Red Hat Enterprise Linux 9

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

Configuring users and hosts, managing them in groups, and controlling access with host-based (HBAC) and role-based access control (RBAC) rules
Red Hat Enterprise Linux 9 Managing IdM users, groups, hosts, and access control rules

Configuring users and hosts, managing them in groups, and controlling access with host-based (HBAC) and role-based access control (RBAC) rules
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** .................................................. 10

**PROVIDING FEEDBACK ON RED HAT DOCUMENTATION** .......................... 11

**CHAPTER 1. INTRODUCTION TO THE IDM COMMAND-LINE UTILITIES** .......... 12
  1.1. WHAT IS THE IPA COMMAND LINE INTERFACE ..................................... 12
  1.2. WHAT IS THE IPA HELP ........................................................................... 12
  1.3. USING IPA HELP TOPICS ....................................................................... 13
  1.4. USING IPA HELP COMMANDS ................................................................. 13
  1.5. STRUCTURE OF IPA COMMANDS .............................................................. 14
  1.6. USING AN IPA COMMAND TO ADD A USER ACCOUNT TO IDM ................. 15
  1.7. USING AN IPA COMMAND TO MODIFY A USER ACCOUNT IN IDM .......... 16
  1.8. HOW TO SUPPLY A LIST OF VALUES TO THE IDM UTILITIES ................. 17
  1.9. HOW TO USE SPECIAL CHARACTERS WITH THE IDM UTILITIES .......... 18

**CHAPTER 2. MANAGING USER ACCOUNTS USING THE COMMAND LINE** ........ 19
  2.1. USER LIFE CYCLE .................................................................................. 19
  2.2. ADDING USERS USING THE COMMAND LINE ........................................... 20
  2.3. ACTIVATING USERS USING THE COMMAND LINE .................................... 21
  2.4. PRESERVING USERS USING THE COMMAND LINE .................................. 22
  2.5. DELETING USERS USING THE COMMAND LINE ....................................... 22
  2.6. RESTORING USERS USING THE COMMAND LINE .................................... 23

**CHAPTER 3. MANAGING USER ACCOUNTS USING THE IDM WEB UI** .......... 25
  3.1. USER LIFE CYCLE .................................................................................. 25
  3.2. ADDING USERS IN THE WEB UI .............................................................. 26
  3.3. ACTIVATING STAGE USERS IN THE IDM WEB UI .................................. 28
  3.4. DISABLING USER ACCOUNTS IN THE WEB UI ........................................ 29
  3.5. ENABLING USER ACCOUNTS IN THE WEB UI ......................................... 30
  3.6. PRESERVING ACTIVE USERS IN THE IDM WEB UI ................................ 31
  3.7. RESTORING USERS IN THE IDM WEB UI ................................................ 32
  3.8. DELETING USERS IN THE IDM WEB UI .................................................... 33

**CHAPTER 4. MANAGING USER ACCOUNTS USING ANSIBLE PLAYBOOKS** .... 35
  4.1. USER LIFE CYCLE .................................................................................. 35
  4.2. ENSURING THE PRESENCE OF AN IDM USER USING AN ANSIBLE PLAYBOOK .................................................. 36
  4.3. ENSURING THE PRESENCE OF MULTIPLE IDM USERS USING ANSIBLE PLAYBOOKS .............................................. 38
  4.4. ENSURING THE PRESENCE OF MULTIPLE IDM USERS FROM A JSON FILE USING ANSIBLE PLAYBOOKS ................. 39
  4.5. ENSURING THE ABSENCE OF USERS USING ANSIBLE PLAYBOOKS .......... 41
  4.6. ADDITIONAL RESOURCES ..................................................................... 42

**CHAPTER 5. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT** ... 43
  5.1. SUDO ACCESS ON AN IDM CLIENT ......................................................... 43
  5.2. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE CLI .................................................. 43
  5.3. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT USING THE IDM WEB UI .............................................. 45
  5.4. CREATING A SUDO RULE ON THE CLI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT .................. 48
  5.5. CREATING A SUDO RULE IN THE IDM WEBUI THAT RUNS A COMMAND AS A SERVICE ACCOUNT ON AN IDM CLIENT .... 50
  5.6. ENABLING GSSAPI AUTHENTICATION FOR SUDO ON AN IDM CLIENT ... 56
  5.7. ENABLING GSSAPI AUTHENTICATION AND ENFORCING KERBEROS AUTHENTICATION INDICATORS FOR SUDO ON AN IDM CLIENT .................. 58
Red Hat Enterprise Linux 9 Managing IdM users, groups, hosts, and access control rules

5.8. SSSD OPTIONS CONTROLLING GSSAPI AUTHENTICATION FOR PAM SERVICES .......................... 60
5.9. TROUBLESHOOTING GSSAPI AUTHENTICATION FOR SUDO ........................................... 61
5.10. USING AN ANSIBLE PLAYBOOK TO ENSURE SUDDO ACCESS FOR AN IDM USER ON AN IDM CLIENT \v

CHAPTER 6. USING LDAPMODIFY TO MANAGE IDM USERS EXTERNALLY .............................. 66
6.1. TEMPLATES FOR MANAGING IDM USER ACCOUNTS EXTERNALLY .................................. 66
6.2. TEMPLATES FOR MANAGING IDM GROUP ACCOUNTS EXTERNALLY .................................. 68
6.3. PRESERVING AN IDM USER WITH LDAPMODIFY ............................................................... 69

CHAPTER 7. CONFIGURING IDM FOR EXTERNAL PROVISIONING OF USERS ......................... 72
7.1. PREPARING IDM ACCOUNTS FOR AUTOMATIC ACTIVATION OF STAGE USER ACCOUNTS ........ 72
7.2. CONFIGURING AUTOMATIC ACTIVATION OF IDM STAGE USER ACCOUNTS .................... 74
7.3. ADDING AN IDM STAGE USER DEFINED IN AN LDIF FILE ............................................... 76
7.4. ADDING AN IDM STAGE USER DIRECTLY FROM THE CLI USING LDAPMODIFY .................. 77
7.5. ADDITIONAL RESOURCES .................................................................................................. 79

CHAPTER 8. MANAGING SELF-SERVICE RULES IN IDM USING THE CLI .............................. 80
8.1. SELF-SERVICE ACCESS CONTROL IN IDM ........................................................................ 80
8.2. CREATING SELF-SERVICE RULES USING THE CLI ......................................................... 80
8.3. EDITING SELF-SERVICE RULES USING THE CLI ............................................................. 81
8.4. DELETING SELF-SERVICE RULES USING THE CLI .......................................................... 81

CHAPTER 9. MANAGING SELF-SERVICE RULES USING THE IDM WEB UI ......................... 83
9.1. SELF-SERVICE ACCESS CONTROL IN IDM ........................................................................ 83
9.2. CREATING SELF-SERVICE RULES USING THE IDM WEB UI ......................................... 83
9.3. EDITING SELF-SERVICE RULES USING THE IDM WEB UI ............................................ 85
9.4. DELETING SELF-SERVICE RULES USING THE IDM WEB UI ........................................ 86

CHAPTER 10. USING ANSIBLE PLAYBOOKS TO MANAGE SELF-SERVICE RULES IN IDM .... 87
10.1. SELF-SERVICE ACCESS CONTROL IN IDM ...................................................................... 87
10.2. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS PRESENT ....................... 87
10.3. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS ABSENT ......................... 89
10.4. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE HAS SPECIFIC ATTRIBUTES ... 90
10.5. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE DOES NOT HAVE SPECIFIC ATTRIBUTES 91

