **userCertificate** but not read **userPassword**.

- **--attrs** gives the list of attributes over which the permission is granted. You can set multiple attributes by using multiple **--attrs** options or by listing the options in a comma-separated list inside curly braces. For example:

```
$ ipa permission-add "dns admin" --attrs=description --attrs=automountKey
```

```
$ ipa permission-add "dns admin" --attrs={description,automountKey}
```

The attributes provided with **--attrs** must exist and be allowed attributes for the given object type, otherwise the command fails with schema syntax errors.

- **--type** defines the entry object type to which the permission applies, such as user, host, or service. Each type has its own set of allowed attributes.

For example:

```
$ ipa permission-add "manage service" --right=all --type=service --attrs=krbprincipalkey - -attrs=krbprincipalname --attrs=managedby
```

- **--subtree** gives a subtree entry; the filter then targets every entry beneath this subtree entry. Provide an existing subtree entry; **--subtree** does not accept wildcards or non-existent domain names (DNs). Include a DN within the directory. Because IdM uses a simplified, flat directory tree structure, **--subtree** can be used to target some types of entries, like automount locations, which are containers or parent entries for other configuration. For example:

```
$ ipa permission-add "manage automount locations" --subtree="ldap://ldap.example.com:389/cn=automount,dc=example,dc=com" --right=write --attrs=automountmapname --attrs=automountkey --attrs=automountInformation
```



NOTE

The **--type** and **--subtree** options are mutually exclusive: you can see the inclusion of filters for **--type** as a simplification of **--subtree**, intending to make life easier for an admin.

- **--filter** uses an LDAP filter to identify which entries the permission applies to. IdM automatically checks the validity of the given filter. The filter can be any valid LDAP filter, for example:

```
$ ipa permission-add "manage Windows groups" --filter="(!(objectclass=posixgroup))" --right=write --attrs=description
```

- **--memberof** sets the target filter to members of the given group after checking that the group exists. For example, to let the users with this permission modify the login shell of members of the engineers group:

```
$ ipa permission-add ManageShell --right="write" --type=user --attr=loginshell --memberof=engineers
```

- **--targetgroup** sets target to the specified user group after checking that the group exists. For example, to let those with the permission write the member attribute in the engineers group (so they can add or remove members):

```
$ ipa permission-add ManageMembers --right="write" --
subtree=cn=groups,cn=accounts,dc=example,dc=test --attr=member --
targetgroup=engineers
```

- Optionally, you can specify a target domain name (DN):
 - **--target** specifies the DN to apply the permission to. Wildcards are accepted.
 - **--targetto** specifies the DN subtree where an entry can be moved to.
 - **--targetfrom** specifies the DN subtree from where an entry can be moved.

18.3. COMMAND OPTIONS FOR EXISTING PERMISSIONS

Use the following variants to modify existing permissions as needed:

- To edit existing permissions, use the **ipa permission-mod** command. You can use the same command options as for adding permissions.
- To find existing permissions, use the **ipa permission-find** command. You can use the same command options as for adding permissions.
- To view a specific permission, use the **ipa permission-show** command. The **--raw** argument shows the raw 389-ds ACI that is generated. For example:

```
$ ipa permission-show <permission> --raw
```

- The **ipa permission-del** command deletes a permission completely.

Additional resources

For further details about the **ipa permission** commands, refer to the ipa man page and the **ipa help** command.

18.4. MANAGING IDM PRIVILEGES IN THE CLI

This section describes how to manage Identity Management (IdM) privileges using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- Existing permissions. For details about permissions, see [Managing IdM permissions in the CLI](#).

Procedure

1. Add privilege entries using the **ipa privilege-add** command
For example, to add a privilege named *managing filesystems* with a description:

```
$ ipa privilege-add "managing filesystems" --desc="for filesystems"
```

2. Assign the required permissions to the privilege group with the **privilege-add-permission** command

For example, to add the permissions named *managing automount* and *managing ftp services* to the *managing filesystems* privilege:

```
$ ipa privilege-add-permission "managing filesystems" --permissions="managing automount"  
--permissions="managing ftp services"
```

18.5. COMMAND OPTIONS FOR EXISTING PRIVILEGES

Use the following variants to modify existing privileges as needed:

- To modify existing privileges, use the **ipa privilege-mod** command.
- To find existing privileges, use the **ipa privilege-find** command.
- To view a specific privilege, use the **ipa privilege-show** command.
- The **ipa privilege-remove-permission** command removes one or more permissions from a privilege.
- The **ipa privilege-del** command deletes a privilege completely.

Additional resources

For further details about the **ipa privilege** commands, refer to the ipa man page and the **ipa help** command.

18.6. MANAGING IDM ROLES IN THE CLI

This section describes how to manage Identity Management (IdM) roles using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- Existing privileges. For details about privileges, see [Managing IdM privileges in the CLI](#).

Procedure

1. Add new role entries using the **ipa role-add** command:

```
$ ipa role-add --desc="User Administrator" useradmin  
-----  
Added role "useradmin"  
-----  
Role name: useradmin  
Description: User Administrator
```

2. Add the required privileges to the role using the **ipa role-add-privilege** command:

```
$ 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 members to the role using the **ipa role-add-member** command. Allowed member types are: users, groups, hosts and hostgroups.

For example, to add the group named *useradmins* to the previously created *useradmin* role:

```
$ ipa role-add-member --groups=useradmins useradmin
Role name: useradmin
Description: User Administrator
Member groups: useradmins
Privileges: user administrators
-----
Number of members added 1
-----
```

18.7. COMMAND OPTIONS FOR EXISTING ROLES

Use the following variants to modify existing roles as needed:

- To modify existing roles, use the **ipa role-mod** command.
- To find existing roles, use the **ipa role-find** command.
- To view a specific role, use the **ipa role-show** command.
- To remove a member from the role, use the **ipa role-remove-member** command.
- The **ipa role-remove-privilege** command removes one or more privileges from a role.
- The **ipa role-del** command deletes a role completely.

Additional resources

For further details about the **ipa role** commands, refer to the ipa man page and the **ipa help** command.

CHAPTER 19. MANAGING ROLE-BASED ACCESS CONTROLS USING THE IDM WEB UI

This chapter introduces role-based access control in Identity Management (IdM) and describes the following operations in the web interface (Web UI):

- Managing permissions
- Managing privileges
- Managing roles

19.1. ROLE-BASED ACCESS CONTROL IN IDM

Role-based access control (RBAC) in IdM grants a very different kind of authority to users compared to self-service and delegation access controls.

Role-based access control is composed of three parts:

- **Permissions** grant the right to perform a specific task such as adding or deleting users, modifying a group, enabling read-access, etc.
- **Privileges** combine permissions, for example all the permissions needed to add a new user.
- **Roles** grant a set of privileges to users, user groups, hosts or host groups.

19.1.1. Permissions in IdM

Permissions are the lowest level unit of role-based access control, they define operations together with the LDAP entries to which those operations apply. Comparable to building blocks, permissions can be assigned to as many privileges as needed.

One or more **rights** define what operations are allowed:

- **write**
- **read**
- **search**
- **compare**
- **add**
- **delete**
- **all**

These operations apply to three basic **targets**:

- **subtree**: a domain name (DN); the subtree under this DN
- **target filter**: an LDAP filter
- **target**: DN with possible wildcards to specify entries

Additionally, the following convenience options set the corresponding attribute(s):

- **type**: a type of object (user, group, etc); sets **subtree** and **target filter**
- **memberof**: members of a group; sets a **target filter**
- **targetgroup**: grants access to modify a specific group (such as granting the rights to manage group membership); sets a **target**

With IdM permissions, you can control which users have access to which objects and even which attributes of these objects. IdM enables you to allow or block individual attributes or change the entire visibility of a specific IdM function, such as users, groups, or sudo, to all anonymous users, all authenticated users, or just a certain group of privileged users.

For example, the flexibility of this approach to permissions is useful for an administrator who wants to limit access of users or groups only to the specific sections these users or groups need to access and to make the other sections completely hidden to them.



NOTE

A permission cannot contain other permissions.

19.1.2. Default managed permissions

Managed permissions are permissions that come by default with IdM. They behave like other permissions created by the user, with the following differences:

- You cannot delete them or modify their name, location, and target attributes.
- They have three sets of attributes:
 - **Default** attributes, the user cannot modify them, as they are managed by IdM
 - **Included** attributes, which are additional attributes added by the user
 - **Excluded** attributes, which are attributes removed by the user

A managed permission applies to all attributes that appear in the default and included attribute sets but not in the excluded set.



NOTE

While you cannot delete a managed permission, setting its bind type to permission and removing the managed permission from all privileges effectively disables it.

Names of all managed permissions start with **System**: for example **System: Add Sudo rule** or **System: Modify Services**. Earlier versions of IdM used a different scheme for default permissions. For example, the user could not delete them and was only able to assign them to privileges. Most of these default permissions have been turned into managed permissions, however, the following permissions still use the previous scheme:

- Add Automember Rebuild Membership Task
- Add Configuration Sub-Entries
- Add Replication Agreements
- Certificate Remove Hold

- Get Certificates status from the CA
- Read DNA Range
- Modify DNA Range
- Read PassSync Managers Configuration
- Modify PassSync Managers Configuration
- Read Replication Agreements
- Modify Replication Agreements
- Remove Replication Agreements
- Read LDBM Database Configuration
- Request Certificate
- Request Certificate ignoring CA ACLs
- Request Certificates from a different host
- Retrieve Certificates from the CA
- Revoke Certificate
- Write IPA Configuration



NOTE

If you attempt to modify a managed permission from the command line, the system does not allow you to change the attributes that you cannot modify, the command fails. If you attempt to modify a managed permission from the Web UI, the attributes that you cannot modify are disabled.

19.1.3. Privileges in IdM

A privilege is a group of permissions applicable to a role.

While a permission provides the rights to do a single operation, there are certain IdM tasks that require multiple permissions to succeed. Therefore, a privilege combines the different permissions required to perform a specific task.

For example, setting up an account for a new IdM user requires the following permissions:

- Creating a new user entry
- Resetting a user password
- Adding the new user to the default IPA users group

Combining these three low-level tasks into a higher level task in the form of a custom privilege named, for example, **Add User** makes it easier for a system administrator to manage roles. IdM already contains several default privileges. Apart from users and user groups, privileges are also assigned to hosts and host groups, as well as network services. This practice permits a fine-grained control of operations by a set of users on a set of hosts using specific network services.

**NOTE**

A privilege may not contain other privileges.

19.1.4. Roles in IdM

A role is a list of privileges that users specified for the role possess.

In effect, permissions grant the ability to perform given low-level tasks (create a user entry, add an entry to a group, etc.), privileges combine one or more of these permissions needed for a higher-level task (such as creating a new user in a given group). Roles gather privileges together as needed: for example, a User Administrator role would be able to add, modify, and delete users.

**IMPORTANT**

Roles are used to classify permitted actions. They are not used as a tool to implement privilege separation or to protect from privilege escalation.

**NOTE**

Roles can not contain other roles.

19.1.5. Predefined roles in Identity Management

Red Hat Identity Management provides the following range of pre-defined roles:

Table 19.1. Predefined Roles in Identity Management

Role	Privilege	Description
Helpdesk	Modify Users and Reset passwords, Modify Group membership	Responsible for performing simple user administration tasks
IT Security Specialist	Netgroups Administrators, HBAC Administrator, Sudo Administrator	Responsible for managing security policy such as host-based access controls, sudo rules
IT Specialist	Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators	Responsible for managing hosts
Security Architect	Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator	Responsible for managing the Identity Management environment, creating trusts, creating replication agreements
User Administrator	User Administrators, Group Administrators, Stage User Administrators	Responsible for creating users and groups

19.2. MANAGING PERMISSIONS IN THE IDM WEB UI

This section describes how to manage permissions in Identity Management (IdM) using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

- To add a new permission, open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Permissions**:

The screenshot shows the 'Identity' tab selected in the top navigation bar. Below it, the 'Role Based Access Control' sub-menu is expanded, with 'Permissions' highlighted. Under 'Permissions', there is a list of tasks: 'Add Automember Rebuild Membership Task', 'Add Replication Agreements', and 'Certificate Remove Hold'. The 'Add Replication Agreements' item is currently selected.

- The list of permissions opens: Click the **Add** button at the top of the list of the permissions:

The screenshot shows the 'Permissions' list page. At the top right, there are buttons for 'Refresh', 'Delete', and a redboxed 'Add' button. The list below contains four items: 'Add Automember Rebuild Membership Task', 'Add Replication Agreements', and 'Certificate Remove Hold'. The first item, 'Add Automember Rebuild Membership Task', is currently selected.

- The **Add Permission** form opens. Specify the name of the new permission and define its properties accordingly:

Add Permission

Permission name *

Bind rule type permission all anonymous

Granted rights * read search compare
 write add delete
 all

Type

Subtree *

Extra target filter

Target DN

Member of group

Effective attributes

* Required field

4. Select the appropriate Bind rule type:

- **permission** is the default permission type, granting access through privileges and roles
- **all** specifies that the permission applies to all authenticated users
- **anonymous** specifies that the permission applies to all users, including unauthenticated users



NOTE

It is not possible to add permissions with a non-default bind rule type to privileges. You also cannot set a permission that is already present in a privilege to a non-default bind rule type.

5. Choose the rights to grant with this permission in **Granted rights**.
6. Define the method to identify the target entries for the permission:
 - **Type** specifies an entry type, such as user, host, or service. If you choose a value for the **Type** setting, a list of all possible attributes which will be accessible through this ACI for that entry type appears under **Effective Attributes**. Defining **Type** sets **Subtree** and **Target DN** to one of the predefined values.
 - **Subtree** (required) specifies a subtree entry; every entry beneath this subtree entry is then targeted. Provide an existing subtree entry, as **Subtree** does not accept wildcards or non-existent domain names (DNs). For example: **cn=automount,dc=example,dc=com**
 - **Extra target filter** uses an LDAP filter to identify which entries the permission applies to. The filter can be any valid LDAP filter, for example: **(!(objectclass=posixgroup))** IdM automatically checks the validity of the given filter. If you enter an invalid filter, IdM warns you about this when you attempt to save the permission.
 - **Target DN** specifies the domain name (DN) and accepts wildcards. For example: **uid=*,cn=users,cn=accounts,dc=com**
 - **Member of group** sets the target filter to members of the given group. After you specify the filter settings and click **Add**, IdM validates the filter. If all the permission settings are correct, IdM will perform the search. If some of the permissions settings are incorrect, IdM will display a message informing you about which setting is set incorrectly.
7. Add attributes to the permission:
 - If you set **Type**, choose the **Effective attributes** from the list of available ACI attributes.
 - If you did not use **Type**, add the attributes manually by writing them into the **Effective attributes** field. Add a single attribute at a time; to add multiple attributes, click **Add** to add another input field.

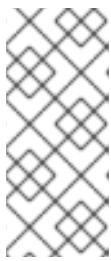


IMPORTANT

If you do not set any attributes for the permission, then the permissions includes all attributes by default.

8. Finish adding the permissions with the **Add** buttons at the bottom of the form:
 - Click the **Add** button to save the permission and go back to the list of permissions.
 - Alternatively, you can save the permission and continue adding additional permissions in the same form by clicking the **Add and Add another** button
 - The **Add and Edit** button enables you to save and continue editing the newly created permission.

9. *Optional.* You can also edit the properties of an existing permission by clicking its name from the list of permissions to display the **Permission settings** page.
10. *Optional.* If you need to remove an existing permission, click the **Delete** button once you ticked the check box next to its name in the list, to display The **Remove permissions** dialog.



NOTE

Operations on default managed permissions are restricted: the attributes you cannot modify are disabled in the IdM Web UI and you cannot delete the managed permissions completely.

However, you can effectively disable a managed permission that has a bind type set to permission, by removing the managed permission from all privileges.

For example, to let those with the permission write the member attribute in the engineers group (so they can add or remove members):

Add permission

Permission name *

Bind rule type permission all anonymous

Granted rights * read search compare
 write add delete
 all

Type

Subtree *

Extra target filter

Target DN

Member of group

Effective attributes

* Required field

19.3. MANAGING PRIVILEGES IN THE IDM WEBUI

This section describes how to manage privileges in IdM using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

- Existing permissions. For details about permissions, see [Managing permissions in the IdM Web UI](#).

Procedure

- To add a new privilege, open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Privileges**:

The screenshot shows the IDM Web UI interface. At the top, there is a navigation bar with tabs: Identity, Policy, Authentication, Network Services, IPA Server, and Config. The IPA Server tab is selected. Below the navigation bar, there is a sub-navigation menu with items: Roles, Privileges (which is currently selected and highlighted in blue), Permissions, Self Service Permissions, and Delegations. To the right of this menu is a search icon. Further down is a table titled 'Privileges' with columns: 'Privilege name' and 'Description'. The table contains four rows: 'Password Policy Readers' (description: Read password policies), 'RBAC Readers' (description: Read roles, privileges, permissions and ACIs), and 'Replication Administrators'. There are also two empty rows above these entries. At the bottom of the table are three buttons: Refresh, Delete, and + Add.

- The list of privileges opens. Click the **Add** button at the top of the list of privileges:

This screenshot shows the same 'Privileges' list as the previous one, but with a red box highlighting the '+ Add' button located at the top right of the table. The rest of the interface, including the table structure and the three existing privilege entries, remains the same.

- The **Add Privilege** form opens. Enter the name and a description of the privilege:

This screenshot shows the 'Add Privilege' dialog box. It has a title bar 'Add Privilege' with a close button. Inside, there are two input fields: 'Privilege name *' with a value 'New Privilege' and 'Description' with a value 'For employees'. Below these fields is a note '* Required field'. At the bottom of the dialog are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'. The 'Add and Edit' button is highlighted with a red box.

- Click the **Add and Edit** button in order to save the new privilege and continue to the privilege configuration page to add permissions.
- Edit the properties of privileges by clicking on the privileges name in the privileges list. The privileges configuration page opens.

6. The **Permissions** tab displays a list of permissions included in the selected privilege. Click the **Add** button at the top of the list to add permissions to the privilege:

The screenshot shows a web-based interface for managing privileges. At the top, there are three tabs: 'Permissions (5)', 'Settings', and 'Roles'. Below the tabs, there are buttons for 'Refresh', 'Delete', and a redboxed 'Add' button. A table lists six permissions, each with a checkbox and a link: 'Permission name', 'System: Add Groups', 'System: Add HBAC Rule', 'System: Add Hosts', 'System: Add Privileges', and 'System: Add Roles'.

7. Tick the check box next to the name of each permission to add, and use the **>** button to move the permissions to the **Prospective** column:

The screenshot shows a modal dialog titled 'Add Privilege New Privilege into Permissions'. It has a 'Filter available Permissions' input field and a 'Filter' button. The interface is divided into two columns: 'Available' on the left and 'Prospective' on the right. The 'Available' column lists ten permissions, each with a checkbox. The 'Prospective' column lists three permissions, also with checkboxes. Between the columns are two buttons: a right-pointing arrow above a left-pointing arrow, used for moving items between the lists. At the bottom are 'Add' and 'Cancel' buttons.

8. Confirm by clicking the **Add** button.

9. *Optional.* If you need to remove permissions, click the **Delete** button after you ticked the check box next to the relevant permission: the **Remove privileges from permissions** dialog opens.
10. *Optional.* If you need to delete an existing privilege, click the **Delete** button after you ticked the check box next to its name in the list: the **Remove privileges** dialog opens.

19.4. MANAGING ROLES IN THE IDM WEB UI

This section describes how to manage roles in Identity Management (IdM) using the web interface (IdM Web UI).

Prerequisites

- Administrator privileges for managing IdM or the **User Administrator** role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).
- Existing privileges. For details about privileges, see [Managing privileges in the IdM Web UI](#).

Procedure

1. To add a new role, open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Roles**:

The screenshot shows the IdM Web UI interface. At the top, there is a navigation bar with tabs: Identity, Policy, Authentication, Network Services, and IPA Server. The IPA Server tab is currently selected. Below the navigation bar, there is a secondary menu titled "Role Based Access Control". This menu has several options: Roles (which is highlighted in blue), Privileges, Permissions, Self Service Permissions, and Delegations. There is also a search icon. At the bottom of this menu, there are three buttons: "Add Automember Rebuild Membership Task", "Add Replication Agreements", and "Certificate Remove Hold".

2. The list of roles opens. Click the **Add** button at the top of the list of the role-based access control instructions.

The screenshot shows the 'Identity' tab selected in the navigation bar. Below it, the 'Role Based Access Control' tab is active. The main content area displays a table of roles:

<input type="checkbox"/>	Role name	Description
<input type="checkbox"/>	IT Security Specialist	IT Security Specialist
<input type="checkbox"/>	IT Specialist	IT Specialist
<input type="checkbox"/>	Security Architect	Security Architect

At the top right of the table, there are three buttons: 'Refresh', 'Delete', and '+ Add'. The '+ Add' button is highlighted with a red box.

3. The **Add Role** form opens. Enter the role name and a description:

The dialog box has a title 'Add Role' and a close button. It contains two input fields: 'Role name *' with the value 'Example Role' and 'Description' with the value 'For engineers'. Below the fields is a note: '* Required field'. At the bottom are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'. The 'Add and Edit' button is highlighted with a red box.

4. Click the **Add and Edit** button to save the new role and go to the role configuration page to add privileges and users.
5. Edit the properties of roles by clicking on the roles name in the role list. The roles configuration page opens.
6. Add members using the **Users, Users Groups, Hosts, Host Groups or Services** tabs, by clicking the **Add** button on top of the relevant list(s).

Identity Policy Authentication Network Services IPA Server

Role Based Access Control ID Ranges ID Views Realm Domains

Roles » Example Role

Role: Example Role

Example Role members:

Users (1)	User Groups	Hosts	Host Groups	Services	Privileges
<input type="button" value="Refresh"/>	<input type="button" value="Delete"/>	<input type="button" value="+ Add"/>			
<input type="checkbox"/> User login					
<input type="checkbox"/> employee					

Showing 1 to 1 of 1 entries.

- In the window that opens, select the members on the left and use the > button to move them to the **Prospective** column.

Add Users into Role Example Role

Filter available Users Filter

Available

<input type="checkbox"/> User login
<input checked="" type="checkbox"/> admin
<input type="checkbox"/> helpdesk

Prospective

<input type="checkbox"/> User login
<input type="checkbox"/> manager

> <

Add Cancel

- At the top of the **Privileges** tab, click **Add**.

Identity Policy Authentication Network Services IPA Server

Role Based Access Control ID Ranges ID Views Realm Domains Config

Roles » Example Role

Role: Example Role

Example Role members:

Users (1)	User Groups	Hosts	Host Groups	Services	Privileges (4)	Settings
Refresh	Delete	+ Add				
<input type="checkbox"/> Privilege name						
<input type="checkbox"/> Automount Administrators						
<input type="checkbox"/> Certificate Administrators						
<input type="checkbox"/> DNS Administrators						

9. Select the privileges on the left and use the > button to move them to the **Prospective** column.

Add Role Example Role into Privileges

Filter available Privileges Filter

Available

<input type="checkbox"/> Privilege name
<input type="checkbox"/> ADTrust Agents
<input type="checkbox"/> Automember Readers
<input type="checkbox"/> Automember Task Administrator
<input type="checkbox"/> DNS Servers
<input type="checkbox"/> HBAC Administrator
<input type="checkbox"/> Host Enrollment
<input type="checkbox"/> IPA Masters Readers
<input type="checkbox"/> Kerberos Ticket Policy Readers
<input type="checkbox"/> Modify Group membership
<input type="checkbox"/> Modify Users and Reset passwords
<input type="checkbox"/> Netgroups Administrators
<input type="checkbox"/> New Privilege
<input type="checkbox"/> Password Policy Administrator
<input type="checkbox"/> Password Policy Readers

Prospective

<input type="checkbox"/> Privilege name
<input type="checkbox"/> Group Administrators
<input type="checkbox"/> Host Administrators
<input type="checkbox"/> Host Group Administrators

> <

Add Cancel

10. Click the **Add** button to save.

11. *Optional.* If you need to remove privileges or members from a role, click the **Delete** button after you ticked the check box next to the name of the entity you want to remove. A dialog opens.
12. *Optional.* If you need to remove an existing role, click the **Delete** button after you ticked the check box next to its name in the list, to display the **Remove roles** dialog.

CHAPTER 20. PREPARING YOUR ENVIRONMENT FOR MANAGING IDM USING ANSIBLE PLAYBOOKS

As a system administrator managing Identity Management (IdM), when working with Red Hat Ansible Engine, it is good practice to do the following:

- Create a subdirectory dedicated to Ansible playbooks in your home directory, for example `~/MyPlaybooks`.
- Copy and adapt sample Ansible playbooks from the `/usr/share/doc/ansible-freeipa/*` and `/usr/share/doc/rhel-system-roles/*` directories and subdirectories into your `~/MyPlaybooks` directory.
- Include your inventory file in your `~/MyPlaybooks` directory.

Using this practice, you can find all your playbooks in one place and you can run your playbooks without invoking root privileges.



NOTE

You only need **root** privileges on the managed nodes to execute the **ipaserver**, **ipareplica**, **ipaclient** and **ipabackup ansible-freeipa** roles. These roles require privileged access to directories and the **dnf** software package manager.

This section describes how to create the `~/MyPlaybooks` directory and configure it so that you can use it to store and run Ansible playbooks.

Prerequisites

- You have installed an IdM server on your managed nodes, `server.idm.example.com` and `replica.idm.example.com`.
- You have configured DNS and networking so you can log in to the managed nodes, `server.idm.example.com` and `replica.idm.example.com`, directly from the control node.
- You know the IdM **admin** password.

Procedure

1. Create a directory for your Ansible configuration and playbooks in your home directory:

```
$ mkdir ~/MyPlaybooks/
```

2. Change into the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

3. Create the `~/MyPlaybooks/ansible.cfg` file with the following content:

```
[defaults]
inventory = /home/your_username/MyPlaybooks/inventory
```

```
[privilege_escalation]
become=True
```

4. Create the `~/MyPlaybooks/inventory` file with the following content:

```
[eu]
server.idm.example.com

[us]
replica.idm.example.com

[ipaserver:children]
eu
us
```

This configuration defines two host groups, `eu` and `us`, for hosts in these locations. Additionally, this configuration defines the `ipaserver` host group, which contains all hosts from the `eu` and `us` groups.

5. [Optional] Create an SSH public and private key. To simplify access in your test environment, do not set a password on the private key:

```
$ ssh-keygen
```

6. Copy the SSH public key to the IdM `admin` account on each managed node:

```
$ ssh-copy-id admin@server.idm.example.com
$ ssh-copy-id admin@replica.idm.example.com
```

These commands require that you enter the IdM `admin` password.

Additional resources

- For more information on installing an IdM server using an Ansible playbook, see [Installing an Identity Management server using an Ansible playbook](#).
- For an overview of available formats for an Ansible inventory file including examples, see [How to build your inventory](#).

CHAPTER 21. USING ANSIBLE PLAYBOOKS TO MANAGE ROLE-BASED ACCESS CONTROL IN IDM

Role-based access control (RBAC) is a policy-neutral access-control mechanism defined around roles and privileges. The components of RBAC in Identity Management (IdM) are roles, privileges and permissions:

- **Permissions** grant the right to perform a specific task such as adding or deleting users, modifying a group, enabling read-access, etc.
- **Privileges** combine permissions, for example all the permissions needed to add a new user.
- **Roles** grant a set of privileges to users, user groups, hosts or host groups.

Especially in large companies, using RBAC can help create a hierarchical system of administrators with their individual areas of responsibility.

This chapter describes the following operations performed when managing RBAC using Ansible playbooks:

- [Permissions in IdM](#)
- [Default managed permissions](#)
- [Privileges in IdM](#)
- [Roles in IdM](#)
- [Predefined roles in IdM](#)
- [Using Ansible to ensure an IdM RBAC role with privileges is present](#)
- [Using Ansible to ensure an IdM RBAC role is absent](#)
- [Using Ansible to ensure that a group of users is assigned to an IdM RBAC role](#)
- [Using Ansible to ensure that specific users are not assigned to an IdM RBAC role](#)
- [Using Ansible to ensure a service is a member of an IdM RBAC role](#)
- [Using Ansible to ensure a host is a member of an IdM RBAC role](#)
- [Using Ansible to ensure a host group is a member of an IdM RBAC role](#)

21.1. PERMISSIONS IN IDM

Permissions are the lowest level unit of role-based access control, they define operations together with the LDAP entries to which those operations apply. Comparable to building blocks, permissions can be assigned to as many privileges as needed.

One or more **rights** define what operations are allowed:

- **write**
- **read**
- **search**

- **compare**
- **add**
- **delete**
- **all**

These operations apply to three basic **targets**:

- **subtree**: a domain name (DN); the subtree under this DN
- **target filter**: an LDAP filter
- **target**: DN with possible wildcards to specify entries

Additionally, the following convenience options set the corresponding attribute(s):

- **type**: a type of object (user, group, etc); sets **subtree** and **target filter**
- **memberof**: members of a group; sets a **target filter**
- **targetgroup**: grants access to modify a specific group (such as granting the rights to manage group membership); sets a **target**

With IdM permissions, you can control which users have access to which objects and even which attributes of these objects. IdM enables you to allow or block individual attributes or change the entire visibility of a specific IdM function, such as users, groups, or sudo, to all anonymous users, all authenticated users, or just a certain group of privileged users.

For example, the flexibility of this approach to permissions is useful for an administrator who wants to limit access of users or groups only to the specific sections these users or groups need to access and to make the other sections completely hidden to them.



NOTE

A permission cannot contain other permissions.

21.2. DEFAULT MANAGED PERMISSIONS

Managed permissions are permissions that come by default with IdM. They behave like other permissions created by the user, with the following differences:

- You cannot delete them or modify their name, location, and target attributes.
- They have three sets of attributes:
 - **Default** attributes, the user cannot modify them, as they are managed by IdM
 - **Included** attributes, which are additional attributes added by the user
 - **Excluded** attributes, which are attributes removed by the user

A managed permission applies to all attributes that appear in the default and included attribute sets but not in the excluded set.

**NOTE**

While you cannot delete a managed permission, setting its bind type to permission and removing the managed permission from all privileges effectively disables it.

Names of all managed permissions start with **System:**, for example **System: Add Sudo rule** or **System: Modify Services**. Earlier versions of IdM used a different scheme for default permissions. For example, the user could not delete them and was only able to assign them to privileges. Most of these default permissions have been turned into managed permissions, however, the following permissions still use the previous scheme:

- Add Automember Rebuild Membership Task
- Add Configuration Sub-Entries
- Add Replication Agreements
- Certificate Remove Hold
- Get Certificates status from the CA
- Read DNA Range
- Modify DNA Range
- Read PassSync Managers Configuration
- Modify PassSync Managers Configuration
- Read Replication Agreements
- Modify Replication Agreements
- Remove Replication Agreements
- Read LDBM Database Configuration
- Request Certificate
- Request Certificate ignoring CA ACLs
- Request Certificates from a different host
- Retrieve Certificates from the CA
- Revoke Certificate
- Write IPA Configuration

**NOTE**

If you attempt to modify a managed permission from the command line, the system does not allow you to change the attributes that you cannot modify, the command fails. If you attempt to modify a managed permission from the Web UI, the attributes that you cannot modify are disabled.

21.3. PRIVILEGES IN IDM

A privilege is a group of permissions applicable to a role.

While a permission provides the rights to do a single operation, there are certain IdM tasks that require multiple permissions to succeed. Therefore, a privilege combines the different permissions required to perform a specific task.

For example, setting up an account for a new IdM user requires the following permissions:

- Creating a new user entry
- Resetting a user password
- Adding the new user to the default IPA users group

Combining these three low-level tasks into a higher level task in the form of a custom privilege named, for example, **Add User** makes it easier for a system administrator to manage roles. IdM already contains several default privileges. Apart from users and user groups, privileges are also assigned to hosts and host groups, as well as network services. This practice permits a fine-grained control of operations by a set of users on a set of hosts using specific network services.



NOTE

A privilege may not contain other privileges.

21.4. ROLES IN IDM

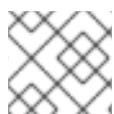
A role is a list of privileges that users specified for the role possess.

In effect, permissions grant the ability to perform given low-level tasks (create a user entry, add an entry to a group, etc.), privileges combine one or more of these permissions needed for a higher-level task (such as creating a new user in a given group). Roles gather privileges together as needed: for example, a User Administrator role would be able to add, modify, and delete users.



IMPORTANT

Roles are used to classify permitted actions. They are not used as a tool to implement privilege separation or to protect from privilege escalation.



NOTE

Roles can not contain other roles.

21.5. PREDEFINED ROLES IN IDENTITY MANAGEMENT

Red Hat Identity Management provides the following range of pre-defined roles:

Table 21.1. Predefined Roles in Identity Management

Role	Privilege	Description
Helpdesk	Modify Users and Reset passwords, Modify Group membership	Responsible for performing simple user administration tasks

Role	Privilege	Description
IT Security Specialist	Netgroups Administrators, HBAC Administrator, Sudo Administrator	Responsible for managing security policy such as host-based access controls, sudo rules
IT Specialist	Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators	Responsible for managing hosts
Security Architect	Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator	Responsible for managing the Identity Management environment, creating trusts, creating replication agreements
User Administrator	User Administrators, Group Administrators, Stage User Administrators	Responsible for creating users and groups

21.6. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE WITH PRIVILEGES IS PRESENT

To exercise more granular control over role-based access (RBAC) to resources in Identity Management (IdM) than the default roles provide, create a custom role.

The following procedure describes how to use an Ansible playbook to define privileges for a new IdM custom role and ensure its presence. In the example, the new `user_and_host_administrator` role contains a unique combination of the following privileges that are present in IdM by default:

- **Group Administrators**
- **User Administrators**
- **Stage User Administrators**
- **Group Administrators**

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the **role-member-user-present.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/role/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-user-present.yml role-member-user-present-copy.yml
```

3. Open the **role-member-user-present-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **iparole** task section:
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the new role.
 - Set the **privilege** list to the names of the IdM privileges that you want to include in the new role.
 - Optionally, set the **user** variable to the name of the user to whom you want to grant the new role.
 - Optionally, set the **group** variable to the name of the group to which you want to grant the new role.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
    - iparole:
        ipaadmin_password: Secret123
        name: user_and_host_administrator
        user: idm_user01
        group: idm_group01
        privilege:
          - Group Administrators
          - User Administrators
          - Stage User Administrators
          - Group Administrators
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-user-present-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see:
 - The **README-role** file available in the `/usr/share/doc/ansible-freeipa/` directory. This file also contains the definitions of the **iparole** variables.
 - The `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

21.7. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE IS ABSENT

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure the absence of an obsolete role so that no administrator assigns it to any user accidentally.

The following procedure describes how to use an Ansible playbook to ensure a role is absent. The example below describes how to make sure the custom **user_and_host_administrator** role does not exist in IdM.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>
```

2. Make a copy of the **role-is-absent.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-is-absent.yml role-is-absent-copy.yml
```

3. Open the **role-is-absent-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **iparole** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the role.
- Ensure that the **state** variable is set to **absent**.

This is the modified Ansible playbook file for the current example:

```

---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
    - iparole:
        ipaadmin_password: Secret123
        name: user_and_host_administrator
        state: absent

```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-is-absent-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see:
 - The **README-role** Markdown file available in the `/usr/share/doc/ansible-freeipa` directory. This file also contains the definitions of the **iparole** variables.
 - The `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

21.8. USING ANSIBLE TO ENSURE THAT A GROUP OF USERS IS ASSIGNED TO AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to assign a role to a specific group of users, for example junior administrators.

The following example describes how to use an Ansible playbook to ensure the built-in IdM RBAC **helpdesk** role is assigned to **junior_sysadmins**.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the `role-member-group-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-group-present.yml  
role-member-group-present-copy.yml
```

3. Open the `role-member-group-present-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `iparole` task section:

- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the role you want to assign.
- Set the `group` variable to the name of the group.
- Set the `action` variable to `member`.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Playbook to manage IPA role with members.  
  hosts: ipaserver  
  become: yes  
  gather_facts: no  
  
  tasks:  
  - iparole:  
      ipaadmin_password: Secret123  
      name: helpdesk  
      group: junior_sysadmins  
      action: member
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-group-present-  
copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the `iparole` module, see the `README-role` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `iparole` variables.

- For more sample Ansible playbooks that use the **iparole** module, see the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

21.9. USING ANSIBLE TO ENSURE THAT SPECIFIC USERS ARE NOT ASSIGNED TO AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure that an RBAC role is not assigned to specific users after they have, for example, moved to different positions within the company.

The following procedure describes how to use an Ansible playbook to ensure that the users named `user_01` and `user_02` are not assigned to the `helpdesk` role.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the `role-member-user-absent.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-user-absent.yml role-member-user-absent-copy.yml
```

3. Open the `role-member-user-absent-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `iparole` task section:

- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the role you want to assign.
- Set the `user` list to the names of the users.
- Set the `action` variable to `member`.
- Set the `state` variable to `absent`.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
```

```

become: yes
gather_facts: no

tasks:
- iparole:
    ipaadmin_password: Secret123
    name: helpdesk
    user
    - user_01
    - user_02
    action: member
    state: absent

```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-user-absent-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see the **README-role** Markdown file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of the **iparole** variables.
- For more sample Ansible playbooks that use the **iparole** module, see the **/usr/share/doc/ansible-freeipa/playbooks/iparole** directory.

21.10. USING ANSIBLE TO ENSURE A SERVICE IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure that a specific service that is enrolled into IdM is a member of a particular role. The following example describes how to ensure that the custom **web_administrator** role can manage the **HTTP** service that is running on the **client01.idm.example.com** server.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the **~/<MyPlaybooks>/** directory.
- The **web_administrator** role exists in IdM.

- The **HTTP/client01.idm.example.com@IDM.EXAMPLE.COM** service exists in IdM.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the **role-member-service-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-service-present-absent.yml role-member-service-present-copy.yml
```

3. Open the **role-member-service-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **iparole** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the role you want to assign.
- Set the **service** list to the name of the service.
- Set the **action** variable to **member**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
    - iparole:
        ipaadmin_password: Secret123
        name: web_administrator
        service:
          - HTTP/client01.idm.example.com
        action: member
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-service-present-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).

- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see the **README-role** Markdown file available in the **/usr/share/doc/ansible-freeipa** directory. The file also contains the definitions of the **iparole** variables.
- For more sample Ansible playbooks that use the **iparole** module, see the **/usr/share/doc/ansible-freeipa/playbooks/iparole** directory.

21.11. USING ANSIBLE TO ENSURE A HOST IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control in Identity Management (IdM), you may want to ensure that a specific host or host group is associated with a specific role. The following example describes how to ensure that the custom **web_administrator** role can manage the **client01.idm.example.com** IdM host on which the **HTTP** service is running.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.
- The **web_administrator** role exists in IdM.
- The **client01.idm.example.com** host exists in IdM.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the **role-member-host-present.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/role/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-host-present.yml role-member-host-present-copy.yml
```

3. Open the **role-member-host-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **iparole** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the role you want to assign.
- Set the **host** list to the name of the host.

This is the modified Ansible playbook file for the current example:

```

---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
    - iparole:
        ipaadmin_password: Secret123
        name: web_administrator
        host:
          - client01.idm.example.com
        action: member

```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-host-present-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see the **README-role** Markdown file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of the **iparole** variables.
- For more sample Ansible playbooks that use the **iparole** module, see the **/usr/share/doc/ansible-freeipa/playbooks/iparole** directory.

21.12. USING ANSIBLE TO ENSURE A HOST GROUP IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control in Identity Management (IdM), you may want to ensure that a specific host or host group is associated with a specific role. The following example describes how to ensure that the custom **web_administrator** role can manage the **web_servers** group of IdM hosts on which the **HTTP** service is running.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the **~/<MyPlaybooks>/** directory.

- The **web_administrator** role exists in IdM.
- The **web_servers** host group exists in IdM.

Procedure

1. Navigate to the `~/<MyPlaybooks>/` directory:

```
$ cd ~/<MyPlaybooks>/
```

2. Make a copy of the **role-member-hostgroup-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-hostgroup-present.yml role-member-hostgroup-present-copy.yml
```

3. Open the **role-member-hostgroup-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **iparole** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the role you want to assign.
- Set the **hostgroup** list to the name of the hostgroup.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
    - iparole:
        ipaadmin_password: Secret123
        name: web_administrator
        hostgroup:
          - web_servers
        action: member
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-hostgroup-present-copy.yml
```

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).

- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the **iparole** module, see the **README-role** Markdown file available in the **/usr/share/doc/ansible-freeipa** directory. The file also contains the definitions of the **iparole** variables.
- For more sample Ansible playbooks that use the **iparole** module, see the **/usr/share/doc/ansible-freeipa/playbooks/iparole** directory.

CHAPTER 22. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PRIVILEGES

Role-based access control (RBAC) is a policy-neutral access-control mechanism defined around roles, privileges, and permissions. Especially in large companies, using RBAC can help create a hierarchical system of administrators with their individual areas of responsibility.

This chapter describes the following operations for using Ansible playbooks to manage RBAC privileges in Identity Management (IdM):

- Using Ansible to ensure a custom RBAC privilege is present
- Using Ansible to ensure member permissions are present in a custom IdM RBAC privilege
- Using Ansible to ensure an IdM RBAC privilege does not include a permission
- Using Ansible to rename a custom IdM RBAC privilege
- Using Ansible to ensure an IdM RBAC privilege is absent

Prerequisites

- You understand the [concepts and principles of RBAC](#).

22.1. USING ANSIBLE TO ENSURE A CUSTOM IDM RBAC PRIVILEGE IS PRESENT

To have a fully-functioning custom privilege in Identity Management (IdM) role-based access control (RBAC), you need to proceed in stages:

1. Create a privilege with no permissions attached.
2. Add permissions of your choice to the privilege.

The following procedure describes how to create an empty privilege using an Ansible playbook so that you can later add permissions to it. The example describes how to create a privilege named **full_host_administration** that is meant to combine all IdM permissions related to host administration.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **privilege-present.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/privilege/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-present.yml privilege-present-copy.yml
```

3. Open the **privilege-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipaprivilege** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the new privilege, **full_host_administration**.
- Optionally, describe the privilege using the **description** variable.

This is the modified Ansible playbook file for the current example:

```
---
- name: Privilege present example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure privilege full_host_administration is present
      ipaprivilege:
        ipaadmin_password: Secret123
        name: full_host_administration
        description: This privilege combines all IdM permissions related to host administration
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory privilege-present-copy.yml
```

22.2. USING ANSIBLE TO ENSURE MEMBER PERMISSIONS ARE PRESENT IN A CUSTOM IDM RBAC PRIVILEGE

To have a fully-functioning custom privilege in Identity Management (IdM) role-based access control (RBAC), you need to proceed in stages:

1. Create a privilege with no permissions attached.
2. Add permissions of your choice to the privilege.

The following procedure describes how to use an Ansible playbook to add permissions to a privilege created in the previous step. The example describes how to add all IdM permissions related to host administration to a privilege named **full_host_administration**. By default, the permissions are distributed between the **Host Enrollment**, **Host Administrators** and **Host Group Administrator** privileges.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.
- The ***full_host_administration*** privilege exists. For information on how to create a privilege using Ansible, see [Using Ansible to ensure a custom IdM RBAC privilege is present](#) .

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **privilege-member-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/privilege/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-member-present.yml  
privilege-member-present-copy.yml
```

3. Open the **privilege-member-present-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipaprivilege** task section:

- Adapt the **name** of the task to correspond to your use case.
- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the privilege.
- Set the **permission** list to the names of the permissions that you want to include in the privilege.
- Make sure that the **action** variable is set to **member**.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Privilege member present example  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Ensure that permissions are present for the "full_host_administration" privilege  
      ipaprivilege:  
        ipaadmin_password: Secret123  
        name: full_host_administration  
        permission:  
          - "System: Add krbPrincipalName to a Host"  
          - "System: Enroll a Host"  
          - "System: Manage Host Certificates"  
          - "System: Manage Host Enrollment Password"
```

- "System: Manage Host Keytab"
- "System: Manage Host Principals"
- "Retrieve Certificates from the CA"
- "Revoke Certificate"
- "System: Add Hosts"
- "System: Add krbPrincipalName to a Host"
- "System: Enroll a Host"
- "System: Manage Host Certificates"
- "System: Manage Host Enrollment Password"
- "System: Manage Host Keytab"
- "System: Manage Host Keytab Permissions"
- "System: Manage Host Principals"
- "System: Manage Host SSH Public Keys"
- "System: Manage Service Keytab"
- "System: Manage Service Keytab Permissions"
- "System: Modify Hosts"
- "System: Remove Hosts"
- "System: Add Hostgroups"
- "System: Modify Hostgroup Membership"
- "System: Modify Hostgroups"
- "System: Remove Hostgroups"

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory privilege-member-present-copy.yml
```

22.3. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE DOES NOT INCLUDE A PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control.

The following procedure describes how to use an Ansible playbook to remove a permission from a privilege. The example describes how to remove the **Request Certificates ignoring CA ACLs** permission from the default **Certificate Administrators** privilege because, for example, the administrator considers it a security risk.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the *~/MyPlaybooks/* directory.

Procedure

1. Navigate to the *~/MyPlaybooks/* directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **privilege-member-present.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/privilege/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-member-absent.yml  
privilege-member-absent-copy.yml
```

3. Open the **privilege-member-absent-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipaprivilege** task section:

- Adapt the **name** of the task to correspond to your use case.
- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the privilege.
- Set the **permission** list to the names of the permissions that you want to remove from the privilege.
- Make sure that the **action** variable is set to **member**.
- Make sure that the **state** variable is set to **absent**.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Privilege absent example  
hosts: ipaserver  
become: true  
  
tasks:  
- name: Ensure that the "Request Certificate ignoring CA ACLs" permission is absent from  
the "Certificate Administrators" privilege  
ipaprivilege:  
  ipaadmin_password: Secret123  
  name: Certificate Administrators  
  permission:  
  - "Request Certificate ignoring CA ACLs"  
  action: member  
  state: absent
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory privilege-member-absent-copy.yml
```

22.4. USING ANSIBLE TO RENAME A CUSTOM IDM RBAC PRIVILEGE

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control.

The following procedure describes how to rename a privilege because, for example, you have removed a few permissions from it. As a result, the name of the privilege is no longer accurate. In the example, the administrator renames a **full_host_administration** privilege to **limited_host_administration**.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.
- The **full_host_administration** privilege exists. For more information on how to add a privilege, see [Using Ansible to ensure a custom IdM RBAC privilege is present](#) .

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **privilege-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/privilege/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-present.yml rename-privilege.yml
```

3. Open the **rename-privilege.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipaprivilege** task section:

- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the current name of the privilege.
- Add the **rename** variable and set it to the new name of the privilege.
- Add the **state** variable and set it to **renamed**.