CHAPTER 11. MANAGING USER GROUPS IN IDM CLI ............................................................ 94
11.1. THE DIFFERENT GROUP TYPES IN IDM ............................................................................ 94
11.2. DIRECT AND INDIRECT GROUP MEMBERS ..................................................................... 95
11.3. ADDING A USER GROUP USING IDM CLI ....................................................................... 95
11.4. SEARCHING FOR USER GROUPS USING IDM CLI .......................................................... 96
11.5. DELETING A USER GROUP USING IDM CLI ................................................................. 96
11.6. ADDING A MEMBER TO A USER GROUP USING IDM CLI ............................................. 97
11.7. ADDING USERS WITHOUT A USER PRIVATE GROUP ....................................................... 98
11.7.1. Users without a user private group .............................................................................. 98
11.7.2. Adding a user without a user private group when private groups are globally enabled .... 98
11.7.3. Disabling user private groups globally for all users .................................................... 99
11.7.4. Adding a user when user private groups are globally disabled .................................. 99
11.8. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE IDM CLI ............................................................... 100
11.9. VIEWING GROUP MEMBERS USING IDM CLI ............................................................... 101
11.10. REMOVING A MEMBER FROM A USER GROUP USING IDM CLI .................................. 102
11.11. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE IDM CLI ............................................................... 102
CHAPTER 12. MANAGING USER GROUPS IN IDM WEB UI .......................................................... 104
12.1. THE DIFFERENT GROUP TYPES IN IDM ................................................................. 104
12.2. DIRECT AND INDIRECT GROUP MEMBERS ......................................................... 105
12.3. ADDING A USER GROUP USING IDM WEB UI ....................................................... 105
12.4. DELETING A USER GROUP USING IDM WEB UI .................................................... 106
12.5. ADDING A MEMBER TO A USER GROUP USING IDM WEB UI ............................... 107
12.6. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE WEB UI ................................................................. 108
12.7. VIEWING GROUP MEMBERS USING IDM WEB UI ................................................... 110
12.8. REMOVING A MEMBER FROM A USER GROUP USING IDM WEB UI ................. 110
12.9. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE WEB UI ................................................................. 111

CHAPTER 13. MANAGING USER GROUPS USING ANSIBLE PLAYBOOKS ................. 113
13.1. THE DIFFERENT GROUP TYPES IN IDM ................................................................. 113
13.2. DIRECT AND INDIRECT GROUP MEMBERS ......................................................... 114
13.3. ENSURING THE PRESENCE OF IDM GROUPS AND GROUP MEMBERS USING ANSIBLE PLAYBOOKS ................................................................................. 115
13.4. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS ................................................................. 116
13.5. ENSURING THE ABSENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS ................................................................. 117

CHAPTER 14. AUTOMATING GROUP MEMBERSHIP USING IDM CLI .......................... 120
14.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP ............................................. 120
14.2. AUTOMEMBER RULES ............................................................................................ 120
14.3. ADDING AN AUTOMEMBER RULE USING IDM CLI ............................................ 121
14.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM CLI ........... 122
14.5. VIEWING EXISTING AUTOMEMBER RULES USING IDM CLI ......................... 123
14.6. DELETING AN AUTOMEMBER RULE USING IDM CLI ......................................... 124
14.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM CLI .... 124
14.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM CLI ........ 125
14.9. CONFIGURING A DEFAULT AUTOMEMBER GROUP USING IDM CLI ................ 126

CHAPTER 15. AUTOMATING GROUP MEMBERSHIP USING IDM WEB UI ............... 128
15.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP ............................................. 128
15.2. AUTOMEMBER RULES ............................................................................................ 128
15.3. ADDING AN AUTOMEMBER RULE USING IDM WEB UI .................................. 129
15.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM WEB UI ....... 130
15.5. VIEWING EXISTING AUTOMEMBER RULES AND CONDITIONS USING IDM WEB UI ........................................................................................................ 131
15.6. DELETING AN AUTOMEMBER RULE USING IDM WEB UI ................................ 132
15.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM WEB UI ........................................................................................................ 133
15.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM WEB UI ........................................................................................................ 134
15.8.1. Rebuilding automatic membership for all users or hosts .................................. 134
15.8.2. Rebuilding automatic membership for a single user or host only ..................... 135
15.9. CONFIGURING A DEFAULT USER GROUP USING IDM WEB UI .......................... 136
15.10. CONFIGURING A DEFAULT HOST GROUP USING IDM WEB UI ...................... 136

CHAPTER 16. USING ANSIBLE TO AUTOMATE GROUP MEMBERSHIP IN IDM .... 138
16.1. PREPARING YOUR ANSIBLE CONTROL NODE FOR MANAGING IDM ............ 138
16.2. USING ANSIBLE TO ENSURE THAT AN AUTOMEMBER RULE FOR AN IDM USER GROUP IS PRESENT ................................................................. 140
16.3. USING ANSIBLE TO ENSURE THAT A SPECIFIED CONDITION IS PRESENT IN AN IDM USER GROUP AUTOMEMBER RULE ................................................. 141
16.4. USING ANSIBLE TO ENSURE THAT A CONDITION IS ABSENT FROM AN IDM USER GROUP ........................................................................................................ 141
CHAPTER 22. PREPARING YOUR ENVIRONMENT FOR MANAGING IDM USING ANSIBLE PLAYBOOKS

CHAPTER 23. USING ANSIBLE PLAYBOOKS TO MANAGE ROLE-BASED ACCESS CONTROL IN IDM

23.1. PERMISSIONS IN IDM
23.2. DEFAULT MANAGED PERMISSIONS
23.3. PRIVILEGES IN IDM
23.4. ROLES IN IDM
23.5. PREDEFINED ROLES IN IDENTITY MANAGEMENT
23.6. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE WITH PRIVILEGES IS PRESENT
23.7. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE IS ABSENT
23.8. USING ANSIBLE TO ENSURE THAT A GROUP OF USERS IS ASSIGNED TO AN IDM RBAC ROLE
23.9. USING ANSIBLE TO ENSURE THAT SPECIFIC USERS ARE NOT ASSIGNED TO AN IDM RBAC ROLE
23.10. USING ANSIBLE TO ENSURE A SERVICE IS A MEMBER OF AN IDM RBAC ROLE
23.11. USING ANSIBLE TO ENSURE A HOST IS A MEMBER OF AN IDM RBAC ROLE
23.12. USING ANSIBLE TO ENSURE A HOST GROUP IS A MEMBER OF AN IDM RBAC ROLE

CHAPTER 24. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PRIVILEGES

24.1. USING ANSIBLE TO ENSURE A CUSTOM IDM RBAC PRIVILEGE IS PRESENT
24.2. USING ANSIBLE TO ENSURE MEMBER PERMISSIONS ARE PRESENT IN A CUSTOM IDM RBAC PRIVILEGE
24.3. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE DOES NOT INCLUDE A PERMISSION
24.4. USING ANSIBLE TO RENAME A CUSTOM IDM RBAC PRIVILEGE
24.5. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE IS ABSENT
24.6. ADDITIONAL RESOURCES

CHAPTER 25. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PERMISSIONS IN IDM

25.1. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS PRESENT
25.2. USING ANSIBLE TO ENSURE AN RBAC PERMISSION WITH AN ATTRIBUTE IS PRESENT
25.3. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS ABSENT
25.4. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS A MEMBER OF AN IDM RBAC PERMISSION
25.5. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS NOT A MEMBER OF AN IDM RBAC PERMISSION
25.6. USING ANSIBLE TO RENAME AN IDM RBAC PERMISSION
25.7. ADDITIONAL RESOURCES

CHAPTER 26. USING AN ID VIEW TO OVERRIDE A USER ATTRIBUTE VALUE ON AN IDM CLIENT

26.1. ID VIEWS
26.2. POTENTIAL NEGATIVE IMPACT OF ID VIEWS ON SSSD PERFORMANCE
26.3. ATTRIBUTES AN ID VIEW CAN OVERRIDE
26.4. GETTING HELP FOR ID VIEW COMMANDS
26.5. USING AN ID VIEW TO OVERRIDE THE LOGIN NAME OF AN IDM USER ON A SPECIFIC HOST
26.6. MODIFYING AN IDM ID VIEW
26.7. ADDING AN ID VIEW TO OVERRIDE AN IDM USER HOME DIRECTORY ON AN IDM CLIENT
26.8. APPLYING AN ID VIEW TO AN IDM HOST GROUP
26.9. MIGRATING NIS DOMAINS TO IDENTITY MANAGEMENT

CHAPTER 27. USING ID VIEWS FOR ACTIVE DIRECTORY USERS

27.1. HOW THE DEFAULT TRUST VIEW WORKS
27.2. DEFINING GLOBAL ATTRIBUTES FOR AN AD USER BY MODIFYING THE DEFAULT TRUST VIEW
27.3. OVERRIDING DEFAULT TRUST VIEW ATTRIBUTES FOR AN AD USER ON AN IDM CLIENT WITH AN ID VIEW
27.4. APPLYING AN ID VIEW TO AN IDM HOST GROUP

CHAPTER 28. ADJUSTING ID RANGES MANUALLY

28.1. ID RANGES
39.1. ID OVERRIDES FOR AD USERS
39.2. USING ID OVERRIDES TO ENABLE AD USERS TO ADMINISTER IDM
39.3. MANAGING IDM CLI AS AN AD USER
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.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

- 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 feedback via Bugzilla, create a new 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. INTRODUCTION TO THE IDM COMMAND-LINE UTILITIES

The following sections describe the basics of using the Identity Management (IdM) command-line utilities.

Prerequisites

- Installed and accessible IdM server. For details, see Installing Identity Management.
- To use the IPA command-line interface, authenticate to IdM with a valid Kerberos ticket.

1.1. WHAT IS THE IPA COMMAND LINE INTERFACE

The IPA command-line interface (CLI) is the basic command-line interface for Identity Management (IdM) administration.

It supports a lot of subcommands for managing IdM, such as the `ipa user-add` command to add a new user.

IPA CLI allows you to:

- Add, manage, or remove users, groups, hosts and other objects in the network.
- Manage certificates.
- Search entries.
- Display and list objects.
- Set access rights.
- Get help with the correct command syntax.

1.2. WHAT IS THE IPA HELP

The IPA help is a built-in documentation system for the IdM server.

The IPA command-line interface (CLI) generates available help topics from loaded IdM plugin modules. To use the IPA help utility, you must:

- Have an IdM server installed and running.
- Be authenticated with a valid Kerberos ticket.

Entering the `ipa help` command without options displays information about basic help usage and the most common command examples.

You can use the following options for different `ipa help` use cases:

```plaintext
$ ipa help [TOPIC | COMMAND | topics | commands]
```
1.3. USING IPA HELP TOPICS

The following procedure describes how to use the IPA help in the command-line interface.

Procedure

1. Open a terminal and connect to the IdM server.

2. Enter `ipa help topics` to display a list of topics covered by help.

```$ ipa help topics```

3. Select one of the topics and create a command according to the following pattern: `ipa help [topic_name]`. Instead of the `topic_name` string, add one of the topics you listed in the previous step.

In the example, we use the following topic: `user`

```$ ipa help user```

4. If the IPA help output is too long and you cannot see the whole text, use the following syntax:

```$ ipa help user | less```

You can then scroll down and read the whole help.

The IPA CLI displays a help page for the `user` topic. After reading the overview, you can see many examples with patterns for working with topic commands.

1.4. USING IPA HELP COMMANDS

The following procedure describes how to create IPA help commands in the command-line interface.

Procedure

1. Open a terminal and connect to the IdM server.
2. Enter **ipa help commands** to display a list of commands covered by help.

   ```
   $ ipa help commands
   ```

3. Select one of the commands and create a help command according to the following pattern: **ipa help <COMMAND>**. Instead of the `<COMMAND>` string, add one of the commands you listed in the previous step.

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

Additional resources

- The **ipa** man page.

### 1.5. STRUCTURE OF IPA COMMANDS

The IPA CLI distinguishes the following types of commands:

- **Built-in commands** – Built-in commands are all available in the IdM server.
- **Plug-in provided commands**

The structure of IPA commands allows you to manage various types of objects. For example:

- Users,
- Hosts,
- DNS records,
- Certificates,

and many others.

For most of these objects, the IPA CLI includes commands to:

- Add (**add**)
- Modify (**mod**)
- Delete (**del**)
- Search (**find**)
- Display (**show**)

Commands have the following structure:

- **ipa user-add**, **ipa user-mod**, **ipa user-del**, **ipa user-find**, **ipa user-show**
- **ipa host-add**, **ipa host-mod**, **ipa host-del**, **ipa host-find**, **ipa host-show**
- **ipa dnsrecord-add**, **ipa dnsrecord-mod**, **ipa dnsrecord-del**, **ipa dnsrecord-find**, **ipa dnrecord-show**

You can create a user with the **ipa user-add [options]**, where [options] are optional. If you use just the **ipa user-add** command, the script asks you for details one by one.
To change an existing object, you need to define the object, therefore the command also includes an object: `ipa user-mod USER_NAME [options].`

1.6. USING AN IPA COMMAND TO ADD A USER ACCOUNT TO IDM

The following procedure describes how to add a new user to the Identity Management (IdM) database using the command line.

Prerequisites

- You need to have administrator privileges to add user accounts to the IdM server.

Procedure

1. Open a terminal and connect to the IdM server.

2. Enter the command for adding a new user:

   $ ipa user-add

   The command runs a script that prompts you to provide basic data necessary for creating a user account.

3. In the **First name:** field, enter the first name of the new user and press the **Enter** key.

4. In the **Last name:** field, enter the last name of the new user and press the **Enter** key.

5. In the **User login [suggested user name]:** enter the user name, or just press the **Enter** key to accept the suggested user name.
   The user name must be unique for the whole IdM database. If an error occurs because that user name already exists, repeat the process with the `ipa user-add` command and use a different, unique user name.

After you add the user name, the user account is added to the IdM database and the IPA command line interface (CLI) prints the following output:

```
----------------------
Added user "euser"
----------------------
User login: euser
First name: Example
Last name: User
Full name: Example User
Display name: Example User
Initials: EU
Home directory: /home/euser
GECOS: Example User
Login shell: /bin/sh
Principal name: euser@IDM.EXAMPLE.COM
Principal alias: euser@IDM.EXAMPLE.COM
Email address: euser@idm.example.com
UID: 427200006
GID: 427200006
```
NOTE

By default, a user password is not set for the user account. To add a password while creating a user account, use the `ipa user-add` command with the following syntax:

```
$ ipa user-add --first=Example --last=User --password
```

The IPA CLI then prompts you to add or confirm a user name and password.

If the user has been created already, you can add the password with the `ipa user-mod` command.

Additional resources

- Run the `ipa help user-add` command for more information about parameters.

1.7. USING AN IPA COMMAND TO MODIFY A USER ACCOUNT IN IDM

You can change many parameters for each user account. For example, you can add a new password to the user.

Basic command syntax is different from the `user-add` syntax because you need to define the existing user account for which you want to perform changes, for example, add a password.

Prerequisites

- You need to have administrator privileges to modify user accounts.

Procedure

1. Open a terminal and connect to the IdM server.

2. Enter the `ipa user-mod` command, specify the user to modify, and any options, such as `--password` for adding a password:

```
$ ipa user-mod euser --password
```

The command runs a script where you can add the new password.

3. Enter the new password and press the Enter key.

The IPA CLI prints the following output:

```
----------------------
Modified user "euser"
----------------------
User login: euser
First name: Example
Last name: User
Home directory: /home/euser
```
The user password is now set for the account and the user can log into IdM.

Additional resources
  - Run the `ipa help user-mod` command for more information about parameters.

1.8. HOW TO SUPPLY A LIST OF VALUES TO THE IDM UTILITIES

Identity Management (IdM) stores values for multi-valued attributes in lists. IdM supports the following methods of supplying multi-valued lists:

- Using the same command-line argument multiple times within the same command invocation:
  
  $ ipa permission-add --right=read --permissions=write --permissions=delete ...

- Alternatively, you can enclose the list in curly braces, in which case the shell performs the expansion:
  
  $ ipa permission-add --right={read,write,delete} ...

The examples above show a command `permission-add` which adds permissions to an object. The object is not mentioned in the example. Instead of ... you need to add the object for which you want to add permissions.

When you update such multi-valued attributes from the command line, IdM completely overwrites the previous list of values with a new list. Therefore, when updating a multi-valued attribute, you must specify the whole new list, not just a single value you want to add.

For example, in the command above, the list of permissions includes reading, writing and deleting. When you decide to update the list with the `permission-mod` command, you must add all values, otherwise those not mentioned will be deleted.

Example 1: The `ipa permission-mod` command updates all previously added permissions.

  $ ipa permission-mod --right=read --right=write --right=delete ...

  or

  $ ipa permission-mod --right={read,write,delete} ...

Example 2: The `ipa permission-mod` command deletes the `--right=delete` argument because it is not included in the command:

  $ ipa permission-mod --right=read --right=write ...
1.9. HOW TO USE SPECIAL CHARACTERS WITH THE IDM UTILITIES

When passing command-line arguments that include special characters to the `ipa` commands, escape these characters with a backslash (`\`). For example, common special characters include angle brackets (`<` and `>`) and vertical bar (`|`).

For example, to escape an asterisk (`*`):

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

Commands containing unescaped special characters do not work as expected because the shell cannot properly parse such characters.
CHAPTER 2. 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

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 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.

### 2.3. 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:

```bash
$ 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:

```bash
$ ipa user-find
```

This command lists all user accounts with details.

### 2.4. 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:

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

**NOTE**

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

### 2.5. 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.

### 2.6. 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:

```bash
$ 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:

```bash
$ ipa user-find
```

This command lists all user accounts with details.
CHAPTER 3. 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 is dismissed, delete the account without a backup.

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. 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**

1. Log in to the IdM Web UI.
2. 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.
3. Click the **+ Add** icon.
4. In the **Add stage user** dialog box, enter **First name** and **Last name** of the new user.
5. [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.
6. [Optional] In the **GID** drop down menu, select groups in which the user should be included.
7. [Optional] In the **Password** and **Verify password** fields, enter your password and confirm it, ensuring they both match.
8. Click on the **Add** button.
At this point, you can see the user account in the **Stage Users** table.

![Add stage user form](image)

**NOTE**

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

### 3.3. 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.

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.

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.

**NOTE**

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

### 3.4. 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

1. Log in to the IdM Web UI.
2. Go to Users → Active users tab.
3. Click the check-box of the user accounts you want to disable.
4. Click on the Disable button.
5. 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.

3.5. 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

1. Log in to the IdM Web UI.

2. Go to Users → Active users tab.

3. Click the check-box of the user accounts you want to enable.

4. Click on the Enable button.

5. 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.

3.6. 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 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.

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.**

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**.

### 3.7. 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.
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.
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.

### 3.8. 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.
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 4. 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.

### 4.1. 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.

![User Life Cycle Diagram]

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.

4.2. 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.
- 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 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:

   ```yaml
   ---
   - name: Playbook to handle users
     hosts: ipaserver
     become: true
   ```
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`: 
The user named `idm_user` is present in IdM.

### 4.3. 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`:

   ```yaml
   ---
   - 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
           passwordexpiration: "2023-01-19 23:59:59"
           password: "Password123"
         - name: idm_user_2
           first: Bob
           last: Acme
         ```
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:

```bash
$ 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:

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

2. Display information about `idm_user_1`:

   ```bash
   $ 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.

### 4.4. 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:

```yaml
---
- 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`:

```json
{
  "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:
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.

4.5. 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**:

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

### 4.6. ADDITIONAL RESOURCES

- See the [README-user.md](/usr/share/doc/ansible-freeipa/) Markdown file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/user` directory.
CHAPTER 5. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT

This section describes how to grant `sudo` access to users in Identity Management.

5.1. 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.

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

- See Managing sudo access.

5.2. 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:

5.3. 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

1. Add the /usr/sbin/reboot 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 you want the user to be able to perform using sudo: /usr/sbin/reboot.
   d. Click Add.

2. Use the new sudo command entry to create a sudo rule to allow idm_user to reboot the idmclient machine:
   b. Click Add in the upper right corner to open the Add sudo rule dialog box.
   c. Enter the name of the sudo rule: idm_user_reboot.
   d. Click Add and Edit
   e. Specify the user:
      i. In the Who section, check the Specified Users and Groups radio button.
      ii. In the User category the rule applies to subsection, click Add to open the Add users into sudo rule "idm_user_reboot" dialog box.
iii. 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.

Figure 5.2. Adding IdM sudo rule

h. 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.

5.4. 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"
   ```
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
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.
   ```

5.5. CREATING A SUUDO 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`.
   d. Click Add.

2. Use the new `sudo` command entry to create the new `sudo` rule:
   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`.
d. Click **Add and Edit**

e. Specify the user:

i. In the **Who** section, check the **Specified Users and Groups** radio button.

ii. 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.

iii. 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](image)

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 "run_third-party-app_report"** dialog box.

iii. 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.

![Add RunAs users into sudo rule 'run_third-party-app_report' dialog box](image)

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.
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.

   ```bash
   [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,
   ```
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.

5.6. 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.
- 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:
   ```

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

5.7. 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.
- 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 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.
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",
   ```
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

- SSSD options controlling GSSAPI authentication for PAM services
- 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

5.8. 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](#)

**5.9. 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:

  ```bash
  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:

  ```bash
  [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:

  ```bash
  No Kerberos credentials available
  ```

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

  ```bash
  [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:

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

  ```bash
  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:

  ```bash
  [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`:

  ```bash
  [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.

  ```bash
  [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
  ```

### 5.10. 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 link: 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:

   ```yaml
   [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:

      ```yaml
      ---
      - 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:

      ```bash
      $ 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:

      ```yaml
      ---
      - 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:

```bash
$ 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:

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

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

Additional resources

- See the `README-sudocmd.md`, `README-sudocmdgroup.md`, and `README-sudorule.md` files in the `/usr/share/doc/ansible-freeipa/` directory.
CHAPTER 6. 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.

6.1. 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**

```ldap
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**

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

- **Disabling a user**

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

- **Enabling a user**

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

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

- **Preserving a user**

```ldap
dn: distinguished_name
changetype: modrdn
newrdn: uid=user_login
```
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 and group information, for example activator or IdM administrator. The password in the example is this user’s password:

```bash
[...]
# 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
```

6.2. 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

```plaintext
dn: cn=group_name, cn=groups, cn=accounts, dc=idm, dc=example, dc=com
changetype: add
objectClass: top
objectClass: ipaobject
objectClass: ipausergroup
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.

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

6.3. 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:

```bash
$ 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:

```bash
# 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:

```bash
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:

```bash
changetype: modrdn
```

5. Specify the `newrdn` for the user:

```bash
newrdn: uid=user1
```

6. Indicate that you want to preserve the user:

```bash
deleteoldrdn: 0
```

7. Specify the `new superior DN`:

```bash
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:

```bash
[Enter]
```

```bash
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:

```bash
$ 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 7. 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

7.1. 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

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:
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:

- See Managing user accounts using the command line.
- See Delegating Permissions over Users.
- See Defining IdM Password Policies.