5. Rename the playbook itself, for example:

```
---
- name: Rename a privilege
  hosts: ipaserver
  become: true
```

6. Rename the task in the playbook, for example:

```
[...]
tasks:
- name: Ensure the full_host_administration privilege is renamed to
```

limited_host_administration

```
ipaprivilege:  
[...]
```

This is the modified Ansible playbook file for the current example:

```
---  
- name: Rename a privilege  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Ensure the full_host_administration privilege is renamed to  
      limited_host_administration  
      ipaprivilege:  
        ipaadmin_password: Secret123  
        name: full_host_administration  
        rename: limited_host_administration  
        state: renamed
```

7. Save the file.
8. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory rename-privilege.yml
```

22.5. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE IS ABSENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control. The following procedure describes how to use an Ansible playbook to ensure that an RBAC privilege is absent. The example describes how to ensure that the **CA administrator** privilege is absent. As a result of the procedure, the **admin** administrator becomes the only user capable of managing certificate authorities in IdM.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- You have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **privilege-absent.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/privilege/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-absent.yml privilege-absent-copy.yml
```

3. Open the **privilege-absent-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **ipaprivilege** task section:
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the privilege you want to remove.
 - Make sure that the **state** variable is set it to **absent**.
5. Rename the task in the playbook, for example:

```
[...]
tasks:
- name: Ensure privilege "CA administrator" is absent
  ipaprivilege:
  [...]
```

This is the modified Ansible playbook file for the current example:

```
---
- name: Privilege absent example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure privilege "CA administrator" is absent
      ipaprivilege:
        ipaadmin_password: Secret123
        name: CA administrator
        state: absent
```

6. Save the file.
7. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory privilege-absent-copy.yml
```

Additional resources

- For more information on the concept of a privilege in IdM RBAC, see [Privileges in IdM](#).
- For more information on the concept of a permission in IdM RBAC, see [Permissions in IdM](#).
- For more sample Ansible playbooks that use the **ipaprivilege** module, see the **README-privilege** file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of the **ipaprivilege** variables.
- For more sample Ansible playbooks that use the **ipaprivilege** module, see the **/usr/share/doc/ansible-freeipa/playbooks/ipaprivilege** directory.

CHAPTER 23. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PERMISSIONS IN IDM

Role-based access control (RBAC) is a policy-neutral access control mechanism defined around roles, privileges, and permissions. Especially in large companies, using RBAC can help create a hierarchical system of administrators with their individual areas of responsibility.

This chapter describes the following operations performed when managing RBAC permissions in Identity Management (IdM) using Ansible playbooks:

- Using Ansible to ensure an RBAC permission is present
- Using Ansible to ensure an RBAC permission with an attribute is present
- Using Ansible to ensure an RBAC permission is absent
- Using Ansible to ensure an attribute is a member of an IdM RBAC permission
- Using Ansible to ensure an attribute is not a member of an IdM RBAC permission
- Using Ansible to rename an IdM RBAC permission

Prerequisites

- You understand the [concepts and principles of RBAC](#).

23.1. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS PRESENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure a permission is present in IdM so that it can be added to a privilege. The example describes how to ensure the following target state:

- The **MyPermission** permission exists.
- The **MyPermission** permission can only be applied to hosts.
- A user granted a privilege that contains the permission can do all of the following possible operations on an entry:
 - Write
 - Read
 - Search
 - Compare
 - Add
 - Delete

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `permission-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/permission/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-present.yml  
permission-present-copy.yml
```

3. Open the `permission-present-copy.yml` Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the `ipapermission` task section:

- Adapt the `name` of the task to correspond to your use case.
- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the permission.
- Set the `object_type` variable to `host`.
- Set the `right` variable to `all`.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Permission present example  
hosts: ipaserver  
become: true  
  
tasks:  
- name: Ensure that the "MyPermission" permission is present  
ipapermission:  
  ipaadmin_password: Secret123  
  name: MyPermission  
  object_type: host  
  right: all
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-present-copy.yml
```

23.2. USING ANSIBLE TO ENSURE AN RBAC PERMISSION WITH AN ATTRIBUTE IS PRESENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure a permission is present in IdM so that it can be added to a privilege. The example describes how to ensure the following target state:

- The **MyPermission** permission exists.
- The **MyPermission** permission can only be used to add hosts.
- A user granted a privilege that contains the permission can do all of the following possible operations on a host entry:
 - Write
 - Read
 - Search
 - Compare
 - Add
 - Delete
- The host entries created by a user that is granted a privilege that contains the **MyPermission** permission can have a **description** value.



NOTE

The type of attribute that you can specify when creating or modifying a permission is not constrained by the IdM LDAP schema. However, specifying, for example, **attrs: car_licence** if the **object_type** is **host** later results in the **ipa: ERROR: attribute "car-license" not allowed** error message when you try to exercise the permission and add a specific car licence value to a host.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the **~/MyPlaybooks/** directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **permission-present.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks/permission/** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-present.yml
permission-present-with-attribute.yml
```

3. Open the **permission-present-with-attribute.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipapermission** task section:

- Adapt the **name** of the task to correspond to your use case.
- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the permission.
- Set the **object_type** variable to **host**.
- Set the **right** variable to **all**.
- Set the **attrs** variable to **description**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Permission present example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure that the "MyPermission" permission is present with an attribute
      ipapermission:
        ipaadmin_password: Secret123
        name: MyPermission
        object_type: host
        right: all
        attrs: description
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-present-with-attribute.yml
```

Additional resources

- For more information on the IdM schema, see [User and group schema in Linux Domain Identity, Authentication and Policy Guide](#) in RHEL 7.

23.3. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS ABSENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure a permission is absent in IdM so that it cannot be added to a privilege.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `permission-absent.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/permission/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-absent.yml  
permission-absent-copy.yml
```

3. Open the `permission-absent-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipapermission` task section:

- Adapt the `name` of the task to correspond to your use case.
- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the permission.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Permission absent example  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Ensure that the "MyPermission" permission is absent  
      ipapermission:  
        ipaadmin_password: Secret123  
        name: MyPermission  
        state: absent
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-absent-copy.yml
```

23.4. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS A MEMBER OF AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure that an attribute is a member of an RBAC permission in IdM. As a result, a user with the permission can create entries that have the attribute.

The example describes how to ensure that the host entries created by a user with a privilege that contains the **MyPermission** permission can have **gecos** and **description** values.



NOTE

The type of attribute that you can specify when creating or modifying a permission is not constrained by the IdM LDAP schema. However, specifying, for example, **attrs: car_licence** if the **object_type** is **host** later results in the **ipa: ERROR: attribute "car-license" not allowed** error message when you try to exercise the permission and add a specific car licence value to a host.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.
- The **MyPermission** permission exists.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **permission-member-present.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/permission/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-member-present.yml permission-member-present-copy.yml
```

3. Open the **permission-member-present-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **ipapermission** task section:
 - Adapt the **name** of the task to correspond to your use case.
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the permission.
 - Set the **attrs** list to the **description** and **gecos** variables.

- Make sure the **action** variable is set to **member**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Permission member present example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure that the "gecos" and "description" attributes are present in
      "MyPermission"
      ipapermission:
        ipaadmin_password: Secret123
        name: MyPermission
        attrs:
          - description
          - gecos
        action: member
```

5. Save the file.
6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-member-present-copy.yml
```

23.5. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS NOT A MEMBER OF AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure that an attribute is not a member of an RBAC permission in IdM. As a result, when a user with the permission creates an entry in IdM LDAP, that entry cannot have a value associated with the attribute.

The example describes how to ensure the following target state:

- The **MyPermission** permission exists.
- The host entries created by a user with a privilege that contains the **MyPermission** permission cannot have the **description** attribute.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.
- The **MyPermission** permission exists.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `permission-member-absent.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/permission/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-member-absent.yml permission-member-absent-copy.yml
```

3. Open the `permission-member-absent-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipapermission` task section:

- Adapt the `name` of the task to correspond to your use case.
- Set the `ipaadmin_password` variable to the password of the IdM administrator.
- Set the `name` variable to the name of the permission.
- Set the `attrs` variable to `description`.
- Set the `action` variable to `member`.
- Make sure the `state` variable is set to `absent`

This is the modified Ansible playbook file for the current example:

```
---
- name: Permission absent example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure that an attribute is not a member of "MyPermission"
      ipapermission:
        ipaadmin_password: Secret123
        name: MyPermission
        attrs: description
        action: member
        state: absent
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-member-absent-copy.yml
```

23.6. USING ANSIBLE TO RENAME AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control.

The following procedure describes how to use an Ansible playbook to rename a permission. The example describes how to rename **MyPermission** to **MyNewPermission**.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible control node.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.
- The **MyPermission** exists in IdM.
- The **MyNewPermission** does not exist in IdM.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the **permission-renamed.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/permission/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-renamed.yml  
permission-renamed-copy.yml
```

3. Open the **permission-renamed-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipapermission** task section:

- Adapt the **name** of the task to correspond to your use case.
- Set the **ipaadmin_password** variable to the password of the IdM administrator.
- Set the **name** variable to the name of the permission.

This is the modified Ansible playbook file for the current example:

```
---  
- name: Permission present example  
  hosts: ipaserver  
  become: true  
  
  tasks:  
    - name: Rename the "MyPermission" permission  
      ipapermission:  
        ipaadmin_password: Secret123  
        name: MyPermission  
        rename: MyNewPermission  
        state: renamed
```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i inventory permission-renamed-copy.yml
```

23.7. ADDITIONAL RESOURCES

- For more information on the concept of a permission in IdM RBAC, see [Permissions in IdM](#).
- For more information on the concept of a privilege in IdM RBAC, see [Privileges in IdM](#).
- For more sample Ansible playbooks that use the **ipapermission** module, see the **README-permission** file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of the **ipapermission** variables.
- For more sample Ansible playbooks that use the **ipapermission** module, see the **/usr/share/doc/ansible-freeipa/playbooks/ipapermission** directory.

CHAPTER 24. USING AN ID VIEW TO OVERRIDE A USER ATTRIBUTE VALUE ON AN IDM CLIENT

If an Identity Management (IdM) user would like to override some of their user or group attributes stored in the IdM LDAP server, for example the login name, home directory, certificate used for authentication, or **SSH** keys, you as IdM administrator can redefine these values for a specific IdM client, using IdM ID views. For example, you can specify a different home directory for a user on the IdM client that the user most commonly uses for logging in to IdM.

This chapter describes how to redefine a POSIX attribute value associated with an IdM user on a host enrolled into IdM as a client. Specifically, the chapter describes how to redefine the user login name and home directory.

This chapter includes the following sections:

- [ID views](#)
- [Potential negative impact of ID views on SSSD performance](#)
- [Attributes an ID view can override](#)
- [Getting help for ID view commands](#)
- [Using an ID view to override the login name of an IdM user on a specific host](#)
- [Modifying an IdM ID view](#)
- [Adding an ID view to override an IdM user home directory on an IdM client](#)
- [Applying an ID view to an IdM host group](#)

24.1. ID VIEWS

An ID view in Identity Management (IdM) is an IdM client-side view specifying the following information:

- New values for centrally defined POSIX user or group attributes
- The client host or hosts on which the new values apply.

An ID view contains one or more overrides. An override is a specific replacement of a centrally defined POSIX attribute value.

You can only define an ID view for an IdM client centrally on IdM servers. You cannot configure client-side overrides for an IdM client locally.

For example, you can use ID views to achieve the following goals:

- Define different attribute values for different environments. For example, you can allow the IdM administrator or another IdM user to have different home directories on different IdM clients: you can configure **/home/encrypted/username** to be this user's home directory on one IdM client and **/dropbox/username** on another client. Using ID views in this situation is convenient as alternatively, for example, changing **fallback_homedir**, **override_homedir** or other home directory variables in the client's **/etc/sssd/sssd.conf** file would affect all users. See [Adding an ID view to override an IdM user home directory on an IdM client](#) for an example procedure.
- Replace a previously generated attribute value with a different value, such as overriding a user's

UID. This ability can be useful when you want to achieve a system-wide change that would otherwise be difficult to do on the LDAP side, for example make 1009 the UID of an IdM user. IdM ID ranges, which are used to generate an IdM user UID, never start as low as 1000 or even 10000. If a reason exists for an IdM user to impersonate a local user with UID 1009 on all IdM clients, you can use ID views to override the UID of this IdM user that was generated when the user was created in IdM.



IMPORTANT

You can only apply ID views to IdM clients, not to IdM servers.

Additional resources

- You can also use ID views in environments involving Active Directory (AD). For details, see the [ID Views and Migrating Existing Environments to Trust](#) chapter in the *Windows integration guide*.
- You can also configure ID views for hosts that are not part of a centralized identity management domain. For details, see the [SSSD Client-side Views](#) chapter in the *System-level authentication guide*.

24.2. POTENTIAL NEGATIVE IMPACT OF ID VIEWS ON SSSD PERFORMANCE

When you define an ID view, IdM places the desired override value in the IdM server’s System Security Services Daemon (SSSD) cache. The SSSD running on an IdM client then retrieves the override value from the server cache.

Applying an ID view can have a negative impact on System Security Services Daemon (SSSD) performance, because certain optimizations and ID views cannot run at the same time. For example, ID views prevent SSSD from optimizing the process of looking up groups on the server:

- With ID views, SSSD must check every member on the returned list of group member names if the group name is overridden.
- Without ID views, SSSD can only collect the user names from the member attribute of the group object.

This negative effect becomes most apparent when the SSSD cache is empty or after you clear the cache, which makes all entries invalid.

24.3. ATTRIBUTES AN ID VIEW CAN OVERRIDE

ID views consist of user and group ID overrides. The overrides define the new POSIX attribute values.

User and group ID overrides can define new values for the following POSIX attributes:

User attributes

- Login name (**uid**)
- GECOS entry (**gecos**)
- UID number (**uidNumber**)
- GID number (**gidNumber**)

- Login shell (**loginShell**)
- Home directory (**homeDirectory**)
- SSH public keys (**ipaSshPubkey**)
- Certificate (**userCertificate**)

Group attributes

- Group name (**cn**)
- Group GID number (**gidNumber**)

24.4. GETTING HELP FOR ID VIEW COMMANDS

You can get help for commands involving Identity Management (IdM) ID views on the IdM command-line interface (CLI).

Prerequisites

- You have obtained a Kerberos ticket for an IdM user.

Procedure

- To display all commands used to manage ID views and overrides:

```
$ ipa help idviews
ID Views

Manage ID Views

IPA allows to override certain properties of users and groups[...]
[...]
Topic commands:
  idoverridegroup-add      Add a new Group ID override
  idoverridegroup-del      Delete a Group ID override
[...]
```

- To display detailed help for a particular command, add the **--help** option to the command:

```
$ ipa idview-add --help
Usage: ipa [global-options] idview-add NAME [options]

Add a new ID View.
Options:
  -h, --help    show this help message and exit
  --desc=STR   Description
[...]
```

24.5. USING AN ID VIEW TO OVERRIDE THE LOGIN NAME OF AN IDM USER ON A SPECIFIC HOST

This section describes how you as an Identity Management (IdM) system administrator can create an ID view for a specific IdM client that overrides a POSIX attribute value associated with a specific IdM user. The procedure uses the example of an ID view that enables an IdM user named **idm_user** to log in to an IdM client named **host1** using the **user_1234** login name.

Prerequisites

- You are logged in as IdM administrator.

Procedure

1. Create a new ID view. For example, to create an ID view named **example_for_host1**:

```
$ ipa idview-add example_for_host1
```

Added ID View "example_for_host1"

ID View Name: example_for_host1

2. Add a user override to the **example_for_host1** ID view. To override the user login:

- Enter the **ipa idoverrideuser-add** command
- Add the name of the ID view
- Add the user name, also called the anchor
- Add the **--login** option:

```
$ ipa idoverrideuser-add example_for_host1 idm_user --login=user_1234
```

Added User ID override "idm_user"

Anchor to override: idm_user

User login: user_1234

For a list of the available options, run `ipa idoverrideuser-add --help`.



NOTE

The **ipa idoverrideuser-add --certificate** command replaces all existing certificates for the account in the specified ID view. To append an additional certificate, use the **ipa idoverrideuser-add-cert** command instead:

```
$ ipa idoverrideuser-add-cert example_for_host1 user --certificate="MIIEATCC..."
```

3. Optional: Using the **ipa idoverrideuser-mod** command, you can specify new attribute values for an existing user override.
4. Apply **example_for_host1** to the **host1.idm.example.com** host:

```
$ ipa idview-apply example_for_host1 --hosts=host1.idm.example.com
```

```
Applied ID View "example_for_host1"
-----
```

```
hosts: host1.idm.example.com
-----
```

```
Number of hosts the ID View was applied to: 1
-----
```



NOTE

The **ipa idview-apply** command also accepts the **--hostgroups** option. The option applies the ID view to hosts that belong to the specified host group, but does not associate the ID view with the host group itself. Instead, the **--hostgroups** option expands the members of the specified host group and applies the **--hosts** option individually to every one of them.

This means that if a host is added to the host group in the future, the ID view does not apply to the new host.

5. To apply the new configuration to the **host1.idm.example.com** system immediately:

- a. SSH to the system as root:

```
$ ssh root@host1
Password:
```

- b. Clear the SSSD cache:

```
root@host1 ~]# sss_cache -E
```

- c. Restart the SSSD daemon:

```
root@host1 ~]# systemctl restart sssd
```

Verification steps

- If you have the credentials of **user_1234**, you can use them to log in to IdM on **host1**:

1. SSH to **host1** using **user_1234** as the login name:

```
[root@r8server ~]# ssh user_1234@host1.idm.example.com
Password:
```

```
Last login: Sun Jun 21 22:34:25 2020 from 192.168.122.229
[user_1234@host1 ~]$
```

2. Display the working directory:

```
[user_1234@host1 ~]$ pwd
/home/idm_user/
```

- Alternatively, if you have root credentials on **host1**, you can use them to check the output of the **id** command for **idm_user** and **user_1234**:

```
[root@host1 ~]# id idm_user
uid=779800003(user_1234) gid=779800003(idm_user) groups=779800003(idm_user)
[root@host1 ~]# user_1234
uid=779800003(user_1234) gid=779800003(idm_user) groups=779800003(idm_user)
```

24.6. MODIFYING AN IDM ID VIEW

An ID view in Identity Management (IdM) overrides a POSIX attribute value associated with a specific IdM user. This section describes how to modify an existing ID view. Specifically, it describes how to modify an ID view to enable the user named **idm_user** to use the **/home/user_1234** directory as the user home directory instead of **/home/idm_user** on the **host1.idm.example.com** IdM client.

Prerequisites

- You have root access to **host1.idm.example.com**.
- You are logged in as a user with the required privileges, for example **admin**.
- You have an ID view configured for **idm_user** that applies to the **host1** IdM client.

Procedure

1. As root, create the directory that you want **idm_user** to use on **host1.idm.example.com** as the user home directory:

```
[root@host1 ~]# mkdir /home/user_1234/
```

2. Change the ownership of the directory:

```
[root@host1 ~]# chown idm_user:idm_user /home/user_1234/
```

3. Display the ID view, including the hosts to which the ID view is currently applied. To display the ID view named **example_for_host1**:

```
$ ipa idview-show example_for_host1 --all
dn: cn=example_for_host1,cn=views,cn=accounts,dc=idm,dc=example,dc=com
ID View Name: example_for_host1
User object override: idm_user
Hosts the view applies to: host1.idm.example.com
objectclass: ipaIDView, top, nsContainer
```

The output shows that the ID view currently applies to **host1.idm.example.com**.

4. Modify the user override of the **example_for_host1** ID view. To override the user home directory:

- Enter the **ipa idoverrideuser-add** command
- Add the name of the ID view
- Add the user name, also called the anchor
- Add the **--homedir** option:

```
$ ipa idoverrideuser-mod example_for_host1 idm_user --
homedir=/home/user_1234
-----
Modified an User ID override "idm_user"
-----
Anchor to override: idm_user
User login: user_1234
Home directory: /home/user_1234/
```

For a list of the available options, run **ipa idoverrideuser-mod --help**.

5. To apply the new configuration to the **host1.idm.example.com** system immediately:

- a. SSH to the system as root:

```
$ ssh root@host1
Password:
```

- b. Clear the SSSD cache:

```
root@host1 ~]# sss_cache -E
```

- c. Restart the SSSD daemon:

```
root@host1 ~]# systemctl restart sssd
```

Verification steps

1. **SSH** to **host1** as **idm_user**:

```
[root@r8server ~]# ssh idm_user@host1.idm.example.com
Password:
Last login: Sun Jun 21 22:34:25 2020 from 192.168.122.229
[user_1234@host1 ~]$
```

2. Print the working directory:

```
[user_1234@host1 ~]$ pwd
/home/user_1234/
```

24.7. ADDING AN ID VIEW TO OVERRIDE AN IDM USER HOME DIRECTORY ON AN IDM CLIENT

An ID view in Identity Management (IdM) overrides a POSIX attribute value associated with a specific IdM user. This section describes how to create an ID view that applies to **idm_user** on an IdM client named **host1** to enable the user to use the **/home/user_1234/** directory as the user home directory instead of **/home/idm_user/**.

Prerequisites

- You have root access to **host1.idm.example.com**.

- You are logged in as a user with the required privileges, for example **admin**.

Procedure

1. As root, create the directory that you want **idm_user** to use on **host1.idm.example.com** as the user home directory:

```
[root@host1 /]# mkdir /home/user_1234/
```

2. Change the ownership of the directory:

```
[root@host1 /]# chown idm_user:idm_user /home/user_1234/
```

3. Create an ID view. For example, to create an ID view named **example_for_host1**:

```
$ ipa idview-add example_for_host1
```

```
-----  
Added ID View "example_for_host1"
```

```
-----  
ID View Name: example_for_host1
```

4. Add a user override to the **example_for_host1** ID view. To override the user home directory:

- Enter the **ipa idoverrideuser-add** command
- Add the name of the ID view
- Add the user name, also called the anchor
- Add the **--homedir** option:

```
$ ipa idoverrideuser-add example_for_host1 idm_user --homedir=/home/user_1234
```

```
-----  
Added User ID override "idm_user"
```

```
-----  
Anchor to override: idm_user
```

```
Home directory: /home/user_1234/
```

5. Apply **example_for_host1** to the **host1.idm.example.com** host:

```
$ ipa idview-apply example_for_host1 --hosts=host1.idm.example.com
```

```
-----  
Applied ID View "example_for_host1"
```

```
-----  
hosts: host1.idm.example.com
```

```
-----  
Number of hosts the ID View was applied to: 1
```



NOTE

The **ipa idview-apply** command also accepts the **--hostgroups** option. The option applies the ID view to hosts that belong to the specified host group, but does not associate the ID view with the host group itself. Instead, the **--hostgroups** option expands the members of the specified host group and applies the **--hosts** option individually to every one of them.

This means that if a host is added to the host group in the future, the ID view does not apply to the new host.

6. To apply the new configuration to the **host1.idm.example.com** system immediately:

- a. SSH to the system as root:

```
$ ssh root@host1
```

Password:

- b. Clear the SSSD cache:

```
root@host1 ~]# sss_cache -E
```

- c. Restart the SSSD daemon:

```
root@host1 ~]# systemctl restart sssd
```

Verification steps

1. **SSH** to **host1** as **idm_user**:

```
[root@r8server ~]# ssh idm_user@host1.idm.example.com
```

Password:

Activate the web console with: systemctl enable --now cockpit.socket

```
Last login: Sun Jun 21 22:34:25 2020 from 192.168.122.229
```

```
[idm_user@host1 /]$
```

2. Print the working directory:

```
[idm_user@host1 /]$ pwd
```

```
/home/user_1234/
```

24.8. APPLYING AN ID VIEW TO AN IDM HOST GROUP

The **ipa idview-apply** command accepts the **--hostgroups** option. However, the option acts as a one-time operation that applies the ID view to hosts that currently belong to the specified host group, but does not dynamically associate the ID view with the host group itself. The **--hostgroups** option expands the members of the specified host group and applies the **--hosts** option individually to every one of them.

If you add a new host to the host group later, you must apply the ID view to the new host manually, using the **ipa idview-apply** command with the **--hosts** option.

Similarly, if you remove a host from a host group, the ID view is still assigned to the host after the removal. To unapply the ID view from the removed host, you must run the **ipa idview-unapply *id_view_name* --hosts=*name_of_the_removed_host*** command.

This section describes how to achieve the following goals:

1. How to create a host group and add hosts to it.
2. How to apply an ID view to the host group.
3. How to add a new host to the host group and apply the ID view to the new host.

Prerequisites

- Ensure that the ID view you want to apply to the host group exists in IdM. For example, to create an ID view to override an IdM user login name on a specific IdM client, see [Using an ID view to override the login name of an IdM user on a specific host](#).

Procedure

1. Create a host group and add hosts to it:

- a. Create a host group. For example, to create a host group named **baltimore**:

```
[root@server ~]# ipa hostgroup-add --desc="Baltimore hosts" baltimore
-----
Added hostgroup "baltimore"
-----
Host-group: baltimore
Description: Baltimore hosts
```

- b. Add hosts to the host group. For example, to add the **host102** and **host103** to the **baltimore** host group:

```
[root@server ~]# ipa hostgroup-add-member --hosts={host102,host103} baltimore
Host-group: baltimore
Description: Baltimore hosts
Member hosts: host102.idm.example.com, host103.idm.example.com
-----
Number of members added 2
```

2. Apply an ID view to the hosts in the host group. For example, to apply the **example_for_host1** ID view to the **baltimore** host group:

```
[root@server ~]# ipa idview-apply --hostgroups=baltimore
ID View Name: example_for_host1
-----
Applied ID View "example_for_host1"
-----
hosts: host102.idm.example.com, host103.idm.example.com
-----
Number of hosts the ID View was applied to: 2
```

3. Add a new host to the host group and apply the ID view to the new host:
 - a. Add a new host to the host group. For example, to add the **somehost.idm.example.com** host to the **baltimore** host group:

```
[root@server ~]# ipa hostgroup-add-member --hosts=somehost.idm.example.com baltimore
Host-group: baltimore
Description: Baltimore hosts
Member hosts: host102.idm.example.com,
host103.idm.example.com,somehost.idm.example.com
-----
Number of members added 1
-----
```

- a. Optionally, display the ID view information. For example, to display the details about the **example_for_host1** ID view:

```
[root@server ~]# ipa idview-show example_for_host1 --all
dn: cn=example_for_host1,cn=views,cn=accounts,dc=idm,dc=example,dc=com
ID View Name: example_for_host1
...
Hosts the view applies to: host102.idm.example.com, host103.idm.example.com
objectclass: ipalDView, top, nsContainer
```

The output shows that the ID view is not applied to **somehost.idm.example.com**, the newly-added host in the **baltimore** host group.

- c. Apply the ID view to the new host. For example, to apply the **example_for_host1** ID view to **somehost.idm.example.com**:

```
[root@server ~]# ipa idview-apply --host=somehost.idm.example.com
ID View Name: example_for_host1
-----
Applied ID View "example_for_host1"
-----
hosts: somehost.idm.example.com
-----
Number of hosts the ID View was applied to: 1
-----
```

Verification steps

- Display the ID view information again:

```
[root@server ~]# ipa idview-show example_for_host1 --all
dn: cn=example_for_host1,cn=views,cn=accounts,dc=idm,dc=example,dc=com
ID View Name: example_for_host1
...
Hosts the view applies to: host102.idm.example.com, host103.idm.example.com,
somehost.idm.example.com
objectclass: ipalDView, top, nsContainer
```

The output shows that ID view is now applied to **somehost.idm.example.com**, the newly-added host in the **baltimore** host group.

CHAPTER 25. ADJUSTING ID RANGES MANUALLY

An IdM server generates unique user ID (UID) and group ID (GID) numbers. By creating and assigning different ID ranges to replicas, it also ensures that they never generate the same ID numbers. By default, this process is automatic. However, you can manually adjust the IdM ID range during the IdM server installation, or manually define a replica's DNA ID range.

25.1. ID RANGES

ID numbers are divided into *ID ranges*. Keeping separate numeric ranges for individual servers and replicas eliminates the chance that an ID number issued for an entry is already used by another entry on another server or replica.

Note that there are two distinct types of ID ranges:

- The IdM ***ID range***, which is assigned during the installation of the first server. This range cannot be modified after it is created. However, if you need to, you can create a new IdM ID range in addition to the original one. For more information, see [Automatic ID ranges assignment](#) and [Adding a new IdM ID range](#).
- The ***Distributed Numeric Assignment*** (DNA) ID ranges, which can be modified by the user. These have to fit within an existing IdM ID range. For more information, see [Adjusting DNA ID ranges manually](#).

Replicas can also have a **next** DNA ID range assigned. A replica uses its next range when it runs out of IDs in its current range. Next ranges are [assigned automatically](#) when a replica is deleted or you can [set them manually](#).

The ranges are updated and shared between the server and replicas by the DNA plug-in, as part of the back end 389 Directory Server instance for the domain.

The DNA range definition is set by two attributes: the server's next available number (the low end of the DNA range) and its maximum value (the top end of the DNA range). The initial bottom range is set during the plug-in instance configuration. After that, the plug-in updates the bottom value. Breaking the available numbers into ranges allows the servers to continually assign numbers without overlapping with each other.

25.2. AUTOMATIC ID RANGES ASSIGNMENT

By default, an IdM ID range is automatically assigned during the IdM server installation. The **ipa-server-install** command randomly selects and assigns a range of 200,000 IDs from a total of 10,000 possible ranges. Selecting a random range in this way significantly reduces the probability of conflicting IDs in case you decide to merge two separate IdM domains in the future.



NOTE

This IdM ID range cannot be modified after it is created. You can only manually adjust the Distributed Numeric Assignment (DNA) ID ranges, using the commands described in [Adjusting DNA ID ranges manually](#). A DNA range matching the IdM ID range is automatically created during installation.

If you have a single IdM server installed, it controls the whole DNA ID range. When you install a new replica and the replica requests its own DNA ID range, the initial ID range for the server splits and is distributed between the server and replica: the replica receives half of the remaining DNA ID range that is available on the initial server. The server and replica then use their respective portions of the original

ID range for new user or group entries. Also, if the replica is close to depleting its allocated ID range and fewer than 100 IDs remain, the replica contacts the other available servers to request a new DNA ID range.



IMPORTANT

When you install a replica, it **does not** immediately receive an ID range. A replica receives an ID range the first time the DNA plug-in is used, for example when you first add a user. Until then, the replica has no ID range defined.

If the initial server stops functioning before the replica requests a DNA ID range from it, the replica is unable to contact the server to request the ID range. Attempting to add a new user on the replica then fails. In such situations, [you can find out what ID range is assigned to the disabled server](#), and [assign an ID range to the replica manually](#).

25.3. ASSIGNING THE IDM ID RANGE MANUALLY DURING SERVER INSTALLATION

You can override the default behavior and set an IdM ID range manually instead of having it assigned randomly.



IMPORTANT

Do not set ID ranges that include UID values of 1000 and lower; these values are reserved for system use. Also, do not set an ID range that would include the 0 value; the SSSD service does not handle the 0 ID value.

Procedure

- You can define the IdM ID range manually during server installation by using the following two options with **ipa-server-install**:
 - **--idstart** gives the starting value for UID and GID numbers.
 - **--idmax** gives the maximum UID and GID number; by default, the value is the **--idstart** starting value plus 199,999.

Verification steps

- To check if the ID range was assigned correctly, you can display the assigned IdM ID range by using the **ipa idrange-find** command:

```
# ipa idrange-find
-----
1 range matched
-----
Range name: IDM.EXAMPLE.COM_id_range
First Posix ID of the range: 882200000
Number of IDs in the range: 200000
Range type: local domain range
-----
Number of entries returned 1
-----
```

25.4. ADDING A NEW IDM ID RANGE

In some cases, you may want to create a new IdM ID range in addition to the original one; for example, when a replica has run out of IDs and the original IdM ID range is depleted.



IMPORTANT

Adding a new IdM ID range does not create new DNA ID ranges automatically. You need to assign new DNA ID ranges manually as needed. For more information on how to do this, see [Adjusting DNA ID ranges manually](#).

Procedure

- To create a new IdM ID range, use the **ipa idrange-add** command. You need to specify the new range name, the first ID number of the range and the range size:

```
# ipa idrange-add IDM.EXAMPLE.COM_new_range --base-id=1000000 --range-size=200000
-----
Added ID range "IDM.EXAMPLE.COM_new_range"
-----
Range name: IDM.EXAMPLE.COM_new_range
First Posix ID of the range: 1000000
Number of IDs in the range: 200000
Range type: local domain range
```

- Optional: Update the ID range immediately:

- Clear the System Security Services Daemon (SSSD) cache:

```
# sss_cache -E
```

- Restart the SSSD daemon:

```
# systemctl restart sssd
```



NOTE

If you do not clear the SSSD cache and restart the service, it takes some time for SSSD to notice the new ID range. More specifically, it notices the range when it updates the domain list and other configuration data stored on the IdM server.

Verification steps

- You can check if the new range is set correctly by using the **ipa idrange-find** command:

```
# ipa idrange-find
-----
2 ranges matched
-----
Range name: IDM.EXAMPLE.COM_id_range
First Posix ID of the range: 882200000
Number of IDs in the range: 200000
```

Range type: local domain range

Range name: IDM.EXAMPLE.COM_new_range

First Posix ID of the range: 1000000

Number of IDs in the range: 200000

Range type: local domain range

Number of entries returned 2

25.5. REMOVING AN ID RANGE AFTER REMOVING A TRUST TO AD

If you have removed a trust between your IdM and Active Directory (AD) environments, you might want to remove the ID range associated with it.



WARNING

IDs allocated to ID ranges associated with trusted domains might still be used for ownership of files and directories on systems enrolled into IdM.

If you remove the ID range that corresponds to an AD trust that you have removed, you will not be able to resolve the ownership of any files and directories owned by AD users.

Prerequisites

- You have removed a trust to an AD environment.

Procedure

1. Display all the ID ranges that are currently in use:

```
[root@server ~]# ipa idrange-find
```

2. Identify the name of the ID range associated with the trust you have removed. The first part of the name of the ID range is the name of the trust, for example **AD.EXAMPLE.COM_id_range**.

3. Remove the range:

```
[root@server ~]# ipa idrange-del AD.EXAMPLE.COM_id_range
```

4. Restart the SSSD service to remove references to the ID range you have removed.

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

Additional resources

- For steps on removing a trust to AD from the command line, see [Removing the trust using the command line](#).

- For steps on removing a trust to AD from the IdM Web UI, see [Removing the trust using the IdM Web UI](#).

25.6. DISPLAYING CURRENTLY ASSIGNED DNA ID RANGES

You can display both the currently active Distributed Numeric Assignment (DNA) ID range on a server, as well as its next DNA range if it has one assigned.

Procedure

- To display which DNA ID ranges are configured for the servers in the topology, use the following commands:
 - ipa-replica-manage dnarange-show** displays the current DNA ID range that is set on all servers or, if you specify a server, only on the specified server, for example:

```
# ipa-replica-manage dnarange-show
serverA.example.com: 1001-1500
serverB.example.com: 1501-2000
serverC.example.com: No range set

# ipa-replica-manage dnarange-show serverA.example.com
serverA.example.com: 1001-1500
```

- ipa-replica-manage dnanextrange-show** displays the next DNA ID range currently set on all servers or, if you specify a server, only on the specified server, for example:

```
# ipa-replica-manage dnanextrange-show
serverA.example.com: 2001-2500
serverB.example.com: No on-deck range set
serverC.example.com: No on-deck range set

# ipa-replica-manage dnanextrange-show serverA.example.com
serverA.example.com: 2001-2500
```

25.7. AUTOMATIC DNA ID RANGE EXTENSION

When you are deleting a RHEL 8 Identity Management (IdM) replica, the ID range that was assigned to the replica is not, by default, transferred to any other remaining replica in the topology. If you want to use the ID range, you must transfer it to another replica manually.

Procedure

In this example we want to delete **serverB.example.com** and preserve its ID range by transferring it to **serverC.example.com**.

- To check what range is currently assigned to **serverB.example.com**, use the **ipa-replica-manage dnarange-show** command.

```
# ipa-replica-manage dnarange-show
serverA.example.com: 1001-1500
serverB.example.com: 1501-2000
serverC.example.com: No range set
```

The output shows that **serverB.example.com** has the 1501-2000 ID range assigned to it.



NOTE

If you want to check the next ID range set for a server, use **ipa-replica-manage dnanager-show** instead.

2. Assign the ID range to **serverC.example.com** using the **ipa-replica-manage dnarange-set** command.

```
# ipa-replica-manage dnarange-set serverC.example.com 1501-2000
```

Verification steps

1. Delete **serverB.example.com**:

```
# ipa-server-del serverB.example.com
```

2. Verify which DNA ID range is set for the serverC.example.com:

```
# ipa-replica-manage dnarange-show serverC.example.com
serverC.example.com: 1501-2000
```

25.8. MANUAL DNA ID RANGE ADJUSTMENT

In certain situations, it is necessary to manually adjust a Distributed Numeric Assignment (DNA) ID range, for example when:

- A replica has run out of IDs and the IdM ID range is depleted
A replica has exhausted the DNA ID range that was assigned to it, and requesting additional IDs failed because no more free IDs are available in the IdM range.

To solve this situation, extend the DNA ID range assigned to the replica. You can do this in two ways:

- Shorten the DNA ID range assigned to a different replica, then assign the newly available values to the depleted replica.
- Create a new IdM ID range, then set a new DNA ID range for the replica within this created IdM range.
For information on how to create a new IdM ID range, see [Adding a new IdM ID range](#).

- A replica stopped functioning
A replica's DNA ID range is not automatically retrieved when the replica dies and needs to be deleted, which means the DNA ID range previously assigned to the replica becomes unavailable. You want to recover the DNA ID range and make it available for other replicas.

If you want to recover a DNA ID range belonging to a replica that stopped functioning and assign it to another server, you first need to [find out what the ID range values are](#), before manually assigning that range to a different server. Also, to avoid duplicate UIDs or GIDs, make sure that no ID value from the recovered range was previously assigned to a user or group; you can do this by examining the UIDs and GIDs of existing users and groups.

You can manually adjust a DNA ID range for a replica using the commands in [Adjusting DNA ID ranges manually](#).



NOTE

If you assign a new DNA ID range, the UIDs of the already existing entries on the server or replica stay the same. This does not pose a problem because even if you change the current DNA ID range, IdM keeps a record of what ranges were assigned in the past.

25.9. ADJUSTING DNA ID RANGES MANUALLY

In some cases, you may need to manually adjust Distributed Numeric Assignment (DNA) ID ranges for existing replicas, for example to reassign a DNA ID range assigned to a non-functioning replica. For more information, see [Manual DNA ID range adjustment](#).

When adjusting a DNA ID range manually, make sure that the newly adjusted range is included in the IdM ID range; you can check this using the **ipa idrange-find** command. Otherwise, the command will fail.



IMPORTANT

Be careful not to create overlapping ID ranges. If any of the ID ranges you assign to servers or replicas overlap, it could result in two different servers assigning the same ID value to different entries.

Prerequisites

- *Optional.* If you are recovering a DNA ID range from a non-functioning replica, first find the ID range using the commands described in [Displaying currently assigned DNA ID ranges](#).

Procedure

- To define the current DNA ID range for a specified server, use the **ipa-replica-manage dnarange-set**:

```
# ipa-replica-manage dnarange-set serverA.example.com 1250-1499
```
- To define the next DNA ID range for a specified server, use the **ipa-replica-manage dnanextrange-set**:

```
# ipa-replica-manage dnanextrange-set serverB.example.com 1500-5000
```

Verification steps

- You can check that the new DNA ranges are set correctly by using the commands described in [Displaying the currently assigned DNA ID ranges](#).

CHAPTER 26. MANAGING HOSTS IN IDM CLI

This chapter introduces [hosts](#) and [host entries](#) in Identity Management (IdM), and the following operations performed when managing hosts and host entries in IdM CLI:

- [Host Enrollment](#)
- [Adding IdM host entries](#)
- [Deleting IdM host entries](#)
- [Re-enrolling hosts](#)
- [Renaming hosts](#)
- [Disabling hosts](#)
- [Re-enabling hosts](#)

The chapter also contains an [overview table](#) of the prerequisites, the context, and the consequences of these operations.

26.1. HOSTS IN IDM

Identity Management (IdM) manages these identities:

- Users
- Services
- Hosts

A host represents a machine. As an IdM identity, a host has an entry in the IdM LDAP, that is the 389 Directory Server instance of the IdM server.

The host entry in IdM LDAP 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. Any host can be used in **host-based access control** (HBAC) rules.

IdM domain establishes a commonality between machines, with common identity information, common policies, and shared services. Any machine that belongs to a domain functions as a client of the domain, which means it uses the services that the domain provides. IdM domain provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

Hosts in IdM are closely connected with the services running on them:

- Service entries are associated with a host.
- A host stores both the host and the service Kerberos principals.

26.2. HOST ENROLLMENT

This section describes enrolling hosts as IdM clients and what happens during and after the enrollment. The section compares the enrollment of IdM hosts and IdM users. The section also outlines alternative types of authentication available to hosts.

Enrolling a host consists of:

- Creating a host entry in IdM LDAP: possibly using the [ipa host-add command](#) in IdM CLI, or the equivalent [IdM Web UI operation](#).
- Configuring IdM services on the host, for example the System Security Services Daemon (SSSD), Kerberos, and certmonger, and joining the host to the IdM domain.

The two actions can be performed separately or together.

If performed separately, they allow for dividing the two tasks between two users with different levels of privilege. This is useful for bulk deployments.

The **ipa-client-install** command can perform the two actions together. The command creates a host entry in IdM LDAP if that entry does not exist yet, and configures both the Kerberos and SSSD services for the host. The command brings the host within the IdM domain and allows it to identify the IdM server it will connect with. If the host belongs to a DNS zone managed by IdM, **ipa-client-install** adds DNS records for the host too. The command must be run on the client.

26.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running **ipa-client-install**
- If a one-time password (OTP) is used for enrollment

User privileges for optionally manually creating a host entry in IdM LDAP

The user privilege required for creating a host entry in IdM LDAP using the **ipa host-add** CLI command or the IdM Web UI is **Host Administrators**. The **Host Administrators** privilege can be obtained through the **IT Specialist** role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the **ipa-client-install** command. The level of credentials required for executing the **ipa-client-install** command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator's credentials or the **Host Administrators** role. A full administrator is a member of the **admins** group. The **Host Administrators** role provides privileges to add hosts and enroll hosts. For details about this scenario, see [Installing a client using user credentials: interactive installation](#).
- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator's credentials to execute **ipa-client-install** successfully. The limited administrator in this case has the **Enrollment Administrator** role, which provides the **Host Enrollment** privilege. For details, see [Installing a client using user credentials: interactive installation](#).
- The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the **ipa-client-install** command with the **--password** option, supplying the correct OTP. For

details, see [Installing a client by using a one-time password: Interactive installation](#).

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

26.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- The enrollment stage ([Table 26.1, “User and host enrollment”](#)):
 - An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is **ipa stageuser-add**; for the host, the command is **ipa host-add**.
 - A file containing a *key table* or, abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the **ipa-client-install** command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.
 - While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

Table 26.1. User and host enrollment

Action	User	Host
Pre-enrollment	\$ ipa stageuser-add <i>user_name</i> [-password]	\$ ipa host-add <i>host_name</i> [--random]
Activating the account	\$ ipa stageuser-activate <i>user_name</i>	\$ ipa-client install [--password] (must be run on the host itself)

- The deployment stage ([Table 26.2, “User and host session authentication”](#)):
 - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
 - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
 - The TGT is then used to obtain specific tickets for specific services.

Table 26.2. User and host session authentication

	User	Host
Default means of authentication	Password	Keytabs

User		Host
Starting a session (ordinary user)	\$ <code>kinit user_name</code>	[switch on the host]
The result of successful authentication	TGT to be used to obtain access to specific services	TGT to be used to obtain access to specific services

TGTs and other Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. The initial granting of a Kerberos ticket, the renewing of the Kerberos credentials, and even the destroying of the Kerberos session are all handled automatically by the IdM services.

26.2.3. Alternative authentication options for IdM hosts

Apart from keytabs, IdM supports two other types of machine authentication:

- SSH keys. The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.
- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the IdM server's certificate authority and then stored in IdM's Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called `certmonger`.

26.3. HOST OPERATIONS

This section lists the most common operations related to host enrollment and enablement, and explains the prerequisites, the context, and the consequences of performing them.

Table 26.3. Host operations part 1

Action	What are the prerequisites of the action?	When does it make sense to run the command?	How is the action performed by a system administrator? What command(s) does he run?
Enrolling a client	see Preparing the system for Identity Management client installation in Installing_Identity_Management	When you want the host to join the IdM realm.	Enrolling machines as clients in the IdM domain is a two-part process. A host entry is created for the client (and stored in the 389 Directory Server instance) when the ipa host-add command is run, and then a keytab is created to provision the client. Both parts are performed automatically by the ipa-client-install command. It is also possible to perform those steps separately; this allows for administrators to prepare machines and IdM in advance of actually configuring the clients. This allows more flexible setup scenarios, including bulk deployments.

Action	What are the prerequisites of the action?	When does it make sense to run the command?	How is the action performed by a system administrator? What command(s) does he run?
Disabling a client	The host must have an entry in IdM. The host needs to have an active keytab.	When you want to remove the host from the IdM realm temporarily, perhaps for maintenance purposes.	ipa host-disable host_name
Enabling a client	The host must have an entry in IdM.	When you want the temporarily disabled host to become active again.	ipa-getkeytab
Re-enrolling a client	The host must have an entry in IdM.	When the original host has been lost but you have installed a host with the same host name.	ipa-client-install --keytab or ipa-client-install --force-join
Un-enrolling a client	The host must have an entry in IdM.	When you want to remove the host from the IdM realm permanently.	ipa-client-install --uninstall

Table 26.4. Host operations part 2

Action	On which machine can the administrator run the command(s)?	What happens when the action is performed? What are the consequences for the host's functioning in IdM? What limitations are introduced/removed?
--------	------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

Action	On which machine can the administrator run the command(s)?	What happens when the action is performed? What are the consequences for the host's functioning in IdM? What limitations are introduced/removed?
Enrolling a client	In the case of a two-step enrollment: ipa host-add can be run on any IdM client; the second step of ipa-client-install must be run on the client itself	By default this configures SSSD to connect to an IdM server for authentication and authorization. Optionally one can instead configure the Pluggable Authentication Module (PAM) and the Name Switching Service (NSS) to work with an IdM server over Kerberos and LDAP.
Disabling a client	Any machine in IdM, even the host itself	The host's Kerberos key and SSL certificate are invalidated, and all services running on the host are disabled.
Enabling a client	Any machine in IdM. If run on the disabled host, LDAP credentials need to be supplied.	The host's Kerberos key and the SSL certificate are made valid again, and all IdM services running on the host are re-enabled.
Re-enrolling a client	The host to be re-enrolled. LDAP credentials need to be supplied.	A new Kerberos key is generated for the host, replacing the previous one.
Un-enrolling a client	The host to be un-enrolled.	The command unconfigures IdM and attempts to return the machine to its previous state. Part of this process is to unenroll the host from the IdM server. Unenrollment consists of disabling the principal key on the IdM server. The machine principal in /etc/krb5.keytab (host/<fqdn>@REALM) is used to authenticate to the IdM server to unenroll itself. If this principal does not exist then unenrollment will fail and an administrator will need to disable the host principal (ipa host-disable <fqdn>).

26.4. HOST ENTRY IN IDM LDAP

This section describes what a host entry in Identity Management (IdM) looks like and what attributes it can contain.

An LDAP host entry contains all relevant information about the client within IdM:

- Service entries associated with the host
- The host and service principal

- Access control rules
- Machine information, such as its physical location and operating system



NOTE

Note that the IdM Web UI **Identity → Hosts** tab does not show all the information about a particular host stored in the IdM LDAP.

26.4.1. Host entry configuration properties

A host entry can contain information about the host that is outside its system configuration, such as its physical location, MAC address, keys, and certificates.

This information can be set when the host entry is created if it is created manually. Alternatively, most of this information can be added to the host entry after the host is enrolled in the domain.

Table 26.5. Host Configuration Properties

UI Field	Command-Line Option	Description
Description	--desc = <i>description</i>	A description of the host.
Locality	--locality = <i>locality</i>	The geographic location of the host.
Location	--location = <i>location</i>	The physical location of the host, such as its data center rack.
Platform	--platform = <i>string</i>	The host hardware or architecture.
Operating system	--os = <i>string</i>	The operating system and version for the host.
MAC address	--macaddress = <i>address</i>	The MAC address for the host. This is a multi-valued attribute. The MAC address is used by the NIS plug-in to create a NIS ethers map for the host.
SSH public keys	--sshpubkey = <i>string</i>	The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.

UI Field	Command-Line Option	Description
Principal name (not editable)	--principalname = <i>principal</i>	The Kerberos principal name for the host. This defaults to the host name during the client installation, unless a different principal is explicitly set in the -p . This can be changed using the command-line tools, but cannot be changed in the UI.
Set One-Time Password	--password = <i>string</i>	This option sets a password for the host which can be used in bulk enrollment.
-	--random	This option generates a random password to be used in bulk enrollment.
-	--certificate = <i>string</i>	A certificate blob for the host.
-	--updatedns	This sets whether the host can dynamically update its DNS entries if its IP address changes.

26.5. ADDING IDM HOST ENTRIES FROM IDM CLI

This section describes how to add host entries in Identity Management (IdM) using the command-line interface (CLI).

Host entries are created using the **host-add** command. This command adds the host entry to the IdM Directory Server. Consult the **ipa host** manpage by typing **ipa help host** in your CLI to get the full list of options available with **host-add**.

There are a few different scenarios when adding a host to IdM:

- At its most basic, specify only the client host name to add the client to the Kerberos realm and to create an entry in the IdM LDAP server:

```
$ ipa host-add client1.example.com
```

- If the IdM server is configured to manage DNS, add the host to the DNS resource records using the **--ip-address** option.

Example 26.1. Creating Host Entries with Static IP Addresses

```
$ ipa host-add --ip-address=192.168.166.31 client1.example.com
```

- If the host to be added does not have a static IP address or if the IP address is not known at the time the client is configured, use the **--force** option with the **ipa host-add** command.

Example 26.2. Creating Host Entries with DHCP

```
$ ipa host-add --force client1.example.com
```

For example, laptops may be preconfigured as IdM clients, but they do not have IP addresses at the time they are configured. Using **--force** essentially creates a placeholder entry in the IdM DNS service. When the DNS service dynamically updates its records, the host's current IP address is detected and its DNS record is updated.

26.6. DELETING HOST ENTRIES FROM IDM CLI

- Use the **host-del** command to delete host records. If your IdM domain has integrated DNS, use the **--updatedns** option to remove the associated records of any kind for the host from the DNS:

```
$ ipa host-del --updatedns client1.example.com
```

26.7. RE-ENROLLING AN IDENTITY MANAGEMENT CLIENT

26.7.1. Client re-enrollment in IdM

This section describes how to re-enroll an Identity Management (IdM) client.

If a client machine has been destroyed and lost connection with the IdM servers, for example due to the client's hardware failure, and you still have its keytab, you can re-enroll the client. In this scenario, you want to get the client back in the IdM environment with the same hostname.

During the re-enrollment, the client generates a new Kerberos key and SSH keys, but the identity of the client in the LDAP database remains unchanged. After the re-enrollment, the host has its keys and other information in the same LDAP object with the same **FQDN** as previously, before the machine's loss of connection with the IdM servers.



IMPORTANT

You can only re-enroll clients whose domain entry is still active. If you uninstalled a client (using **ipa-client-install --uninstall**) or disabled its host entry (using **ipa host-disable**), you cannot re-enroll it.

You cannot re-enroll a client after you have renamed it. This is because in Identity Management, the key attribute of the client's entry in LDAP is the client's hostname, its **FQDN**. As opposed to re-enrolling a client, during which the client's LDAP object remains unchanged, the outcome of renaming a client is that the client has its keys and other information in a different LDAP object with a new **FQDN**. Thus the only way to rename a client is to uninstall the host from IdM, change the host's hostname, and install it as an IdM client with a new name. For details on how to rename a client, see [Section 26.8, "Renaming Identity Management client systems"](#).

26.7.1.1. What happens during client re-enrollment

During re-enrollment, Identity Management:

- Revokes the original host certificate
- Creates new SSH keys
- Generates a new keytab

26.7.2. Re-enrolling a client by using user credentials: Interactive re-enrollment

This procedure describes re-enrolling an Identity Management client interactively by using the credentials of an authorized user.

1. Re-create the client machine with the same host name.
2. Run the **ipa-client-install --force-join** command on the client machine:

```
# ipa-client-install --force-join
```

3. The script prompts for a user whose identity will be used to re-enroll the client. This could be, for example, a **hostadmin** user with the Enrollment Administrator role:

```
User authorized to enroll computers: hostadmin
Password for hostadmin@EXAMPLE.COM:
```

Additional resources

- For a more detailed procedure on enrolling clients by using an authorized user's credentials, see [Installing a client by using user credentials: Interactive installation](#) in *Installing Identity Management*.

26.7.3. Re-enrolling a client by using the client keytab: Non-interactive re-enrollment

Prerequisites

- Back up the original client keytab file, for example in the **/tmp** or **/root** directory.

Procedure

This procedure describes re-enrolling an Identity Management (IdM) client non-interactively by using the keytab of the client system. For example, re-enrollment using the client keytab is appropriate for an automated installation.

1. Re-create the client machine with the same host name.
2. Copy the keytab file from the backup location to the **/etc/** directory on the re-created client machine.
3. Use the **ipa-client-install** utility to re-enroll the client, and specify the keytab location with the **-keytab** option:

```
# ipa-client-install --keytab /etc/krb5.keytab
```

**NOTE**

The keytab specified in the **--keytab** option is only used when authenticating to initiate the enrollment. During the re-enrollment, IdM generates a new keytab for the client.

26.7.4. Testing an Identity Management client after installation

The Command-Line Interface informs you that the **ipa-client-install** was successful, but you can also do your own test.

To test that the Identity Management client can obtain information about users defined on the server, check that you are able to resolve a user defined on the server. For example, to check the default **admin** user:

```
[user@client1 ~]$ id admin
uid=1254400000(admin) gid=1254400000(admins) groups=1254400000(admins)
```

To test that authentication works correctly, **su** - as another IdM user:

```
[user@client1 ~]$ su - idm_user
Last login: Thu Oct 18 18:39:11 CEST 2018 from 192.168.122.1 on pts/0
[idm_user@client1 ~]$
```

26.8. RENAMING IDENTITY MANAGEMENT CLIENT SYSTEMS

The following sections describe how to change the host name of an Identity Management client system.

**WARNING**

Renaming a client is a manual procedure. Do not perform it unless changing the host name is absolutely required.

Renaming an Identity Management client involves:

1. Preparing the host. For details, see [Preparing an IdM client for its renaming](#).
2. Uninstalling the IdM client from the host. For details, see [Section 26.8.2, “Uninstalling an Identity Management client”](#).
3. Renaming the host. For details, see [Section 26.8.3, “Renaming the host system”](#).
4. Installing the IdM client on the host with the new name. For details, see [Section 26.8.4, “Re-installing an Identity Management client”](#).
5. Configuring the host after the IdM client installation. For details, see [Section 26.8.5, “Re-adding services, re-generating certificates, and re-adding host groups”](#).

26.8.1. Preparing an IdM client for its renaming

Before uninstalling the current client, make note of certain settings for the client. You will apply this configuration after re-enrolling the machine with a new host name.

- Identify which services are running on the machine:
 - Use the **ipa service-find** command, and identify services with certificates in the output:

```
$ ipa service-find old-client-name.example.com
```

- In addition, each host has a default *host* service which does not appear in the **ipa service-find** output. The service principal for the host service, also called a *host principal*, is **host/old-client-name.example.com**.
- For all service principals displayed by **ipa service-find old-client-name.example.com**, determine the location of the corresponding keytabs on the **old-client-name.example.com** system:

```
# find / -name "*.keytab"
```

Each service on the client system has a Kerberos principal in the form *service_name/host_name@REALM*, such as **ldap/old-client-name.example.com@EXAMPLE.COM**.

- Identify all host groups to which the machine belongs.

```
# ipa hostgroup-find old-client-name.example.com
```

26.8.2. Uninstalling an Identity Management client

Uninstalling a client removes the client from the Identity Management domain, along with all of the specific Identity Management configuration of system services, such as System Security Services Daemon (SSSD). This restores the previous configuration of the client system.

Procedure

1. Run the **ipa-client-install --uninstall** command:

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

2. Remove the DNS entries for the client host manually from the server:

```
[root@server]# ipa dnsrecord-del
Record name: old-client-client
Zone name: idm.example.com
No option to delete specific record provided.
Delete all? Yes/No (default No): yes
-----
Deleted record "old-client-name"
```

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

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

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

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

26.8.3. Renaming the host system

Rename the machine as required. For example:

```
[root@client]# hostnamectl set-hostname new-client-name.example.com
```

You can now re-install the Identity Management client to the Identity Management domain with the new host name.

26.8.4. Re-installing an Identity Management client

Install an client on your renamed host following the procedure described in [Installing an Identity Management client: Basic scenario](#) in *Installing Identity Management*.

26.8.5. Re-adding services, re-generating certificates, and re-adding host groups

1. On the Identity Management (IdM) server, add a new keytab for every service identified in the [Preparing an IdM client for its renaming](#) .

```
[root@server ~]# ipa service-add service_name/new-client-name
```

2. Generate certificates for services that had a certificate assigned in the [Preparing an IdM client for its renaming](#). You can do this:

- Using the IdM administration tools
- Using the **certmonger** utility

3. Re-add the client to the host groups identified in the [Preparing an IdM client for its renaming](#) .

26.9. DISABLING AND RE-ENABLING HOST ENTRIES

This section describes how to disable and re-enable hosts in Identity Management (IdM).

26.9.1. Disabling Hosts

Complete this procedure to disable a host entry in IdM.

Domain services, hosts, and users can access an active host. There can be situations when it is necessary to remove an active host temporarily, for maintenance reasons, for example. Deleting the host in such situations is not desired as it removes the host entry and all the associated configuration permanently. Instead, choose the option of disabling the host.

Disabling a host prevents domain users from accessing it without permanently removing it from the domain. This can be done by using the **host-disable** command. Disabling a host kills the host's current, active keytabs.

For example:

```
$ kinit admin
$ ipa host-disable client.example.com
```

As a result of disabling a host, the host becomes unavailable to all IdM users, hosts and services.



IMPORTANT

Disabling a host entry not only disables that host. It disables every configured service on that host as well.

26.9.2. Re-enabling Hosts

This section describes how to re-enable a disabled IdM host.

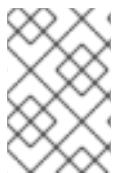
Disabling a host killed its active keytabs, which removed the host from the IdM domain without otherwise touching its configuration entry.

To re-enable a host, use the **ipa-getkeytab** command, adding:

- the **-s** option to specify which IdM server to request the keytab from
- the **-p** option to specify the principal name
- the **-k** option to specify the file to which to save the keytab.

For example, to request a new host keytab from **server.example.com** for **client.example.com**, and store the keytab in the **/etc/krb5.keytab** file:

```
$ ipa-getkeytab -s server.example.com -p host/client.example.com -k /etc/krb5.keytab -D
"cn=directory manager" -w password
```



NOTE

You can also use the administrator's credentials, specifying **-D "uid=admin,cn=users,cn=accounts,dc=example,dc=com"**. It is important that the credentials correspond to a user allowed to create the keytab for the host.

If the **ipa-getkeytab** command is run on an active IdM client or server, then it can be run without any LDAP credentials (**-D** and **-w**) if the user has a TGT obtained using, for example, **kinit admin**. To run the command directly on the disabled host, supply LDAP credentials to authenticate to the IdM server.

CHAPTER 27. ADDING HOST ENTRIES FROM IDM WEB UI

This chapter introduces hosts in Identity Management (IdM) and the operation of adding a host entry in the IdM Web UI.

27.1. HOSTS IN IDM

Identity Management (IdM) manages these identities:

- Users
- Services
- Hosts

A host represents a machine. As an IdM identity, a host has an entry in the IdM LDAP, that is the 389 Directory Server instance of the IdM server.

The host entry in IdM LDAP 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. Any host can be used in **host-based access control** (HBAC) rules.

IdM domain establishes a commonality between machines, with common identity information, common policies, and shared services. Any machine that belongs to a domain functions as a client of the domain, which means it uses the services that the domain provides. IdM domain provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

Hosts in IdM are closely connected with the services running on them:

- Service entries are associated with a host.
- A host stores both the host and the service Kerberos principals.

27.2. HOST ENROLLMENT

This section describes enrolling hosts as IdM clients and what happens during and after the enrollment. The section compares the enrollment of IdM hosts and IdM users. The section also outlines alternative types of authentication available to hosts.

Enrolling a host consists of:

- Creating a host entry in IdM LDAP: possibly using the [ipa host-add command](#) in IdM CLI, or the equivalent [IdM Web UI operation](#).
- Configuring IdM services on the host, for example the System Security Services Daemon (SSSD), Kerberos, and certmonger, and joining the host to the IdM domain.

The two actions can be performed separately or together.

If performed separately, they allow for dividing the two tasks between two users with different levels of privilege. This is useful for bulk deployments.

The **ipa-client-install** command can perform the two actions together. The command creates a host entry in IdM LDAP if that entry does not exist yet, and configures both the Kerberos and SSSD services for the host. The command brings the host within the IdM domain and allows it to identify the IdM server it will connect with. If the host belongs to a DNS zone managed by IdM, **ipa-client-install** adds DNS records for the host too. The command must be run on the client.

27.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running **ipa-client-install**
- If a one-time password (OTP) is used for enrollment

User privileges for optionally manually creating a host entry in IdM LDAP

The user privilege required for creating a host entry in IdM LDAP using the **ipa host-add** CLI command or the IdM Web UI is **Host Administrators**. The **Host Administrators** privilege can be obtained through the **IT Specialist** role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the **ipa-client-install** command. The level of credentials required for executing the **ipa-client-install** command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator's credentials or the **Host Administrators** role. A full administrator is a member of the **admins** group. The **Host Administrators** role provides privileges to add hosts and enroll hosts. For details about this scenario, see [Installing a client using user credentials: interactive installation](#).
- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator's credentials to execute **ipa-client-install** successfully. The limited administrator in this case has the **Enrollment Administrator** role, which provides the **Host Enrollment** privilege. For details, see [Installing a client using user credentials: interactive installation](#).
- The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the **ipa-client-install** command with the **--password** option, supplying the correct OTP. For details, see [Installing a client by using a one-time password: Interactive installation](#).

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

27.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- The enrollment stage ([Table 27.1, “User and host enrollment”](#)):

- An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is **ipa stageuser-add**; for the host, the command is **ipa host-add**.
- A file containing a *key table* or, abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the **ipa-client-install** command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.
- While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

Table 27.1. User and host enrollment

Action	User	Host
Pre-enrollment	\$ ipa stageuser-add <i>user_name</i> [-password]	\$ ipa host-add <i>host_name</i> [--random]
Activating the account	\$ ipa stageuser-activate <i>user_name</i>	\$ ipa-client install [--password] (must be run on the host itself)

- The deployment stage ([Table 27.2, “User and host session authentication”](#)):
 - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
 - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
 - The TGT is then used to obtain specific tickets for specific services.

Table 27.2. User and host session authentication

User	Host
Default means of authentication	Password
Starting a session (ordinary user)	\$ kinit <i>user_name</i> [switch on the host]
The result of successful authentication	TGT to be used to obtain access to specific services
	TGT to be used to obtain access to specific services

TGTs and other Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. The initial granting of a Kerberos ticket, the renewing of the Kerberos credentials, and even the destroying of the Kerberos session are all handled automatically by the IdM services.

27.2.3. Alternative authentication options for IdM hosts

Apart from keytabs, IdM supports two other types of machine authentication:

- SSH keys. The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.
- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the IdM server's certificate authority and then stored in IdM's Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called [certmonger](#).

27.3. HOST ENTRY IN IDM LDAP

This section describes what a host entry in Identity Management (IdM) looks like and what attributes it can contain.

An LDAP host entry contains all relevant information about the client within IdM:

- Service entries associated with the host
- The host and service principal
- Access control rules
- Machine information, such as its physical location and operating system



NOTE

Note that the IdM Web UI **Identity → Hosts** tab does not show all the information about a particular host stored in the IdM LDAP.

27.3.1. Host entry configuration properties

A host entry can contain information about the host that is outside its system configuration, such as its physical location, MAC address, keys, and certificates.

This information can be set when the host entry is created if it is created manually. Alternatively, most of this information can be added to the host entry after the host is enrolled in the domain.

Table 27.3. Host Configuration Properties

UI Field	Command-Line Option	Description
Description	--desc= <i>description</i>	A description of the host.
Locality	--locality= <i>locality</i>	The geographic location of the host.
Location	--location= <i>location</i>	The physical location of the host, such as its data center rack.

UI Field	Command-Line Option	Description
Platform	--platform =string	The host hardware or architecture.
Operating system	--os =string	The operating system and version for the host.
MAC address	--macaddress =address	The MAC address for the host. This is a multi-valued attribute. The MAC address is used by the NIS plug-in to create a NIS ethers map for the host.
SSH public keys	--sshpubkey =string	The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.
Principal name (not editable)	--principalname =principal	The Kerberos principal name for the host. This defaults to the host name during the client installation, unless a different principal is explicitly set in the -p . This can be changed using the command-line tools, but cannot be changed in the UI.
Set One-Time Password	--password =string	This option sets a password for the host which can be used in bulk enrollment.
-	--random	This option generates a random password to be used in bulk enrollment.
-	--certificate =string	A certificate blob for the host.
-	--updatedns	This sets whether the host can dynamically update its DNS entries if its IP address changes.

27.4. ADDING HOST ENTRIES FROM THE WEB UI

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

Figure 27.1. Adding Host Entries

Hosts		
<input type="text"/> Search 		 Refresh  Delete  Add Actions 
<input type="checkbox"/>	Host name	Description
<input type="checkbox"/>	server.example.com	Enrolled True
Showing 1 to 1 of 1 entries.		

3. Enter 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.

The **Class** field has no specific purpose at the moment.

Figure 27.2. Add Host Wizard

Add Host

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

DNS zones can be created in IdM. If the IdM server does not manage the DNS server, the zone can be entered manually in the menu area, like a regular text field.



NOTE

Select the **Force** check box if you want to skip checking whether the host is resolvable via DNS.

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

Figure 27.3. Expanded Entry Page

The screenshot shows the expanded entry page for a host named "server.zone.example.com". The top navigation bar includes tabs for "Settings" (selected), "Host Groups" (highlighted in green), "Netgroups", "Roles", "HBAC Rules", and "Sudo Rules". Below the tabs are buttons for "Refresh", "Revert", "Save", and "Actions". A message above the form states "server.zone.example... is a member of:". The main section is titled "Host Settings" and contains fields for "Host name" (server.zone.example.com), "Principal name" (host/server.zone.example.com@EXAMPLE.COM), "Description" (empty), "Class" (empty), and "Locality" (empty).

CHAPTER 28. MANAGING HOSTS USING ANSIBLE PLAYBOOKS

Ansible is an automation tool used to configure systems, deploy software, and perform rolling updates. Ansible includes support for Identity Management (IdM), and you can use Ansible modules to automate host management.

This chapter describes the following concepts and operations performed when managing hosts and host entries using Ansible playbooks:

- [Hosts in IdM](#)
- [Host enrollment](#)
- [Ensuring the presence of IdM host entries that are only defined by their FQDNs](#)
- [Ensuring the presence of IdM host entries with IP addresses](#)
- [Ensuring the presence of multiple IdM host entries with random passwords](#)
- [Ensuring the presence of an IdM host entry with multiple IP addresses](#)
- [Ensuring the absence of IdM host entries](#)

28.1. HOSTS IN IDM

Identity Management (IdM) manages these identities:

- Users
- Services
- Hosts

A host represents a machine. As an IdM identity, a host has an entry in the IdM LDAP, that is the 389 Directory Server instance of the IdM server.

The host entry in IdM LDAP 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. Any host can be used in **host-based access control** (HBAC) rules.

IdM domain establishes a commonality between machines, with common identity information, common policies, and shared services. Any machine that belongs to a domain functions as a client of the domain, which means it uses the services that the domain provides. IdM domain provides three main services specifically for machines:

- DNS
- Kerberos
- Certificate management

Hosts in IdM are closely connected with the services running on them:

- Service entries are associated with a host.

- A host stores both the host and the service Kerberos principals.

28.2. HOST ENROLLMENT

This section describes enrolling hosts as IdM clients and what happens during and after the enrollment. The section compares the enrollment of IdM hosts and IdM users. The section also outlines alternative types of authentication available to hosts.

Enrolling a host consists of:

- Creating a host entry in IdM LDAP: possibly using the [ipa host-add command](#) in IdM CLI, or the equivalent [IdM Web UI operation](#).
- Configuring IdM services on the host, for example the System Security Services Daemon (SSSD), Kerberos, and certmonger, and joining the host to the IdM domain.

The two actions can be performed separately or together.

If performed separately, they allow for dividing the two tasks between two users with different levels of privilege. This is useful for bulk deployments.

The **ipa-client-install** command can perform the two actions together. The command creates a host entry in IdM LDAP if that entry does not exist yet, and configures both the Kerberos and SSSD services for the host. The command brings the host within the IdM domain and allows it to identify the IdM server it will connect with. If the host belongs to a DNS zone managed by IdM, **ipa-client-install** adds DNS records for the host too. The command must be run on the client.

28.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running **ipa-client-install**
- If a one-time password (OTP) is used for enrollment

User privileges for optionally manually creating a host entry in IdM LDAP

The user privilege required for creating a host entry in IdM LDAP using the **ipa host-add** CLI command or the IdM Web UI is **Host Administrators**. The **Host Administrators** privilege can be obtained through the **IT Specialist** role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the **ipa-client-install** command. The level of credentials required for executing the **ipa-client-install** command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator's credentials or the **Host Administrators** role. A full administrator is a member of the **admins** group. The **Host Administrators** role provides privileges to add hosts and enroll hosts. For details about this scenario, see [Installing a client using user credentials: interactive installation](#).
- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator's credentials to execute **ipa-client-install** successfully. The limited administrator in this case has the **Enrollment Administrator** role, which provides the **Host Enrollment** privilege. For details, see [Installing a client using user credentials: interactive installation](#).

- The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the **ipa-client-install** command with the **--password** option, supplying the correct OTP. For details, see [Installing a client by using a one-time password: Interactive installation](#).

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

28.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- The enrollment stage ([Table 28.1, “User and host enrollment”](#)):
 - An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is **ipa stageuser-add**; for the host, the command is **ipa host-add**.
 - A file containing a *key table* or, abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the **ipa-client-install** command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.
 - While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

Table 28.1. User and host enrollment

Action	User	Host
Pre-enrollment	\$ ipa stageuser-add <i>user_name</i> [- -password]	\$ ipa host-add <i>host_name</i> [-- random]
Activating the account	\$ ipa stageuser-activate <i>user_name</i>	\$ ipa-client install [--password] (must be run on the host itself)

- The deployment stage ([Table 28.2, “User and host session authentication”](#)):
 - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
 - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
 - The TGT is then used to obtain specific tickets for specific services.

Table 28.2. User and host session authentication

User		Host
Default means of authentication	Password	Keytabs
Starting a session (ordinary user)	\$ kinit user_name	[switch on the host]
The result of successful authentication	TGT to be used to obtain access to specific services	TGT to be used to obtain access to specific services

TGTs and other Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. The initial granting of a Kerberos ticket, the renewing of the Kerberos credentials, and even the destroying of the Kerberos session are all handled automatically by the IdM services.

28.2.3. Alternative authentication options for IdM hosts

Apart from keytabs, IdM supports two other types of machine authentication:

- SSH keys. The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.
- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the IdM server's certificate authority and then stored in IdM's Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called [certmonger](#).

28.3. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH FQDN USING ANSIBLE PLAYBOOKS

This section describes ensuring the presence of host entries in Identity Management (IdM) using Ansible playbooks. The host entries are only defined by their **fully-qualified domain names** (FQDNs).

Specifying the **FQDN** name of the host is enough if at least one of the following conditions applies:

- The IdM server is not configured to manage DNS.
- The host does not have a static IP address or the IP address is not known at the time the host is configured. Adding a host defined only by an **FQDN** essentially creates a placeholder entry in the IdM DNS service. For example, laptops may be preconfigured as IdM clients, but they do not have IP addresses at the time they are configured. When the DNS service dynamically updates its records, the host's current IP address is detected and its DNS record is updated.



NOTE

Without Ansible, host entries are created in IdM using the **ipa host-add** command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: **state: present**.

Prerequisites

- You know the IdM administrator password.
- The [ansible-freeipa](#) package is installed on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the **FQDN** of the host whose presence in IdM you want to ensure. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/host/add-host.yml** file:

```
---
- name: Host present
  hosts: ipaserver
  become: true

  tasks:
    - name: Host host01.idm.example.com present
      ipahost:
        ipaadmin_password: MySecret123
        name: host01.idm.example.com
        state: present
        force: yes
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-host-is-present.yml
```



NOTE

The procedure results in a host entry in the IdM LDAP server being created but not in enrolling the host into the IdM Kerberos realm. For that, you must deploy the host as an IdM client. For details, see [Installing an Identity Management client using an Ansible playbook](#).

Verification steps

1. Log in to your IdM server as admin:

```
$ ssh admin@server.idm.example.com
Password:
```

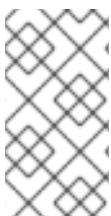
- Enter the **ipa host-show** command and specify the name of the host:

```
$ ipa host-show host01.idm.example.com
Host name: host01.idm.example.com
Principal name: host/host01.idm.example.com@IDM.EXAMPLE.COM
Principal alias: host/host01.idm.example.com@IDM.EXAMPLE.COM
Password: False
Keytab: False
Managed by: host01.idm.example.com
```

The output confirms that **host01.idm.example.com** exists in IdM.

28.4. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH DNS INFORMATION USING ANSIBLE PLAYBOOKS

This section describes ensuring the presence of host entries in Identity Management (IdM) using Ansible playbooks. The host entries are defined by their **fully-qualified domain names** (FQDNs) and their IP addresses.



NOTE

Without Ansible, host entries are created in IdM using the **ipa host-add** command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: **state: present**.

Prerequisites

- You know the IdM administrator password.
- The [ansible-freeipa](#) package is installed on the Ansible controller.

Procedure

- Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

- Create an Ansible playbook file with the **fully-qualified domain name** (FQDN) of the host whose presence in IdM you want to ensure. In addition, if the IdM server is configured to manage DNS and you know the IP address of the host, specify a value for the **ip_address** parameter. The IP address is necessary for the host to exist in the DNS resource records. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/host/host-present.yml** file. You can also include other, additional information:

```
---
- name: Host present
  hosts: ipaserver
```

```

become: true

tasks:
- name: Ensure host01.idm.example.com is present
  ipahost:
    ipaadmin_password: MySecret123
    name: host01.idm.example.com
    description: Example host
    ip_address: 192.168.0.123
    locality: Lab
    ns_host_location: Lab
    ns_os_version: CentOS 7
    ns_hardware_platform: Lenovo T61
    mac_address:
      - "08:00:27:E3:B1:2D"
      - "52:54:00:BD:97:1E"
    state: present

```

- Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-host-is-present.yml
```



NOTE

The procedure results in a host entry in the IdM LDAP server being created but not in enrolling the host into the IdM Kerberos realm. For that, you must deploy the host as an IdM client. For details, see [Installing an Identity Management client using an Ansible playbook](#).

Verification steps

- Log in to your IdM server as admin:

```
$ ssh admin@server.idm.example.com
Password:
```

- Enter the **ipa host-show** command and specify the name of the host:

```
$ ipa host-show host01.idm.example.com
Host name: host01.idm.example.com
Description: Example host
Locality: Lab
Location: Lab
Platform: Lenovo T61
Operating system: CentOS 7
Principal name: host/host01.idm.example.com@IDM.EXAMPLE.COM
Principal alias: host/host01.idm.example.com@IDM.EXAMPLE.COM
MAC address: 08:00:27:E3:B1:2D, 52:54:00:BD:97:1E
Password: False
Keytab: False
Managed by: host01.idm.example.com
```

The output confirms **host01.idm.example.com** exists in IdM.

28.5. ENSURING THE PRESENCE OF MULTIPLE IDM HOST ENTRIES WITH RANDOM PASSWORDS USING ANSIBLE PLAYBOOKS

The **ipahost** module allows the system administrator to ensure the presence or absence of multiple host entries in IdM using just one Ansible task. This section describes how to ensure the presence of multiple host entries that are only defined by their **fully-qualified domain names** (FQDNs). Running the Ansible playbook generates random passwords for the hosts.



NOTE

Without Ansible, host entries are created in IdM using the **ipa host-add** command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: **state: present**.

Prerequisites

- You know the IdM administrator password.
- The [ansible-freeipa](#) package is installed on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the **fully-qualified domain name** (FQDN) of the hosts whose presence in IdM you want to ensure. To make the Ansible playbook generate a random password for each host even when the host already exists in IdM and **update_password** is limited to **on_create**, add the **random: yes** and **force: yes** options. To simplify this step, you can copy and modify the example from the [/usr/share/doc/ansible-freeipa/README-host.md](#) Markdown file:

```
---
- name: Ensure hosts with random password
  hosts: ipaserver
  become: true

  tasks:
    - name: Hosts host01.idm.example.com and host02.idm.example.com present with random
      passwords
        ipahost:
          ipaadmin_password: MySecret123
          hosts:
            - name: host01.idm.example.com
              random: yes
              force: yes
            - name: host02.idm.example.com
              random: yes
              force: yes
        register: ipahost
```

- Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-are-present.yml
[...]
TASK [Hosts host01.idm.example.com and host02.idm.example.com present with random
passwords]
changed: [r8server.idm.example.com] => {"changed": true, "host":
{"host01.idm.example.com": {"randompassword": "0HoIRvjUdH0Ycbf6uYdWTxH"}, "host02.idm.example.com": {"randompassword": "5VdLgrf3wvojmACdHC3uA3s"}}}
```



NOTE

To deploy the hosts as IdM clients using random, one-time passwords (OTPs), see [Authorization options for IdM client enrollment using an Ansible playbook](#) or [Installing a client by using a one-time password: Interactive installation](#).

Verification steps

- Log in to your IdM server as admin:

```
$ ssh admin@server.idm.example.com
Password:
```

- Enter the **ipa host-show** command and specify the name of one of the hosts:

```
$ ipa host-show host01.idm.example.com
Host name: host01.idm.example.com
Password: True
Keytab: False
Managed by: host01.idm.example.com
```

The output confirms **host01.idm.example.com** exists in IdM with a random password.

28.6. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH MULTIPLE IP ADDRESSES USING ANSIBLE PLAYBOOKS

This section describes how to ensure the presence of a host entry in Identity Management (IdM) using Ansible playbooks. The host entry is defined by its **fully-qualified domain name** (FQDN) and its multiple IP addresses.



NOTE

In contrast to the **ipa host** utility, the Ansible **ipahost** module can ensure the presence or absence of several IPv4 and IPv6 addresses for a host. The **ipa host-mod** command cannot handle IP addresses.

Prerequisites

- You know the IdM administrator password.
- The [ansible-freeipa](#) package is installed on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

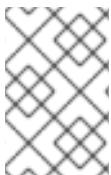
2. Create an Ansible playbook file. Specify, as the **name** of the **ipahost** variable, the **fully-qualified domain name** (FQDN) of the host whose presence in IdM you want to ensure. Specify each of the multiple IPv4 and IPv6 **ip_address** values on a separate line by using the **- ip_address** syntax. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/host/host-member-ipaddresses-present.yml** file. You can also include additional information:

```
---
- name: Host member IP addresses present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure host101.example.com IP addresses present
      ipahost:
        ipaadmin_password: MySecret123
        name: host01.idm.example.com
        ip_address:
          - 192.168.0.123
          - fe80::20c:29ff:fe02:a1b3
          - 192.168.0.124
          - fe80::20c:29ff:fe02:a1b4
        force: yes
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-host-with-multiple-IP-addreses-is-present.yml
```



NOTE

The procedure creates a host entry in the IdM LDAP server but does not enroll the host into the IdM Kerberos realm. For that, you must deploy the host as an IdM client. For details, see [Installing an Identity Management client using an Ansible playbook](#).

Verification steps

1. Log in to your IdM server as admin:

```
$ ssh admin@server.idm.example.com
Password:
```

2. Enter the **ipa host-show** command and specify the name of the host:

```
$ ipa host-show host01.idm.example.com
Principal name: host/host01.idm.example.com@IDM.EXAMPLE.COM
Principal alias: host/host01.idm.example.com@IDM.EXAMPLE.COM
```

```
>Password: False
Keytab: False
Managed by: host01.idm.example.com
```

The output confirms that **host01.idm.example.com** exists in IdM.

3. To verify that the multiple IP addresses of the host exist in the IdM DNS records, enter the **ipa dnsrecord-show** command and specify the following information:
 - The name of the IdM domain
 - The name of the host

```
$ ipa dnsrecord-show idm.example.com host01
[...]
Record name: host01
A record: 192.168.0.123, 192.168.0.124
AAAA record: fe80::20c:29ff:fe02:a1b3, fe80::20c:29ff:fe02:a1b4
```

The output confirms that all the IPv4 and IPv6 addresses specified in the playbook are correctly associated with the **host01.idm.example.com** host entry.

28.7. ENSURING THE ABSENCE OF AN IDM HOST ENTRY USING ANSIBLE PLAYBOOKS

This section describes how to ensure the absence of host entries in Identity Management (IdM) using Ansible playbooks.

Prerequisites

- IdM administrator credentials

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the **fully-qualified domain name** (FQDN) of the host whose absence from IdM you want to ensure. If your IdM domain has integrated DNS, use the **updatedns: yes** option to remove the associated records of any kind for the host from the DNS.

To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/host/delete-host.yml** file:

```
---
- name: Host absent
  hosts: ipaserver
  become: true

  tasks:
    - name: Host host01.idm.example.com absent
      ipahost:
```

```
ipaadmin_password: MySecret123
name: host01.idm.example.com
updatedns: yes
state: absent
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-host-absent.yml
```



NOTE

The procedure results in:

- The host not being present in the IdM Kerberos realm.
- The host entry not being present in the IdM LDAP server.

To remove the specific IdM configuration of system services, such as System Security Services Daemon (SSSD), from the client host itself, you must run the **ipa-client-install --uninstall** command on the client. For details, see [Uninstalling an IdM client](#).

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about *host01.idm.example.com*:

```
$ ipa host-show host01.idm.example.com
ipa: ERROR: host01.idm.example.com: host not found
```

The output confirms that the host does not exist in IdM.

Additional resources

- You can see the definitions of the **ipahost** variables as well as sample Ansible playbooks for ensuring the presence, absence, and disablement of hosts in the **/usr/share/doc/ansible-freeipa/README-host.md** Markdown file.
- Additional playbooks are in the **/usr/share/doc/ansible-freeipa/playbooks/host** directory.

CHAPTER 29. MANAGING HOST GROUPS USING THE IDM CLI

This chapter introduces host groups in Identity Management (IdM) and describes the following operations to manage host groups and their members in the command-line interface (CLI):

- Viewing host groups and their members
- Creating host groups
- Deleting host groups
- Adding host group members
- Removing host group members
- Adding host group member managers
- Removing host group member managers

29.1. HOST GROUPS IN IDM

IdM host groups can be used to centralize control over important management tasks, particularly access control.

Definition of host groups

A host group is an entity that contains a set of IdM hosts with common access control rules and other characteristics. For example, you can define host groups based on company departments, physical locations, or access control requirements.

A host group in IdM can include:

- IdM servers and clients
- Other IdM host groups

Host groups created by default

By default, the IdM server creates the host group **ipaservers** for all IdM server hosts.

Direct and indirect group members

Group attributes in IdM apply to both direct and indirect members: when host group B is a member of host group A, all members of host group B are considered indirect members of host group A.

29.2. VIEWING IDM HOST GROUPS USING THE CLI

This section describes how to view IdM host groups using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Find all host groups using the **ipa hostgroup-find** command.

```
$ ipa hostgroup-find
-----
1 hostgroup matched
-----
Host-group: ipaservers
Description: IPA server hosts
-----
Number of entries returned 1
-----
```

To display all attributes of a host group, add the **--all** option. For example:

```
$ ipa hostgroup-find --all
-----
1 hostgroup matched
-----
dn: cn=ipaservers,cn=hostgroups,cn=accounts,dc=idm,dc=local
Host-group: ipaservers
Description: IPA server hosts
Member hosts: xxx.xxx.xxx.xxx
ipauniqueid: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
objectclass: top, groupOfNames, nestedGroup, ipaobject, ipahostgroup
-----
Number of entries returned 1
-----
```

29.3. CREATING IDM HOST GROUPS USING THE CLI

This section describes how to create IdM host groups using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Add a host group using the **ipa hostgroup-add** command.

For example, to create an IdM host group named *group_name* and give it a description:

```
$ ipa hostgroup-add --desc 'My new host group' group_name
-----
Added hostgroup "group_name"
-----
Host-group: group_name
Description: My new host group
-----
```

29.4. DELETING IDM HOST GROUPS USING THE CLI

This section describes how to delete IdM host groups using the command-line interface (CLI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).

Procedure

1. Delete a host group using the **ipa hostgroup-del** command.
For example, to delete the IdM host group named *group_name*:

```
$ ipa hostgroup-del group_name
-----
Deleted hostgroup "group_name"
```



NOTE

Removing a group does not delete the group members from IdM.

29.5. ADDING IDM HOST GROUP MEMBERS USING THE CLI

You can add hosts as well as host groups as members to an IdM host group using a single command.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- *Optional.* Use the **ipa hostgroup-find** command to find hosts and host groups.

Procedure

1. To add a member to a host group, use the **ipa hostgroup-add-member** and provide the relevant information. You can specify the type of member to add using these options:

- Use the **--hosts** option to add one or more hosts to an IdM host group.
For example, to add the host named *example_member* to the group named *group_name*:

```
$ ipa hostgroup-add-member group_name --hosts example_member
Host-group: group_name
Description: My host group
Member hosts: example_member
-----
Number of members added 1
```

- Use the **--hostgroups** option to add one or more host groups to an IdM host group.
For example, to add the host group named *nested_group* to the group named *group_name*:

```
$ ipa hostgroup-add-member group_name --hostgroups nested_group
```

```

Host-group: group_name
Description: My host group
Member host-groups: nested_group
-----
Number of members added 1
-----
```

- You can add multiple hosts and multiple host groups to an IdM host group in one single command using the following syntax:

```
$ ipa hostgroup-add-member group_name --hosts={host1,host2} --hostgroups={group1,group2}
```



IMPORTANT

When adding a host group as a member of another host group, do not create recursive groups. For example, if Group A is a member of Group B, do not add Group B as a member of Group A. Recursive groups can cause unpredictable behavior.

29.6. REMOVING IDM HOST GROUP MEMBERS USING THE CLI

You can remove hosts as well as host groups from an IdM host group using a single command.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- *Optional*. Use the **ipa hostgroup-find** command to confirm that the group includes the member you want to remove.

Procedure

1. To remove a host group member, use the **ipa hostgroup-remove-member** command and provide the relevant information. You can specify the type of member to remove using these options:

- Use the **--hosts** option to remove one or more hosts from an IdM host group. For example, to remove the host named *example_member* from the group named *group_name*:

```

$ ipa hostgroup-remove-member group_name --hosts example_member
Host-group: group_name
Description: My host group
-----
Number of members removed 1
-----
```

- Use the **--hostgroups** option to remove one or more host groups from an IdM host group. For example, to remove the host group named *nested_group* from the group named *group_name*:

```
$ ipa hostgroup-remove-member group_name --hostgroups example_member
```

```
Host-group: group_name
Description: My host group
```

```
-----
```

```
Number of members removed 1
```



NOTE

Removing a group does not delete the group members from IdM.

- You can remove multiple hosts and multiple host groups from an IdM host group in one single command using the following syntax:

```
$ ipa hostgroup-remove-member group_name --hosts={host1,host2} --hostgroups={group1,group2}
```

29.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE CLI

You can add hosts as well as host groups as member managers to an IdM host group using a single command. Member managers can add hosts or host groups to IdM host groups but cannot change the attributes of a host group.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- You must have the name of the host or host group you are adding as member managers and the name of the host group you want them to manage.

Procedure

1. *Optional.* Use the **ipa hostgroup-find** command to find hosts and host groups.
2. To add a member manager to a host group, use the **ipa hostgroup-add-member-manager**. For example, to add the user named *example_member* as a member manager to the group named *group_name*:

```
$ ipa hostgroup-add-member-manager group_name --user example_member
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Member of netgroups: group_name
Membership managed by users: example_member
```

```
-----
```

```
Number of members added 1
```

3. Use the **--groups** option to add one or more host groups as a member manager to an IdM host group. For example, to add the host group named *admin_group* as a member manager to the group named *group_name*:

```
$ ipa hostgroup-add-member-manager group_name --groups admin_group
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Member of netgroups: group_name
Membership managed by groups: admin_group
Membership managed by users: example_member
-----
Number of members added 1
-----
```



NOTE

After you add a member manager to a host group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Using the **ipa group-show** command to verify the host user and host group were added as member managers.