7.2. 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.
   ```

### 7.3. 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
   ```
2. 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
   217.6KB/s  00:00  100%  364
   ```

3. Use the SSH protocol to connect to the IdM server as provisionator:

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

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

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

5. 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,sn=staged
   users,dc=idm,dc=example,dc=com"
   ```

### 7.4. 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)
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:

```
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.

### 7.5. ADDITIONAL RESOURCES

- See [Using ldapmodify to manage IdM users externally](#).

CHAPTER 8. 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).

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 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"
```

80
8.3. 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:

```bash
$ 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
```

8.4. 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:

```bash
$ 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 9. 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).

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. 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:
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:

![Add Self Service Permission Window](image)

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.

### 9.3. 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.

In the **Self Service Permission: User Self service** page, you can select or deselect attributes to add or remove to the new self-service rule.
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.

### 9.4. 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.

3. A dialog opens, click on **Delete** to confirm.
CHAPTER 10. 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

10.1. 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.

10.2. 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 on the Ansible controller.

• 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:

   ```yaml
   ---
   - 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
   ```
10.3. 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 on the Ansible controller.
  - 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:

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

Additional resources

- See [Self-service access control in IdM](#).
- See the [README-selfservice.md](#) file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/selfservice` directory.

### 10.4. 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 on the Ansible controller.
  - 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:

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

```bash
$ 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:

   ```yaml
   ---
   - 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:

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

Additional resources

- See [Self-service access control in IdM](#).
- See the `README-selfservice.md` file available in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/selfservice` directory.

### 10.5. 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 on the Ansible controller.
  - 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:

   ```yaml
   ---
   - 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

- See Self-service access control in IdM.
- See the README-selfservice.md file in the /usr/share/doc/ansible-freeipa/ directory.
- See the sample playbooks in the /usr/share/doc/ansible-freeipa/playbooks/selfservice directory.
CHAPTER 11. 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

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

<table>
<thead>
<tr>
<th>Group name</th>
<th>Default group members</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipusers</td>
<td>All IdM users</td>
</tr>
<tr>
<td>admins</td>
<td>Users with administrative privileges, including the default admin user</td>
</tr>
<tr>
<td>editors</td>
<td>This is a legacy group that no longer has any special privileges</td>
</tr>
</tbody>
</table>
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 \texttt{admins} group.

\begin{warning}
\textbf{WARNING}

Do not delete the \texttt{admins} group. As \texttt{admins} is a pre-defined group required by IdM, this operation causes problems with certain commands.
\end{warning}

In addition, IdM creates \emph{user private groups} by default whenever a new user is created in IdM. For more information about private groups, see \href{Adding users without a private group}{Adding users without a private group}.

\section*{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 \emph{direct members} of group A.
- User 3, User 4, and User 5 are \emph{indirect members} of group A.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{Figure_11_1.png}
\caption{Direct and Indirect Group Membership}
\end{figure}

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

\section*{11.3. ADDING A USER GROUP USING IDM CLI}

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

\subsubsection*{Prerequisites}
You must be logged in as the administrator. For details, see Using kinit to log in to IdM manually.

Procedure

1. 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.

11.4. SEARCHING FOR USER GROUPS USING IDM CLI

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

Procedure

1. 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.

11.5. 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:

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

### 11.6. 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:

```bash
$ 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.

11.7. 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.

11.7.1. 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.

11.7.2. 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 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
  ```

11.7.3. 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

1. Obtain administrator privileges:

   ```
   $ kinit admin
   ```

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

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

3. 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
   ```

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

4. 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**

11.7.4. 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 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.

11.8. 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.

11.9. 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.
11.10. 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
   ```

11.11. 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**:

  ```bash
  $ 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.

  ```bash
  $ ipa group-show group_a
  Group name: group_a
  GID: 1133400009
  
  Additional resources**

  - See `ipa group-remove-member-manager --help` for more details.
CHAPTER 12. 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

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

<table>
<thead>
<tr>
<th>Group name</th>
<th>Default group members</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipausers</td>
<td>All IdM users</td>
</tr>
<tr>
<td>admins</td>
<td>Users with administrative privileges, including the default admin user</td>
</tr>
<tr>
<td>editors</td>
<td>This is a legacy group that no longer has any special privileges</td>
</tr>
<tr>
<td>Group name</td>
<td>Default group members</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>trust admins</td>
<td>Users with privileges to manage the Active Directory trusts</td>
</tr>
</tbody>
</table>

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. 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.

4. Click **Add** to confirm.

**12.4. 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.

12.5. 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.
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.
12.6. 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**.

7. Click **Add** to confirm.
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.

**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:
12.7. 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.

12.8. 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.

4. Select the check box next to the member you want to remove.

5. Click Delete.

6. Click Delete to confirm.

12.9. 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.
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:

**Additional resources**

- See `ipa group-add-member-manager --help` for more details.
CHAPTER 13. 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

13.1. 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 13.1. User groups created by default

<table>
<thead>
<tr>
<th>Group name</th>
<th>Default group members</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipausers</td>
<td>All IdM users</td>
</tr>
<tr>
<td>admins</td>
<td>Users with administrative privileges, including the default admin user</td>
</tr>
<tr>
<td>editors</td>
<td>This is a legacy group that no longer has any special privileges</td>
</tr>
<tr>
<td>trust admins</td>
<td>Users with privileges to manage the Active Directory trusts</td>
</tr>
</tbody>
</table>

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*.

### 13.2. 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 13.1. 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.
13.3. 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:

```yaml
---
- 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**

- See the `/usr/share/doc/ansible-freeipa/README-group.md` Markdown file.

### 13.4. 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:

```yaml
---
- 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.

**13.5. 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
       ipagroup:
         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 14. 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

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 CLI](#).

### 14.3. 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`. 

---

**SECTION:**

[Automember rules](#)

[Applying automember rules to existing entries using IdM CLI](#)

[Using kinit to log in to IdM manually](#)

[pcresyntax(3)](#)

121
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.

14.4. 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]: .*`
   ```
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:

```bash
$ 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.

### 14.5. 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:

   ```bash
   $ ipa automember-find
   ```
14.6. 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`.

14.7. 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
-------------------------------
```

14.8. 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 link: 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=option:

```bash
$ 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=option.

### 14.9. 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:

   ```bash
   $ 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 15. 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

15.1. 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.

15.2. 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**.

### 15.3. 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](#).

### 15.4. 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**.
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](image)

**15.5. 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.

15.6. 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**.

4. Click **Delete** to confirm.

### 15.7. 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**.
5. Click **Delete** to confirm.

### 15.8. 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](#).

#### 15.8.1. 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.
15.8.2. 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 **Actions → Rebuild auto membership**.
15.9. 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.

![User group rules](image)

15.10. 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.
CHAPTER 16. USING ANSIBLE TO AUTOMATE GROUP MEMBERSHIP IN IDM

Using automatic group membership, you can assign users and hosts user groups and host groups automatically, based on their attributes. For example, you can:

- Divide employees’ user entries into groups based on the employees’ manager, location, position or any other attribute. You can list all attributes by entering `ipa user-add --help` on the command-line.

- Divide hosts into groups based on their class, location, or any other attribute. You can list all attributes by entering `ipa host-add --help` on the command-line.

- Add all users or all hosts to a single global group.

You can use Red Hat Ansible Engine to automate the management of automatic group membership in Identity Management (IdM).

This section covers the following topics:

- Preparing your Ansible control node for managing IdM
- Using Ansible to ensure that an automember rule for an IdM user group is present
- Using Ansible to ensure that a condition is present in an IdM user group automember rule
- Using Ansible to ensure that a condition is absent in an IdM user group automember rule
- Using Ansible to ensure that an automember rule for an IdM group is absent
- Using Ansible to ensure that a condition is present in an IdM host group automember rule

16.1. PREPARING YOUR ANSIBLE CONTROL NODE FOR MANAGING IDM

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.

By following 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:
   
   ```bash
   $ mkdir ~/MyPlaybooks/
   ```

2. Change into the `~/MyPlaybooks/` directory:
   
   ```bash
   $ 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:

   ```bash
   $ ssh-keygen
   ```

6. Copy the SSH public key to the IdM admin account on each managed node:
You must enter the IdM admin password when you enter these commands.

Additional resources

- Installing an Identity Management server using an Ansible playbook.
- How to build your inventory.

16.2. USING ANSIBLE TO ENSURE THAT AN AUTOMEMBER RULE FOR AN IDM USER GROUP IS PRESENT

The following procedure describes how to use an Ansible playbook to ensure an automember rule for an Identity Management (IdM) group exists. In the example, the presence of an automember rule is ensured for the testing_group user group.

Prerequisites

- You know the IdM admin password.
- The testing_group user group exists in IdM.
- 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 on the Ansible controller.
  - 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 your ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks/

2. Copy the automember-group-present.yml Ansible playbook file located in the /usr/share/doc/ansible-freeipa/playbooks/automember/ directory:

   $ cp /usr/share/doc/ansible-freeipa/playbooks/automember/automember-group-present.yml automember-group-present-copy.yml

3. Open the automember-group-present-copy.yml file for editing.

4. Adapt the file by setting the following variables in the ipaautomember task section:
   - Set the ipaadmin_password variable to the password of the IdM admin.
   - Set the name variable to testing_group.
   - Set the automember_type variable to group.
• Ensure that the state variable is set to present.

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

```yaml
---
- name: Automember group present example
  hosts: ipaserver
  become: true
  tasks:
    - name: Ensure group automember rule admins is present
      ipaautomember:
        ipaadmin_password: Secret123
        name: testing_group
        automember_type: group
        state: present
```

5. Save the file.

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

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

Additional resources

• See Benefits of automatic group membership and Automember rules.

• See Using Ansible to ensure that a condition is present in an IdM user group automember rule.

• See the README-automember.md file in the /usr/share/doc/ansible-freeipa/ directory.

• See the /usr/share/doc/ansible-freeipa/playbooks/automember directory.

16.3. USING ANSIBLE TO ENSURE THAT A SPECIFIED CONDITION IS PRESENT IN AN IDM USER GROUP AUTOMEMBER RULE

The following procedure describes how to use an Ansible playbook to ensure that a specified condition exists in an automember rule for an Identity Management (IdM) group. In the example, the presence of a UID-related condition in the automember rule is ensured for the testing_group group. By specifying the .* condition, you ensure that all future IdM users automatically become members of the testing_group.

Prerequisites

• You know the IdM admin password.

• The testing_group user group and automember user group rule exist in IdM.

• 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 on the Ansible controller.
  • 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 your `~/MyPlaybooks/` directory:

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

2. Copy the `automember-hostgroup-rule-present.yml` Ansible playbook file located in the `/usr/share/doc/ansible-freeipa/playbooks/automember/` directory and name it, for example, `automember-usergroup-rule-present.yml`:

   ```bash
   $ cp /usr/share/doc/ansible-freeipa/playbooks/automember/automember-hostgroup-rule-present.yml automember-usergroup-rule-present.yml
   ```

3. Open the `automember-usergroup-rule-present.yml` file for editing.

4. Adapt the file by modifying the following parameters:

   - Rename the playbook to correspond to your use case, for example: *Automember user group rule member present*.
   - Rename the task to correspond to your use case, for example: *Ensure an automember condition for a user group is present*.
   - Set the following variables in the `ipaautomember` task section:
     - Set the `ipaadmin_password` variable to the password of the IdM admin.
     - Set the `name` variable to `testing_group`.
     - Set the `automember_type` variable to `group`.
     - Ensure that the `state` variable is set to `present`.
     - Ensure that the `action` variable is set to `member`.
     - Set the `inclusive key` variable to `UID`.
     - Set the `inclusive expression` variable to `.*`

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

   ```yaml
   ---
   - name: Automember user group rule member present
     hosts: ipaserver
     become: true
     tasks:
       - name: Ensure an automember condition for a user group is present
         ipaautomember:
           ipaadmin_password: Secret123
           name: testing_group
           automember_type: group
           state: present
           action: member
           inclusive:
             - key: UID
               expression: .*
   ```
5. Save the file.

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

$$ \texttt{ansible-playbook -v -i inventory automember-usergroup-rule-present.yml}$$

Verification steps

1. Log in as an IdM administrator.

   $$ \texttt{kinit admin}$$

2. Add a user, for example:

   $$ \texttt{ipa user-add user101 --first user --last 101}$$
   
   Added user "user101"
   
   User login: user101
   First name: user
   Last name: 101
   ...
   Member of groups: ipausers, testing_group
   ...

Additional resources

- See Applying automember rules to existing entries using the IdM CLI.
- See Benefits of automatic group membership and Automember rules.
- See the README-automember.md file in the /usr/share/doc/ansible-freeipa/ directory.
- See the /usr/share/doc/ansible-freeipa/playbooks/automember directory.

16.4. USING ANSIBLE TO ENSURE THAT A CONDITION IS ABSENT FROM AN IDM USER GROUP AUTOMEMBER RULE

The following procedure describes how to use an Ansible playbook to ensure a condition is absent from an automember rule for an Identity Management (IdM) group. In the example, the absence of a condition in the automember rule is ensured that specifies that users whose initials are dp should be included. The automember rule is applied to the testing_group group. By applying the condition, you ensure that no future IdM user whose initials are dp becomes a member of the testing_group.

Prerequisites

- You know the IdM admin password.
- The testing_group user group and automember user group rule exist in IdM.
- 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 on the Ansible controller.

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 your ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks/

2. Copy the automember-hostgroup-rule-absent.yml Ansible playbook file located in the /usr/share/doc/ansible-freeipa/playbooks/automember/ directory and name it, for example, automember-usergroup-rule-absent.yml:

   $ cp /usr/share/doc/ansible-freeipa/playbooks/automember/automember-hostgroup-rule-absent.yml automember-usergroup-rule-absent.yml

3. Open the automember-usergroup-rule-absent.yml file for editing.

4. Adapt the file by modifying the following parameters:

   - Rename the playbook to correspond to your use case, for example: Automember user group rule member absent.

   - Rename the task to correspond to your use case, for example: Ensure an automember condition for a user group is absent.

   - Set the following variables in the ipaautomember task section:

     - Set the ipaadmin_password variable to the password of the IdM admin.

     - Set the name variable to testing_group.

     - Set the automember_type variable to group.

     - Ensure that the state variable is set to absent.

     - Ensure that the action variable is set to member.

     - Set the inclusive key variable to initials.

     - Set the inclusive expression variable to dp.

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

---
- name: Automember user group rule member absent
  hosts: ipaserver
  become: true
  tasks:
  - name: Ensure an automember condition for a user group is absent
    ipaautomember:
      ipaadmin_password: Secret123
      name: testing_group
      automember_type: group
5. Save the file.

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

```bash
$ ansible-playbook -v -i inventory automember-usergroup-rule-absent.yml
```

**Verification steps**

1. Log in as an IdM administrator.

```bash
$ kinit admin
```

2. View the automember group:

```bash
$ ipa automember-show --type=group testing_group
```

Automember Rule: testing_group

The absence of an **Inclusive Regex: initials=dp** entry in the output confirms that the **testing_group** automember rule does not contain the condition specified.

**Additional resources**

- See [Applying automember rules to existing entries using the IdM CLI](#).
- See [Benefits of automatic group membership](#) and [Automember rules](#).
- See the [README-automember.md](#) file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the `/usr/share/doc/ansible-freeipa/playbooks/automember` directory.

### 16.5. USING ANSIBLE TO ENSURE THAT AN AUTOMEMBER RULE FOR AN IDM USER GROUP IS ABSENT

The following procedure describes how to use an Ansible playbook to ensure an **automember** rule is absent for an Identity Management (IdM) group. In the example, the absence of an **automember** rule is ensured for the **testing_group** group.

**NOTE**

Deleting an automember rule also deletes all conditions associated with the rule. To remove only specific conditions from a rule, see [Using Ansible to ensure that a condition is absent in an IdM user group automember rule](#).

**Prerequisites**

- You know the IdM **admin** 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 on the Ansible controller.
- 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 your `~/.MyPlaybooks/` directory:

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

2. Copy the `automember-group-absent.yml` Ansible playbook file located in the `/usr/share/doc/ansible-freeipa/playbooks/automember/` directory:

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

3. Open the `automember-group-absent-copy.yml` file for editing.

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

   - Set the `ipaadmin_password` variable to the password of the IdM admin.
   - Set the `name` variable to `testing_group`.
   - Set the `automember_type` variable to `group`.
   - Ensure that the `state` variable is set to `absent`.

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

   ```
   ---
   - name: Automember group absent example
     hosts: ipaserver
     become: true
     tasks:
       - name: Ensure group automember rule admins is absent
         ipaautomember:
           ipaadmin_password: Secret123
           name: testing_group
           automember_type: group
           state: absent
   ```

5. Save the file.

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

   ```
   $ ansible-playbook -v -i inventory automember-group-absent.yml
   ```

Additional resources
16.6. USING ANSIBLE TO ENSURE THAT A CONDITION IS PRESENT IN AN IDM HOST GROUP AUTOMEMBER RULE

This section describes how to use Ansible to ensure that a condition is present in an IdM host group automember rule. The example describes how to ensure that hosts with the FQDN of .*.idm.example.com are members of the primary_dns_domain_hosts host group and hosts whose FQDN is *.example.org are not members of the primary_dns_domain_hosts host group.