```
$ ipa hostgroup-show group_name
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Membership managed by groups: admin_group
Membership managed by users: example_member
```

Additional resources

- See **ipa hostgroup-add-member-manager --help** for more details.
- See **ipa hostgroup-show --help** for more details.

29.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE CLI

You can remove hosts as well as host groups as member managers from an IdM host group using a single command. Member managers can remove hosts group member managers from IdM host groups but cannot change the attributes of a host group.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see [Using kinit to log in to IdM manually](#).
- You must have the name of the existing member manager host group you are removing and the name of the host group they are managing.

Procedure

- Optional. Use the **ipa hostgroup-find** command to find hosts and host groups.

- To remove a member manager from a host group, use the **ipa hostgroup-remove-member-manager** command.

For example, to remove the user named *example_member* as a member manager from the group named *group_name*:

```
$ ipa hostgroup-remove-member-manager group_name --user example_member
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Member of netgroups: group_name
Membership managed by groups: nested_group
-----
Number of members removed 1
-----
```

- Use the **--groups** option to remove one or more host groups as a member manager from an IdM host group.

For example, to remove the host group named *nested_group* as a member manager from the group named *group_name*:

```
$ ipa hostgroup-remove-member-manager group_name --groups nested_group
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Member of netgroups: group_name
-----
Number of members removed 1
-----
```



NOTE

After you remove a member manager from a host group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- Use the **ipa group-show** command to verify that the host user and host group were removed as member managers.

```
$ ipa hostgroup-show group_name
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
```

Additional resources

- See **ipa hostgroup-remove-member-manager --help** for more details.
- See **ipa hostgroup-show --help** for more details.

CHAPTER 30. MANAGING HOST GROUPS USING THE IDM WEB UI

This chapter introduces host groups in Identity Management (IdM) and describes the following operations to manage host groups and their members in the Web interface (Web UI):

- Viewing host groups and their members
- Creating host groups
- Deleting host groups
- Adding host group members
- Removing host group members
- Adding host group member managers
- Removing host group member managers

30.1. HOST GROUPS IN IDM

IdM host groups can be used to centralize control over important management tasks, particularly access control.

Definition of host groups

A host group is an entity that contains a set of IdM hosts with common access control rules and other characteristics. For example, you can define host groups based on company departments, physical locations, or access control requirements.

A host group in IdM can include:

- IdM servers and clients
- Other IdM host groups

Host groups created by default

By default, the IdM server creates the host group **ipaservers** for all IdM server hosts.

Direct and indirect group members

Group attributes in IdM apply to both direct and indirect members: when host group B is a member of host group A, all members of host group B are considered indirect members of host group A.

30.2. VIEWING HOST GROUPS IN THE IDM WEB UI

This section describes how to view IdM host groups using the Web interface (Web UI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

- Click **Identity → Groups**, and select the **Host Groups** tab.
 - The page lists the existing host groups and their descriptions.
 - You can search for a specific host group.

The screenshot shows the Red Hat Identity Management interface. The top navigation bar has tabs for Identity, Policy, Authentication, Network Services, and IPA Server. The 'Groups' tab is selected. Below the tabs, there are links for Users, Hosts, Services, and Groups. The 'Groups' link is underlined. On the left, a sidebar shows 'Group categories' with 'User Groups' expanded, showing 'Host Groups' which is also underlined and highlighted in blue. Other options include 'Netgroups'. The main content area is titled 'Host Groups' and contains a table with one entry:

	Host-group	Description
<input type="checkbox"/>	group_name	
<input type="checkbox"/>	ipaservers	IPA server hosts

Below the table, it says 'Showing 1 to 2 of 2 entries.'

- Click on a group in the list to display the hosts that belong to this group. You can limit results to direct or indirect members.

The screenshot shows the 'Host Group: ipaservers' details. The top navigation bar and tabs are the same as the previous screenshot. The 'Host Groups' link in the sidebar is followed by 'ipaservers'. The main content area is titled 'Host Group: ipaservers' and shows the following information:

ipaservers members:

<input type="checkbox"/>	Hosts (1)	Host Groups	Settings
--------------------------	-----------	-------------	----------

ipaservers is a member of:

<input type="checkbox"/>	Host Groups	Netgroups	HBAC Rules	Sudo Rules
--------------------------	-------------	-----------	------------	------------

Buttons at the bottom: Refresh, Delete, Add.

Show Results Direct Membership Indirect Membership

Host name entries: host_minimalhost

Showing 1 to 1 of 1 entries.

- Select the **Host Groups** tab to display the host groups that belong to this group (nested host groups). You can limit results to direct or indirect members.

The screenshot shows the 'Host Group: group_name' details. The top navigation bar and tabs are the same. The 'Host Groups' link in the sidebar is followed by 'group_name'. The main content area is titled 'Host Group: group_name' and shows the following information:

group_name members:

<input type="checkbox"/>	Hosts	Host Groups (1)	Settings
--------------------------	-------	-----------------	----------

group_name is a member of:

<input type="checkbox"/>	Host Groups	Netgroups	HBAC Rules	Sudo Rules
--------------------------	-------------	-----------	------------	------------

Buttons at the bottom: Refresh, Delete, Add.

Show Results Direct Membership Indirect Membership

Host-group entries: host_minimalhost, nested_group

Showing 1 to 1 of 1 entries.

30.3. CREATING HOST GROUPS IN THE IDM WEB UI

This section describes how to create IdM host groups using the Web interface (Web UI).

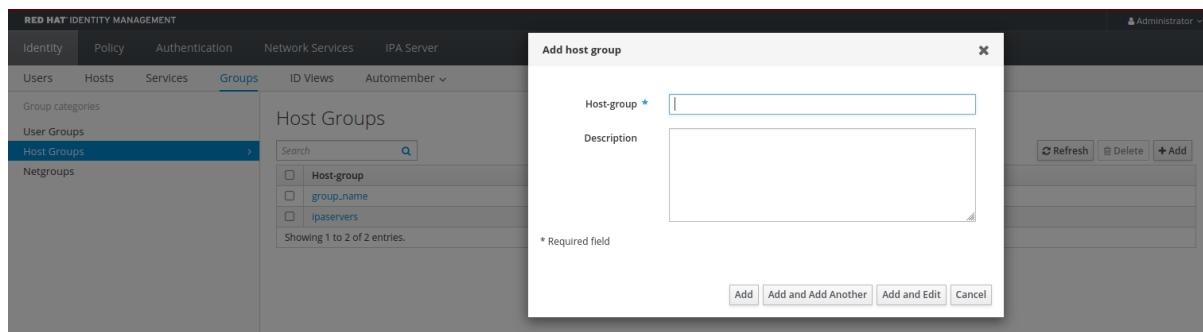
Prerequisites

- Administrator privileges for managing IdM or User Administrator role.

- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Click **Identity → Groups**, and select the **Host Groups** tab.
2. Click **Add**. The **Add host group** dialog appears.
3. Provide the information about the group: name (required) and description (optional).
4. Click **Add** to confirm.



30.4. DELETING HOST GROUPS IN THE IDM WEB UI

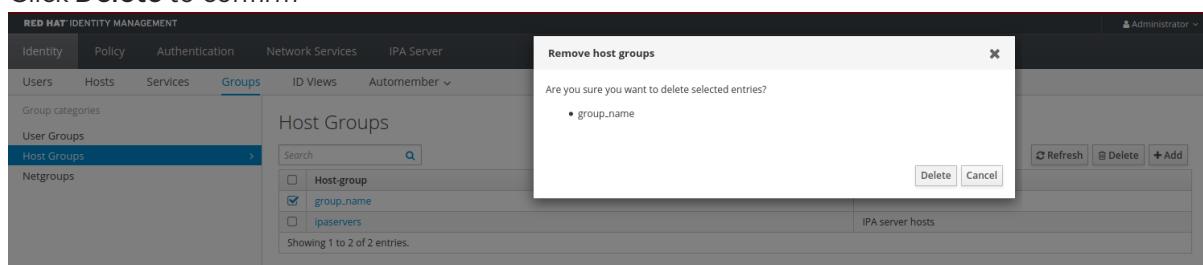
This section describes how to delete IdM host groups using the Web interface (Web UI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Click **Identity → Groups** and select the **Host Groups** tab.
2. Select the IdM host group to remove, and click **Delete**. A confirmation dialog appears.
3. Click **Delete** to confirm



NOTE

Removing a host group does not delete the group members from IdM.

30.5. ADDING HOST GROUP MEMBERS IN THE IDM WEB UI

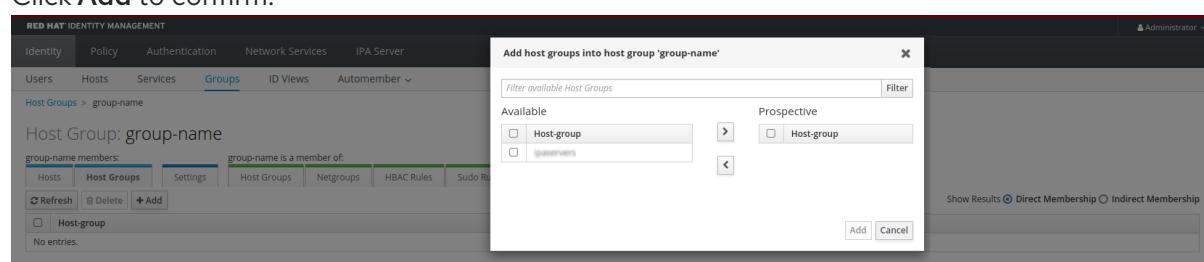
This section describes how to add host group members in IdM using the web interface (Web UI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Click **Identity → Groups** and select the **Host Groups** tab.
2. Click the name of the group to which you want to add members.
3. Click the tab **Hosts** or **Host groups** depending on the type of members you want to add. The corresponding dialog appears.
4. Select the hosts or host groups to add, and click the > arrow button to move them to the **Prospective** column.
5. Click **Add** to confirm.



30.6. REMOVING HOST GROUP MEMBERS IN THE IDM WEB UI

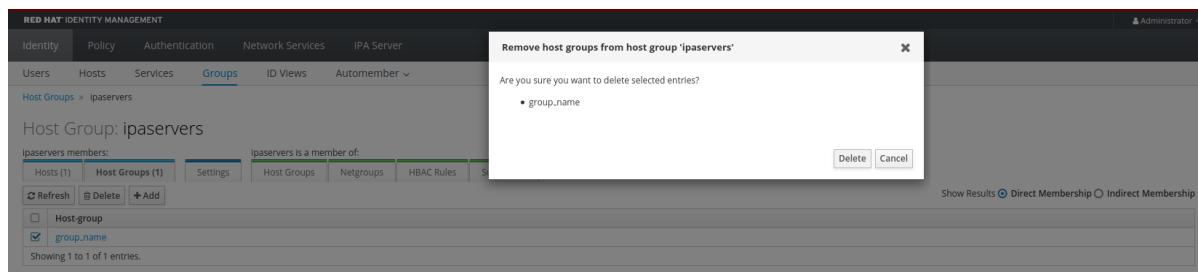
This section describes how to remove host group members in IdM using the web interface (Web UI).

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).

Procedure

1. Click **Identity → Groups** and select the **Host Groups** tab.
2. Click the name of the group from which you want to remove members.
3. Click the tab **Hosts** or **Host groups** depending on the type of members you want to remove.
4. Select the check box next to the member you want to remove.
5. Click **Delete**. A confirmation dialog appears.



- Click Delete to confirm. The selected members are deleted.

30.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI

This section describes how to add users or user groups as host group member managers in IdM using the web interface (Web UI). Member managers can add hosts group member managers to IdM host groups but cannot change the attributes of a host group.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).
- You must have the name of the host group you are adding as member managers and the name of the host group you want them to manage.

Procedure

- Click **Identity → Groups** and select the **Host Groups** tab.

	Host-group	Description
<input type="checkbox"/>	ipaservers	IPA server hosts

- Click the name of the group to which you want to add member managers.
- Click the member managers tab **User Groups** or **Users** depending on the type of member managers you want to add. The corresponding dialog appears.
- Click **Add**.

Add groups as member managers for host group 'ipaservers'

Filter available User Groups Filter

Available		Prospective	
<input type="checkbox"/>	Group name	<input type="checkbox"/>	Group name
<input type="checkbox"/>	editors	<input type="checkbox"/>	
<input type="checkbox"/>	ipausers	<input type="checkbox"/>	
<input type="checkbox"/>	trust admins	<input type="checkbox"/>	

> <

Add Cancel

5. Select the users or user groups to add, and click the > arrow button to move them to the **Prospective** column.
6. Click **Add** to confirm.



NOTE

After you add a member manager to a host group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- On the Host Group dialog, verify the user group or user has been added to the member managers list of groups or users.

RED HAT IDENTITY MANAGEMENT Administrator

Identity	Policy	Authentication	Network Services	IPA Server	
Users	Hosts	Services	Groups	ID Views	Automember ▾

Host Groups » ipaservers

Host Group: ipaservers

ipaservers members:

Hosts (1)	Host Groups	Settings
-----------	-------------	----------

ipaservers is a member of:

Host Groups	Netgroups	HBAC Rules	Sudo Rules
-------------	-----------	------------	------------

ipaservers member managers:

User Groups (1)	Users
-----------------	-------

Refresh Delete + Add

Group name
 admins

Showing 1 to 1 of 1 entries.

30.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI

This section describes how to remove users or user groups as host group member managers in IdM using the web interface (Web UI). Member managers can remove hosts group member managers from IdM host groups but cannot change the attributes of a host group.

Prerequisites

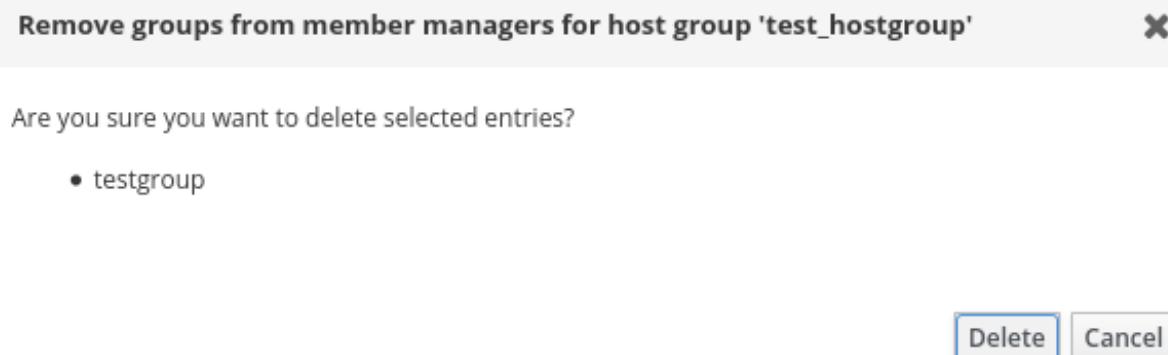
- Administrator privileges for managing IdM or User Administrator role.
- You are logged-in to the IdM Web UI. For details, see [Accessing the IdM Web UI in a web browser](#).
- You must have the name of the existing member manager host group you are removing and the name of the host group they are managing.

Procedure

- Click **Identity → Groups** and select the **Host Groups** tab.

	Host-group	Description
<input type="checkbox"/>	ipaservers	IPA server hosts
<input type="checkbox"/>	test_hostgroup	

- Click the name of the group from which you want to remove member managers.
- Click the member managers tab **User Groups** or **Users** depending on the type of member managers you want to remove. The corresponding dialog appears.
- Select the user or user groups to remove and click **Delete**.
- Click **Delete** to confirm.



NOTE

After you remove a member manager from a host group, the update may take some time to spread to all clients in your Identity Management environment.

Verification steps

- On the Host Group dialog, verify the user group or user has been removed from the member managers list of groups or users.

RED HAT® IDENTITY MANAGEMENT

Administrator

Identity Policy Authentication Network Services IPA Server

Users Hosts Services Groups ID Views Automember

Host Groups » test_hostgroup

Host Group: test_hostgroup

test_hostgroup members:

Hosts	Host Groups	Settings
Host Groups	Netgroups	HBAC Rules
Sudo Rules	User Groups	Users (1)

test_hostgroup is a member of:

Host Groups	Netgroups	HBAC Rules
Sudo Rules	User Groups	Users (1)

test_hostgroup member managers:

Host Groups	Netgroups	HBAC Rules
Sudo Rules	User Groups	Users (1)

Refresh Delete + Add

Group name

No entries.

CHAPTER 31. MANAGING HOST GROUPS USING ANSIBLE PLAYBOOKS

This chapter introduces [host groups in Identity Management](#) (IdM) and describes using Ansible to perform the following operations involving host groups in Identity Management (IdM):

- Host groups in IdM
- Ensuring the presence of IdM host groups
- Ensuring the presence of hosts in IdM host groups
- Nesting IdM host groups
- Ensuring the presence of member managers in IdM host groups
- Ensuring the absence of hosts from IdM host groups
- Ensuring the absence of nested host groups from IdM host groups
- Ensuring the absence of member managers from IdM host groups

31.1. HOST GROUPS IN IDM

IdM host groups can be used to centralize control over important management tasks, particularly access control.

Definition of host groups

A host group is an entity that contains a set of IdM hosts with common access control rules and other characteristics. For example, you can define host groups based on company departments, physical locations, or access control requirements.

A host group in IdM can include:

- IdM servers and clients
- Other IdM host groups

Host groups created by default

By default, the IdM server creates the host group **ipaservers** for all IdM server hosts.

Direct and indirect group members

Group attributes in IdM apply to both direct and indirect members: when host group B is a member of host group A, all members of host group B are considered indirect members of host group A.

31.2. ENSURING THE PRESENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the presence of host groups in Identity Management (IdM) using Ansible playbooks.



NOTE

Without Ansible, host group entries are created in IdM using the **ipa hostgroup-add** command. The result of adding a host group to IdM is the state of the host group being present in IdM. Because of the Ansible reliance on idempotence, to add a host group to IdM using Ansible, you must create a playbook in which you define the state of the host group as present: **state: present**.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host group information. For example, to ensure the presence of a host group named **databases**, specify **name: databases** in the **- ipahostgroup** task. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/user/ensure-hostgroup-is-present.yml** file.

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    # Ensure host-group databases is present
    - ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        state: present
```

In the playbook, **state: present** signifies a request to add the host group to IdM unless it already exists there.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hostgroup-is-present.yml
```

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

3. Display information about the host group whose presence in IdM you wanted to ensure:

```
$ ipa hostgroup-show databases
Host-group: databases
```

The **databases** host group exists in IdM.

31.3. ENSURING THE PRESENCE OF HOSTS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the presence of hosts in host groups in Identity Management (IdM) using Ansible playbooks.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- The hosts you want to reference in your Ansible playbook exist in IdM. For details, see [Ensuring the presence of an IdM host entry using Ansible playbooks](#).
- The host groups you reference from the Ansible playbook file have been added to IdM. For details, see [Ensuring the presence of IdM host groups using Ansible playbooks](#).

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host information. Specify the name of the host group using the **name** parameter of the **ipahostgroup** variable. Specify the name of the host with the **host** parameter of the **ipahostgroup** variable. To simplify this step, you can copy and modify the examples in the **/usr/share/doc/ansible-freeipa/playbooks/hostgroup/ensure-hosts-and-hostgroups-are-present-in-hostgroup.yml** file:

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    # Ensure host-group databases is present
    - ipahostgroup:
        ipaadmin_password: MySecret123
```

```

name: databases
host:
- db.idm.example.com
action: member

```

This playbook adds the `db.idm.example.com` host to the `databases` host group. The `action: member` line indicates that when the playbook is run, no attempt is made to add the `databases` group itself. Instead, only an attempt is made to add `db.idm.example.com` to `databases`.

- Run the playbook:

```

$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-or-hostgroups-are-present-in-
hostgroup.yml

```

Verification steps

- Log into `ipaserver` as admin:

```

$ ssh admin@server.idm.example.com
Password:
[admin@server /]$

```

- Request a Kerberos ticket for admin:

```

$ kinit admin
Password for admin@IDM.EXAMPLE.COM:

```

- Display information about a host group to see which hosts are present in it:

```

$ ipa hostgroup-show databases
Host-group: databases
Member hosts: db.idm.example.com

```

The `db.idm.example.com` host is present as a member of the `databases` host group.

31.4. NESTING IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes ensuring the presence of nested host groups in Identity Management (IdM) host groups using Ansible playbooks.

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.
- The host groups you reference from the Ansible playbook file exist in IdM. For details, see [Ensuring the presence of IdM host groups using Ansible playbooks](#) .

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host group information. To ensure that a nested host group *A* exists in a host group *B*: in the Ansible playbook, specify, among the **- ipahostgroup** variables, the name of the host group *B* using the **name** variable. Specify the name of the nested hostgroup *A* with the **hostgroup** variable. To simplify this step, you can copy and modify the examples in the **/usr/share/doc/ansible-freeipa/playbooks/hostgroup/ensure-hosts-and-hostgroups-are-present-in-hostgroup.yml** file:

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    # Ensure hosts and hostgroups are present in existing databases hostgroup
    - ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        hostgroup:
          - mysql-server
          - oracle-server
        action: member
```

This Ansible playbook ensures the presence of the **mysql-server** and **oracle-server** host groups in the **databases** host group. The **action: member** line indicates that when the playbook is run, no attempt is made to add the **databases** group itself to IdM.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-or-hostgroups-are-present-in-
hostgroup.yml
```

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

3. Display information about the host group in which nested host groups are present:

```
$ ipa hostgroup-show databases
Host-group: databases
Member hosts: db.idm.example.com
Member host-groups: mysql-server, oracle-server
```

The **mysql-server** and **oracle-server** host groups exist in the **databases** host group.

31.5. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the presence of member managers in IdM hosts and host groups using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- You must have the name of the host or host group you are adding as member managers and the name of the host group you want them to manage.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host and host group member management information:

```
---
- name: Playbook to handle host group membership management
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure member manager user example_member is present for group_name
      ipahostgroup:
        ipaadmin_password: MySecret123
        name: group_name
        membermanager_user: example_member

    - name: Ensure member manager group project_admins is present for group_name
      ipahostgroup:
        ipaadmin_password: MySecret123
        name: group_name
        membermanager_group: project_admins
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/add-member-managers-host-groups.yml
```

Verification steps

You can verify if the `group_name` group contains `example_member` and `project_admins` as member managers by using the **ipa group-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about `testhostgroup`:

```
[ipaserver]$ ipa hostgroup-show group_name
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: testhostgroup2
Membership managed by groups: project_admins
Membership managed by users: example_member
```

Additional resources

- See **ipa hostgroup-add-member-manager --help**.
- See the **ipa** man page.

31.6. ENSURING THE ABSENCE OF HOSTS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the absence of hosts from host groups in Identity Management (IdM) using Ansible playbooks.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- The hosts you want to reference in your Ansible playbook exist in IdM. For details, see [Ensuring the presence of an IdM host entry using Ansible playbooks](#).
- The host groups you reference from the Ansible playbook file exist in IdM. For details, see [Ensuring the presence of IdM host groups using Ansible playbooks](#).

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host and host group information. Specify the name of the host group using the **name** parameter of the **ipahostgroup** variable. Specify the name of the host whose absence from the host group you want to ensure using the **host** parameter of the **ipahostgroup** variable. To simplify this step, you can copy and modify the examples in the **/usr/share/doc/ansible-freeipa/playbooks/hostgroup/ensure-hosts-and-hostgroups-are-absent-in-hostgroup.yml** file:

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    # Ensure host-group databases is absent
    - ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        host:
          - db.idm.example.com
        action: member
        state: absent
```

This playbook ensures the absence of the **db.idm.example.com** host from the **databases** host group. The **action: member** line indicates that when the playbook is run, no attempt is made to remove the **databases** group itself.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-or-hostgroups-are-absent-in-
hostgroup.yml
```

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

3. Display information about the host group and the hosts it contains:

```
$ ipa hostgroup-show databases
Host-group: databases
Member host-groups: mysql-server, oracle-server
```

The **db.idm.example.com** host does not exist in the **databases** host group.