Prerequisites

- You know the IdM admin password.
- The primary_dns_domain_hosts host group and automember host group rule exist in IdM.
- 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 on the Ansible controller.
  - 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 your ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks/

2. Copy the automember-hostgroup-rule-present.yml Ansible playbook file located in the /usr/share/doc/ansible-freeipa/playbooks/autmember/ directory:

   $ cp /usr/share/doc/ansible-freeipa/playbooks/autmember/automember-hostgroup-rule-present.yml automember-hostgroup-rule-present-copy.yml

3. Open the automember-hostgroup-rule-present-copy.yml file for editing.

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

   - Set the ipaadmin_password variable to the password of the IdM admin.
   - Set the name variable to primary_dns_domain_hosts.
   - Set the automember_type variable to hostgroup.
   - Ensure that the state variable is set to present.
   - Ensure that the action variable is set to member.
- Ensure that the **inclusive key** variable is set to **fqdn**.
- Set the corresponding **inclusive expression** variable to `.*.idm.example.com`.
- Set the **exclusive key** variable to **fqdn**.
- Set the corresponding **exclusive expression** variable to `.*.example.org`.

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

```yaml
---
- name: Automember user group rule member present
  hosts: ipaserver
  become: true
  tasks:
  - name: Ensure an automember condition for a user group is present
    ipaautomember:
      ipaadmin_password: Secret123
      name: primary_dns_domain_hosts
      automember_type: hostgroup
      state: present
      action: member
    inclusive:
      - key: fqdn
        expression: `.*.idm.example.com`
    exclusive:
      - key: fqdn
        expression: `.*.example.org`
```

5. Save the file.

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

   ```bash
   $ ansible-playbook -v -i inventory automember-hostgroup-rule-present-copy.yml
   ```

**Additional resources**

- See [Applying automember rules to existing entries using the IdM CLI](#).
- See [Benefits of automatic group membership](#) and [Automember rules](#).
- See the [README-automember.md](#) file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the `/usr/share/doc/ansible-freeipa/playbooks/automember` directory.

### 16.7. ADDITIONAL RESOURCES

- Managing user accounts using Ansible playbooks
- Managing hosts using Ansible playbooks
- Managing user groups using Ansible playbooks
- Managing host groups using the IdM CLI
CHAPTER 17. 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
- View existing delegation rules using IdM CLI
- Modifying a delegation rule using IdM CLI
- Deleting a delegation rule using IdM CLI

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 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=productcategory --attrs=departmentnumber --attrs=employeetype --
    attrs=employeenumber --group=managers --membergroup=employees
```

```
Added delegation "basic manager attributes"
```

Delegation name: basic manager attributes  
Permissions: read, write  
Attributes: productcategory, departmentnumber, employeetype, employeenumber  
Member user group: employees  
User group: managers

### 17.3. 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: productcategory, departmentnumber, employeetype, employeenumber  
Member user group: employees  
User group: managers
```

```
Number of entries returned 1
```

### 17.4. 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
```

17.5. 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 18. 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

18.1. 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.

18.2. 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.
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.
Click the **Add** button to save the new delegation rule.

**18.3. VIEWING EXISTING DELEGATION RULES USING IDM WEBUI**

This section describes how to view existing delegation rules using the IdM WebUI.

**Prerequisites**
CHAPTER 18. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM WEBUI

- 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.**

**18.4. 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**.

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.

Click the Save button to save the changes.

18.5. 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**

1. From the IPA Server menu, click Role-Based Access Control → Delegations.

2. Select the check box next to the rule you want to remove.

3. Click Delete.
4. Click **Delete** to confirm.
CHAPTER 19. 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

19.1. 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.

19.2. 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:

```ini
[defaults]
inventory = /home/<username>/MyPlaybooks/inventory

[privilege_escalation]
become=True
```

4. Create the ~/MyPlaybooks/inventory file with the following content:

```ini
[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.

19.3. 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 on the Ansible controller.
  - 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:

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

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

   ```bash
   $ 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 ipadmin_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:

   ```yaml
   ---
   - name: Playbook to manage a delegation rule
     hosts: ipaserver
     become: true

     tasks:
     - name: Ensure delegation "basic manager attributes" is present
       ipadelegation:
         ipadmin_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

- See Delegation rules.
- See the README-delegation.md file in the /usr/share/doc/ansible-freeipa/ directory.
- See the sample playbooks in the /usr/share/doc/ansible-freeipa/playbooks/ipadelegation directory.

19.4. 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 on the Ansible controller.
  - 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:

```yaml
---
- 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**

- See [Delegation rules](#).
- See the [README-delegation.md](#) file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/ipadelegation` directory.

### 19.5. 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 on the Ansible controller.
  - 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

- See Delegation rules.

- See the `README-delegation.md` file in the `/usr/share/doc/ansible-freeipa/` directory.
See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/ipadelegation` directory.

### 19.6. 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 on the Ansible controller.
  - 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:
   ```bash
   $ cd ~/MyPlaybooks/
   ```

2. Make a copy of the `delegation-member-absent.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/delegation/` directory:
   ```bash
   $ 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

- See Delegation rules.
- See the README-delegation.md file in the /usr/share/doc/ansible-freeipa/ directory.
- See the sample playbooks in the /usr/share/doc/ansible-freeipa/playbooks/ipadelegation directory.
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

20.1. 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.

20.1.1. 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):
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.

### 20.1.2. 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.

20.1.3. 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.
20.1.4. 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.

20.1.5. Predefined roles in Identity Management

Red Hat Identity Management provides the following range of pre-defined roles:

Table 20.1. Predefined Roles in Identity Management

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrollment Administrator</td>
<td>Host Enrollment</td>
<td>Responsible for client, or host, enrollment</td>
</tr>
<tr>
<td>helpdesk</td>
<td>Modify Users and Reset passwords, Modify Group membership</td>
<td>Responsible for performing simple user administration tasks</td>
</tr>
<tr>
<td>IT Security Specialist</td>
<td>Netgroups Administrators, HBAC Administrator, Sudo Administrator</td>
<td>Responsible for managing security policy such as host-based access controls, sudo rules</td>
</tr>
<tr>
<td>IT Specialist</td>
<td>Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators</td>
<td>Responsible for managing hosts</td>
</tr>
<tr>
<td>Security Architect</td>
<td>Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator</td>
<td>Responsible for managing the Identity Management environment, creating trusts, creating replication agreements</td>
</tr>
</tbody>
</table>
20.2. 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

1. Create new permission entries with the `ipa permission-add` command. For example, to add a permission named `dns admin`:

   ```
   $ ipa permission-add "dns admin"
   ```

2. 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:

  ```bash
  $ 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:

  ```bash
  $ 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:

  ```bash
  $ 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:

  ```bash
  $ 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.

### 20.3. 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**

- See the `ipa` man page.
- See the `ipa help` command.