31.7. ENSURING THE ABSENCE OF NESTED HOST GROUPS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the absence of nested host groups from outer host groups in Identity Management (IdM) using Ansible playbooks.

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- The host groups you reference from the Ansible playbook file exist in IdM. For details, see [Ensuring the presence of IdM host groups using Ansible playbooks](#).

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host group information. Specify, among the **- ipahostgroup** variables, the name of the outer host group using the **name** variable. Specify the name of the nested hostgroup with the **hostgroup** variable. To simplify this step, you can copy and modify the examples in the **/usr/share/doc/ansible-freeipa/playbooks/hostgroup/ensure-hosts-and-hostgroups-are-absent-in-hostgroup.yml** file:

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    # Ensure hosts and hostgroups are absent in existing databases hostgroup
    - ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        hostgroup:
          - mysql-server
          - oracle-server
        action: member
        state: absent
```

This playbook makes sure that the **mysql-server** and **oracle-server** host groups are absent from the **databases** host group. The **action: member** line indicates that when the playbook is run, no attempt is made to ensure the **databases** group itself is deleted from IdM.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-or-hostgroups-are-absent-in-
hostgroup.yml
```

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

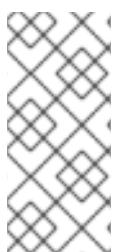
3. Display information about the host group from which nested host groups should be absent:

```
$ ipa hostgroup-show databases
Host-group: databases
```

The output confirms that the **mysql-server** and **oracle-server** nested host groups are absent from the outer **databases** host group.

31.8. ENSURING THE ABSENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the absence of host groups in Identity Management (IdM) using Ansible playbooks.



NOTE

Without Ansible, host group entries are removed from IdM using the **ipa hostgroup-del** command. The result of removing a host group from IdM is the state of the host group being absent from IdM. Because of the Ansible reliance on idempotence, to remove a host group from IdM using Ansible, you must create a playbook in which you define the state of the host group as absent: **state: absent**

Prerequisites

- You know the IdM administrator password.
- You have installed the [ansible-freeipa](#) package on the Ansible controller.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it with the list of IdM servers to target:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host group information. To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/user/ensure-hostgroup-is-absent.yml` file.

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true

  tasks:
    - Ensure host-group databases is absent
      ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        state: absent
```

This playbook ensures the absence of the **databases** host group from IdM. The **state: absent** means a request to delete the host group from IdM unless it is already deleted.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hostgroup-is-absent.yml
```

Verification steps

1. Log into **ipaserver** as admin:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Request a Kerberos ticket for admin:

```
$ kinit admin
Password for admin@IDM.EXAMPLE.COM:
```

3. Display information about the host group whose absence you ensured:

```
$ ipa hostgroup-show databases
ipa: ERROR: databases: host group not found
```

The **databases** host group does not exist in IdM.

31.9. ENSURING THE ABSENCE OF MEMBER MANAGERS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

The following procedure describes ensuring the absence of member managers in IdM hosts and host groups using an Ansible playbook.

Prerequisites

- You know the IdM administrator password.

- You have installed the [ansible-freeipa](#) package on the Ansible controller.
- You must have the name of the user or user group you are removing as member managers and the name of the host group they are managing.

Procedure

1. Create an inventory file, for example **inventory.file**, and define **ipaserver** in it:

```
[ipaserver]
server.idm.example.com
```

2. Create an Ansible playbook file with the necessary host and host group member management information:

```
---
- name: Playbook to handle host group membership management
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure member manager host and host group members are absent for
      group_name
        ipahostgroup:
          ipaadmin_password: MySecret123
          name: group_name
          membermanager_user: example_member
          membermanager_group: project_admins
          action: member
          state: absent
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-member-managers-host-groups-are-absent.yml
```

Verification steps

You can verify if the **group_name** group does not contain **example_member** or **project_admins** as member managers by using the **ipa group-show** command:

1. Log into **ipaserver** as administrator:

```
$ ssh admin@server.idm.example.com
Password:
[admin@server /]$
```

2. Display information about *testhostgroup*:

```
ipaserver]$ ipa hostgroup-show group_name
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: testhostgroup2
```

Additional resources

- See **ipa hostgroup-add-member-manager --help**.
- See the **ipa** man page.

CHAPTER 32. ENSURING THE PRESENCE OF HOST-BASED ACCESS CONTROL RULES IN IDM USING ANSIBLE PLAYBOOKS

This chapter describes Identity Management (IdM) host-based access policies and how to define them using [Ansible](#).

Ansible is an automation tool used to configure systems, deploy software, and perform rolling updates. It includes support for Identity Management (IdM).

32.1. HOST-BASED ACCESS CONTROL RULES IN IDM

Host-based access control (HBAC) rules define which users or user groups can access which hosts or host groups by using which services or services in a service group. As a system administrator, you can use HBAC rules to achieve the following goals:

- Limit access to a specified system in your domain to members of a specific user group.
- Allow only a specific service to be used to access systems in your domain.

By default, IdM is configured with a default HBAC rule named `allow_all`, which means universal access to every host for every user via every relevant service in the entire IdM domain.

You can fine-tune access to different hosts by replacing the default `allow_all` rule with your own set of HBAC rules. For centralized and simplified access control management, you can apply HBAC rules to user groups, host groups, or service groups instead of individual users, hosts, or services.

32.2. ENSURING THE PRESENCE OF AN HBAC RULE IN IDM USING AN ANSIBLE PLAYBOOK

This section describes how to ensure the presence of a host-based access control (HBAC) rule in Identity Management (IdM) using an Ansible playbook.

Prerequisites

- The [ansible-freeipa](#) package is installed on the Ansible controller.
- You know the IdM administrator password.
- The users and user groups you want to use for your HBAC rule exist in IdM. See [Managing user accounts using Ansible playbooks](#) and [Ensuring the presence of IdM groups and group members using Ansible playbooks](#) for details.
- The hosts and host groups to which you want to apply your HBAC rule exist in IdM. See [Managing hosts using Ansible playbooks](#) and [Managing host groups using Ansible playbooks](#) for details.

Procedure

1. Create an inventory file, for example `inventory.file`, and define `ipaserver` in it:

```
[ipaserver]
server.idm.example.com
```

2. Create your Ansible playbook file that defines the HBAC policy whose presence you want to ensure. To simplify this step, you can copy and modify the example in the **/usr/share/doc/ansible-freeipa/playbooks/hbacrule/ensure-hbacrule-allhosts-present.yml** file:

```
---
- name: Playbook to handle hbacrules
  hosts: ipaserver
  become: true

  tasks:
    # Ensure idm_user can access client.idm.example.com via the sshd service
    - ipahbacrule:
        ipaadmin_password: MySecret123
        name: login
        user: idm_user
        host: client.idm.example.com
        hbacsrv:
          - sshd
        state: present
```

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-new-hbacrule-present.yml
```

Verification steps

1. Log in to the IdM Web UI as administrator.
2. Navigate to **Policy → Host-Based-Access-Control → HBAC Test**
3. In the **Who** tab, select **idm_user**.
4. In the **Accessing** tab, select **client.idm.example.com**.
5. In the **Via service** tab, select **sshd**.
6. In the **Rules** tab, select **login**.
7. In the **Run test** tab, click the **Run test** button. If you see ACCESS GRANTED, the HBAC rule is implemented successfully.

Additional resources

- For more details about and examples of, configuring HBAC services, service groups, and rules using Ansible, see the README-hbacsvc.md, README-hbacsvgroup.md, and README-hbacrule.md Markdown files. These files are available in the **/usr/share/doc/ansible-freeipa** directory. Also see the playbooks available in the relevant subdirectories of the **/usr/share/doc/ansible-freeipa/playbooks** directory.

CHAPTER 33. CONFIGURING THE DOMAIN RESOLUTION ORDER TO RESOLVE SHORT AD USER NAMES

By default, you must specify fully qualified names in the format **user_name@domain.com** or **domain.com\user_name** to resolve and authenticate users and groups from an Active Directory (AD) environment. The following sections describe how to configure IdM servers and clients to resolve short AD usernames and group names.

- How domain resolution order works
- Setting the global domain resolution order on an IdM server
- Setting the domain resolution order for an ID view on an IdM server
- Setting the domain resolution order in SSSD on an IdM client

33.1. HOW DOMAIN RESOLUTION ORDER WORKS

In Identity Management (IdM) environments with an Active Directory (AD) trust, Red Hat recommends that you resolve and authenticate users and groups by specifying their fully qualified names. For example:

- **<idm_username>@idm.example.com** for IdM users from the **idm.example.com** domain
- **<ad_username>@ad.example.com** for AD users from the **ad.example.com** domain

By default, if you perform user or group lookups using the *short name* format, such as **ad_username**, IdM only searches the IdM domain and fails to find the AD users or groups. To resolve AD users or groups using short names, change the order in which IdM searches multiple domains by setting the **domain resolution order** option.

You can set the domain resolution order centrally in the IdM database or in the SSSD configuration of individual clients. IdM evaluates domain resolution order in the following order of priority:

- The local **/etc/sssd/sssd.conf** configuration.
- The ID view configuration.
- The global IdM configuration.

Notes

- You must use fully qualified usernames if the SSSD configuration on the host includes the **default_domain_suffix** option and you want to make a request to a domain not specified with this option.
- If you use the **domain resolution order** option and query the **compat** tree, you might receive multiple user IDs (UIDs). If this might affect you, see Pagure bug report [Inconsistent compat user objects for AD users when domain resolution order is set](#).



IMPORTANT

Do not use the **full_name_format** SSSD option on IdM clients or IdM servers. Using a non-default value for this option changes how usernames are displayed and might disrupt lookups in an IdM environment.

Additional resources

- For more information on the **compat** tree, see [Active Directory Trust for Legacy Linux Clients](#).

33.2. SETTING THE GLOBAL DOMAIN RESOLUTION ORDER ON AN IDM SERVER

This procedure sets the domain resolution order for all the clients in the IdM domain. This example sets the domain resolution order to search for users and groups in the following order:

1. Active Directory (AD) root domain **ad.example.com**
2. AD child domain **subdomain1.ad.example.com**
3. IdM domain **idm.example.com**

Prerequisites

- You have configured a trust with an AD environment.

Procedure

- Use the **ipa config-mod --domain-resolution-order** command to list the domains to be searched in your preferred order. Separate the domains with a colon (:).

```
[user@server ~]$ ipa config-mod --domain-resolution-
order='ad.example.com:subdomain1.ad.example.com:idm.example.com'
Maximum username length: 32
Home directory base: /home
...
Domain Resolution Order:
ad.example.com:subdomain1.ad.example.com:idm.example.com
...
```

Verification steps

- Verify you can retrieve user information for a user from the **ad.example.com** domain using only a short name.

```
[root@client ~]# id <ad_username>
uid=1916901102(ad_username) gid=1916900513(domain users)
groups=1916900513(domain users)
```

33.3. SETTING THE DOMAIN RESOLUTION ORDER FOR AN ID VIEW ON AN IDM SERVER

This procedure sets the domain resolution order for an ID view that you can apply to a specific set of IdM servers and clients. This example creates an ID view named **ADsubdomain1_first** for IdM host **client1.idm.example.com**, and sets the domain resolution order to search for users and groups in the following order:

1. Active Directory (AD) child domain **subdomain1.ad.example.com**

2. AD root domain **ad.example.com**
3. IdM domain **idm.example.com**



NOTE

The domain resolution order set in an ID view overrides the global domain resolution order, but it does not override any domain resolution order set locally in the SSSD configuration.

Prerequisites

- You have configured a trust with an AD environment.

Procedure

1. Create an ID view with the **--domain-resolution-order** option set.

```
[user@server ~]$ ipa idview-add ADsubdomain1_first --desc "ID view for resolving AD
subdomain1 first on client1.idm.example.com" --domain-resolution-order
subdomain1.ad.example.com:ad.example.com:idm.example.com
-----
Added ID View "ADsubdomain1_first"
-----
ID View Name: ADsubdomain1_first
Description: ID view for resolving AD subdomain1 first on client1.idm.example.com
Domain Resolution Order:
subdomain1.ad.example.com:ad.example.com:idm.example.com
```

2. Apply the ID view to IdM hosts.

```
[user@server ~]$ ipa idview-apply ADsubdomain1_first --hosts
client1.idm.example.com
-----
Applied ID View "ADsubdomain1_first"
-----
hosts: client1.idm.example.com
-----
Number of hosts the ID View was applied to: 1
```

Verification steps

- Display the details of the ID view.

```
[user@server ~]$ ipa idview-show ADsubdomain1_first --show-hosts
ID View Name: ADsubdomain1_first
Description: ID view for resolving AD subdomain1 first on client1.idm.example.com
Hosts the view applies to: client1.idm.example.com
Domain resolution order:
subdomain1.ad.example.com:ad.example.com:idm.example.com
```

- Verify you can retrieve user information for a user from the **subdomain1.ad.example.com** domain using only a short name.

```
[root@client1 ~]# id <user_from_subdomain1>
uid=1916901106(user_from_subdomain1) gid=1916900513(domain users)
groups=1916900513(domain users)
```

Additional resources

- Using an ID view to override a user attribute value on an IdM client

33.4. SETTING THE DOMAIN RESOLUTION ORDER IN SSSD ON AN IDM CLIENT

This procedure sets the domain resolution order in the SSSD configuration on an IdM client. This example configures IdM host **client2.idm.example.com** to search for users and groups in the following order:

1. Active Directory (AD) child domain **subdomain1.ad.example.com**
2. AD root domain **ad.example.com**
3. IdM domain **idm.example.com**



NOTE

The domain resolution order in the local SSSD configuration overrides any global and ID view domain resolution order.

Prerequisites

- You have configured a trust with an AD environment.

Procedure

1. Open the **/etc/sssd/sssd.conf** file in a text editor.
2. Set the **domain_resolution_order** option in the **[sssd]** section of the file.


```
domain_resolution_order = subdomain1.ad.example.com, ad.example.com,
idm.example.com
```
3. Save and close the file.
4. Restart the SSSD service to load the new configuration settings.

```
[root@client2 ~]# systemctl restart sssd
```

Verification Steps

- Verify you can retrieve user information for a user from the **subdomain1.ad.example.com** domain using only a short name.

```
[root@client2 ~]# id <user_from_subdomain1>
uid=1916901106(user_from_subdomain1) gid=1916900513(domain users)
groups=1916900513(domain users)
```


CHAPTER 34. ENABLING AUTHENTICATION USING AD USER PRINCIPAL NAMES IN IDM

34.1. USER PRINCIPAL NAMES IN AN AD FOREST TRUSTED BY IDM

As an Identity Management (IdM) administrator, you can allow AD users to use alternative **User Principal Names** (UPNs) to access resources in the IdM domain. A UPN is an alternative user login that AD users authenticate with in the format of **user_name@KERBEROS-REALM**. As an AD administrator, you can set alternative values for both **user_name** and **KERBEROS-REALM**, since you can configure both additional Kerberos aliases and UPN suffixes in an AD forest.

For example, if a company uses the Kerberos realm **AD.EXAMPLE.COM**, the default UPN for a user is **user@ad.example.com**. To allow your users to log in using their email addresses, for example **user@example.com**, you can configure **EXAMPLE.COM** as an alternative UPN in AD. Alternative UPNs (also known as *enterprise UPNs*) are especially convenient if your company has recently experienced a merge and you want to provide your users with a unified logon namespace.

UPN suffixes are only visible for IdM when defined in the AD forest root. As an AD administrator, you can define UPNs with the **Active Directory Domain and Trust** utility or the **PowerShell** command line tool.



NOTE

To configure UPN suffixes for users, Red Hat recommends to use tools that perform error validation, such as the **Active Directory Domain and Trust** utility.

Red Hat recommends against configuring UPNs through low-level modifications, such as using **ldapmodify** commands to set the **userPrincipalName** attribute for users, because Active Directory does not validate those operations.

After you define a new UPN on the AD side, run the **ipa trust-fetch-domains** command on an IdM server to retrieve the updated UPNs. See [Ensuring that AD UPNs are up-to-date in IdM](#).

IdM stores the UPN suffixes for a domain in the multi-value attribute **ipaNTAdditionalSuffixes** of the subtree **cn=trusted_domain_name,cn=ad,cn=trusts,dc=idm,dc=example,dc=com**.

Additional resources

- [How to script UPN suffix setup in AD forest root](#)
- [How to manually modify AD user entries and bypass any UPN suffix validation](#)
- [Trust controllers and trust agents](#)

34.2. ENSURING THAT AD UPNS ARE UP-TO-DATE IN IDM

After you add or remove a User Principal Name (UPN) suffix in a trusted Active Directory (AD) forest, refresh the information for the trusted forest on an IdM server.

Prerequisites

- IdM administrator credentials.

Procedure

- Enter the **ipa trust-fetch-domains** command. Note that a seemingly empty output is expected:

```
[root@ipaserver ~]# ipa trust-fetch-domains
Realm-Name: ad.example.com
-----
No new trust domains were found
-----
Number of entries returned 0
```

Verification steps

- Enter the **ipa trust-show** command to verify that the server has fetched the new UPN. Specify the name of the AD realm when prompted:

```
[root@ipaserver ~]# ipa trust-show
Realm-Name: ad.example.com
Realm-Name: ad.example.com
Domain NetBIOS name: AD
Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
Trust direction: One-way trust
Trust type: Active Directory domain
UPN suffixes: example.com
```

The output shows that the **example.com** UPN suffix is now part of the **ad.example.com** realm entry.

34.3. GATHERING TROUBLESHOOTING DATA FOR AD UPN AUTHENTICATION ISSUES

This procedure describes how to gather troubleshooting data about the User Principal Name (UPN) configuration from your Active Directory (AD) environment and your IdM environment. If your AD users are unable to log in using alternate UPNs, you can use this information to narrow your troubleshooting efforts.

Prerequisites

- You must be logged in to an IdM Trust Controller or Trust Agent to retrieve information from an AD domain controller.
- You need **root** permissions to modify the following configuration files, and to restart IdM services.

Procedure

- Open the **/usr/share/ipa/smb.conf.empty** configuration file in a text editor.
- Add the following contents to the file.

```
[global]
log level = 10
```

- Save and close the **/usr/share/ipa/smb.conf.empty** file.

4. Open the **/etc/ipa/server.conf** configuration file in a text editor. If you do not have that file, create one.
5. Add the following contents to the file.

```
[global]
debug = True
```

6. Save and close the **/etc/ipa/server.conf** file.
7. Restart the Apache webserver service to apply the configuration changes:

```
[root@server ~]# systemctl restart httpd
```

8. Retrieve trust information from your AD domain:

```
[root@server ~]# ipa trust-fetch-domains <ad.example.com>
```

9. Review the debugging output and troubleshooting information in the following log files:
 - **/var/log/httpd/error_log**
 - **/var/log/samba/log.***

Additional resources

- For additional troubleshooting steps, see the Knowledgebase article [Using rpcclient to gather troubleshooting data for AD UPN authentication issues](#).

CHAPTER 35. ENABLING AD USERS TO ADMINISTER IDM

35.1. ID OVERRIDES FOR AD USERS

In Red Hat Enterprise Linux (RHEL) 7, external group membership allows Active Directory (AD) users and groups to access Identity Management (IdM) resources in a POSIX environment with the help of the System Security Services Daemon (SSSD).

The IdM LDAP server has its own mechanisms to grant access control. RHEL 8 introduces an update that allows adding an ID user override for an AD user as a member of an IdM group. An ID override is a record describing what a specific Active Directory user or group properties should look like within a specific ID view, in this case the **Default Trust View**. As a consequence of the update, the IdM LDAP server is able to apply access control rules for the IdM group to the AD user.

AD users are now able to use the self service features of IdM UI, for example to upload their SSH keys, or change their personal data. An AD administrator is able to fully administer IdM without having two different accounts and passwords.



NOTE

Currently, selected features in IdM may still be unavailable to AD users. For example, setting passwords for IdM users as an AD user from the IdM **admins** group might fail.

35.2. USING ID OVERRIDES TO ENABLE AD USERS TO ADMINISTER IDM

Prerequisites

- The **idm:DL1** stream is enabled on your Identity Management (IdM) server and you have switched to the RPMs delivered through this stream:

```
# yum module enable idm:DL1
# yum distro-sync
```

- The **idm:DL1/adtrust** profile is installed on your IdM server.

```
# yum module install idm:DL1/adtrust
```

The profile contains all the packages necessary for installing an IdM server that will have a trust agreement with Active Directory (AD), including the **ipa-idoverride-memberof** package.

- A working IdM environment is set up. For details, see [Installing Identity Management](#).
- A working trust between your IdM environment and AD is set up.

Procedure

This procedure describes creating and using an ID override for an AD user to give that user rights identical to those of an IdM user. During this procedure, work on an IdM server that is configured as a trust controller or a trust agent. For details on trust controllers and trust agents, see *Trust controllers and trust agents* in [Planning Identity Management](#).

- As an IdM administrator, create an ID override for an AD user in the Default Trust View. For example, to create an ID override for the **ad_user@ad.example.com** user:

```
# kinit admin
# ipa idoverrideuser-add 'default trust view' ad_user@ad.example.com
```

- Add the ID override from the Default Trust View as a member to an IdM group. If the group in question is a member of an IdM role, the AD user represented by the ID override will gain all permissions granted by the role when using the IdM API, including both the command line interface and the IdM web UI. For example, to add the ID override for the **ad_user@ad.example.com** user to the **admins** group:

```
# ipa group-add-member admins --idoverrideusers=ad_user@ad.example.com
```

35.3. MANAGING IDM CLI AS AN AD USER

This procedure checks that an Active Directory (AD) user can log into Identity Management (IdM) command-line interface (CLI) and run commands appropriate for his role.

- Destroy the current Kerberos ticket of the IdM administrator:

```
# kdestroy -A
```

NOTE



The destruction of the Kerberos ticket is required because the GSSAPI implementation in MIT Kerberos chooses credentials from the realm of the target service by preference, which in this case is the IdM realm. This means that if a credentials cache collection, namely the **KCM:**, **KEYRING:**, or **DIR:** type of credentials cache is in use, a previously obtained **admin** or any other IdM principal's credentials will be used to access the IdM API instead of the AD user's credentials.

- Obtain the Kerberos credentials of the AD user for whom an ID override has been created:

```
# kinit ad_user@AD.EXAMPLE.COM
Password for ad_user@AD.EXAMPLE.COM:
```

- Test that the ID override of the AD user enjoys the same privileges stemming from membership in the IdM group as any IdM user in that group. If the ID override of the AD user has been added to the **admins** group, the AD user can, for example, create groups in IdM:

```
# ipa group-add some-new-group
```

```
-----  
Added group "some-new-group"
```

```
-----  
Group name: some-new-group  
GID: 1997000011
```