### 20.4. 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 link: [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"
   ```

20.5. 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

- See the `ipa` man page.
- See the `ipa help` command.

20.6. 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
   ```
2. Add the required privileges to the role using the `ipa role-add-privilege` command:

```bash
$ 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:

```bash
$ ipa role-add-member --groups=useradmins useradmin
Role name: useradmin
Description: User Administrator
Member groups: useradmins
Privileges: user administrators
--------------------------------------------------
Number of members added 1
--------------------------------------------------
```

### 20.7. 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**

- See the `ipa` man page
- See the `ipa help` command.
CHAPTER 21. 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

21.1. 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.

21.1.1. 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.

### 21.1.2. 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.

21.1.3. 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.

21.1.4. 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 cannot contain other roles.

21.1.5. 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

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrollment Administrator</td>
<td>Host Enrollment</td>
<td>Responsible for client, or host, enrollment</td>
</tr>
<tr>
<td>helpdesk</td>
<td>Modify Users and Reset passwords, Modify Group membership</td>
<td>Responsible for performing simple user administration tasks</td>
</tr>
<tr>
<td>IT Security Specialist</td>
<td>Netgroups Administrators, HBAC Administrator, Sudo Administrator</td>
<td>Responsible for managing security policy such as host-based access controls, sudo rules</td>
</tr>
<tr>
<td>IT Specialist</td>
<td>Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators</td>
<td>Responsible for managing hosts</td>
</tr>
<tr>
<td>Security Architect</td>
<td>Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator</td>
<td>Responsible for managing the Identity Management environment, creating trusts, creating replication agreements</td>
</tr>
</tbody>
</table>
21.2. 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

1. To add a new permission, open the Role-Based Access Control sub-menu in the IPA Server tab and select Permissions:

2. The list of permissions opens: Click the Add button at the top of the list of the permissions:
3. The **Add Permission** form opens. Specify the name of the new permission and define its properties accordingly:

<table>
<thead>
<tr>
<th>Add Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Permission name</strong></td>
</tr>
<tr>
<td><strong>Bind rule type</strong></td>
</tr>
<tr>
<td><strong>Granted rights</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Type</strong></td>
</tr>
<tr>
<td><strong>Subtree</strong></td>
</tr>
<tr>
<td><strong>Extra target filter</strong></td>
</tr>
<tr>
<td><strong>Target DN</strong></td>
</tr>
<tr>
<td><strong>Member of group</strong></td>
</tr>
<tr>
<td><strong>Effective attributes</strong></td>
</tr>
</tbody>
</table>

* 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 nonexistent 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 form](image)

**21.3. 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

1. To add a new privilege, open the **Role-Based Access Control** sub-menu in the **IPA Server** tab and select **Privileges**:

   ![Image of Role-Based Access Control menu](image)

2. The list of privileges opens. Click the **Add** button at the top of the list of privileges:

   ![Image of Add button](image)

3. The **Add Privilege** form opens. Enter the name and a description of the privilege:

   ![Image of Add Privilege form](image)

4. Click the **Add and Edit** button in order to save the new privilege and continue to the privilege configuration page to add permissions.

5. 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:

![Permissions Tab](image)

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:

![Add Privilege New Privilege into Permissions](image)

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.

### 21.4. 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**:

2. The list of roles opens. Click the **Add** button at the top of the list of the role-based access control instructions.
3. The **Add Role** form opens. Enter the role name and a description:

```
Add Role

Role name *: Example Role
Description: For engineers

* Required field
```

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).
7. In the window that opens, select the members on the left and use the > button to move them to the **Prospective** column.

8. At the top of the **Privileges** tab, click **Add**.
9. Select the privileges on the left and use the > button to move them to the Prospective column.

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 22. 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:

   ```sh
   $ mkdir ~/MyPlaybooks/
   ```

2. Change into the ~/MyPlaybooks/ directory:

   ```sh
   $ 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

- See Installing an Identity Management server using an Ansible playbook.
- See How to build your inventory.
CHAPTER 23. 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

23.1. 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
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.

### 23.2. 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.
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

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.
23.3. 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.

23.4. 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.

23.5. PREDEFINED ROLES IN IDENTITY MANAGEMENT

Red Hat Identity Management provides the following range of pre-defined roles:

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrollment Administrator</td>
<td>Host Enrollment</td>
<td>Responsible for client, or host, enrollment</td>
</tr>
</tbody>
</table>
## 23.6. 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 controller.
- 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.

<table>
<thead>
<tr>
<th>Role</th>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>helpdesk</td>
<td>Modify Users and Reset passwords, Modify Group membership</td>
<td>Responsible for performing simple user administration tasks</td>
</tr>
<tr>
<td>IT Security Specialist</td>
<td>Netgroups Administrators, HBAC Administrator, Sudo Administrator</td>
<td>Responsible for managing security policy such as host-based access controls, sudo rules</td>
</tr>
<tr>
<td>IT Specialist</td>
<td>Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators</td>
<td>Responsible for managing hosts</td>
</tr>
<tr>
<td>Security Architect</td>
<td>Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator</td>
<td>Responsible for managing the Identity Management environment, creating trusts, creating replication agreements</td>
</tr>
<tr>
<td>User Administrator</td>
<td>User Administrators, Group Administrators, Stage User Administrators</td>
<td>Responsible for creating users and groups</td>
</tr>
</tbody>
</table>

The procedure to define a custom RBAC role with privileges involves the following steps:

1. **Create an Ansible Playbook:**
   - Define a playbook in your Ansible environment to manage the RBAC role.
   - The playbook should include tasks that add the role, define the privileges, and ensure the role is applied correctly.

2. **Define Privileges:**
   - Identify the specific privileges that are required for the custom role.
   - Use Ansible modules to add these privileges to the new role.

3. **Apply the Role:**
   - Run the Ansible playbook to apply the custom role to the Identity Management environment.
   - Verify that the role is present and that the privileges are correctly assigned.

4. **Test the Role:**
   - Use the new role to manage access to resources in the Identity Management environment.
   - Ensure that the role behaves as intended and that users with the correct privileges can access resources.

By following these steps, you can effectively use Ansible to manage role-based access control in Identity Management (IdM), ensuring that users have the appropriate level of access to resources according to their role and responsibilities.
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:
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 controller.
- 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:
   ```bash
   $ cd ~/<MyPlaybooks>/
   ```

2. Make a copy of the `role-is-absent.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:
   ```bash
   $ 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 `ipadmin_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:

```yaml
---
- 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:

   ```bash
   $ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-is-absent-copy.yml
   ```

**Additional resources**

- See [Encrypting content with Ansible Vault](#).
- See [Roles in IdM](#).
- See the [README-role](#) Markdown file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

### 23.8. 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 controller.
- 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:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```yaml
   ---
   - 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:

   ```bash
   $ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-group-present-copy.yml
   ```

Additional resources

- See Encrypting content with Ansible Vault.
- See Roles in IdM.
- See the README-role Markdown file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.
23.9. 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 controller.
- 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
```
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 controller.
- 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:
2. Make a copy of the `role-member-service-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/role/` directory:

```bash
$ 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:

   ```yaml
   ---
   - 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:

   ```bash
   $ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-service-present-copy.yml
   ```

Additional resources

- See [Encrypting content with Ansible Vault](#).
- See [Roles in IdM](#).
- See the `README-role` Markdown file in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.
23.11. 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 controller.
- 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:

```yaml
---
- 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

- See Encrypting content with Ansible Vault.
- See Roles in IdM.
- See the README-role Markdown file in the /usr/share/doc/ansible-freeipa/ directory.
- See the sample playbooks in the /usr/share/doc/ansible-freeipa/playbooks/iparole directory.

23.12. 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 controller.
- 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:
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:

   ```yaml
   ---
   - 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:

   ```bash
   $ ansible-playbook -v -i ~/<MyPlaybooks>/inventory role-member-hostgroup-present-copy.yml
   ```

   Additional resources

   - See [Encrypting content with Ansible Vault](#).
   - See [Roles in IdM](#).
   - See the [README-role](#) Markdown file in the `/usr/share/doc/ansible-freeipa/` directory.
   - See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.
CHAPTER 24. 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.

24.1. 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 controller.
- 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:

   ```bash
   $ 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:

   ```yaml
   ---
   - 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:

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

24.2. 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 controller.

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:

```yaml
---
- 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"
```
CHAPTER 24. USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PRIVILEGES

- "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

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

```bash
$ cp /usr/share/doc/ansible-freeipa/playbooks/privilege/privilege-member-present.yml
privilege-member-present-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:

```yaml
---
- 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:

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

### 24.4. 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 controller.
- 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:

```yaml
---
- 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`

### 24.5. 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 controller.
- 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:

   ```bash
   $ 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:

   ```
   - name: Ensure privilege "CA administrator" is absent
     ipaprivilege:
   ```

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

   ```yaml
   - name: Privilege absent example
     hosts: ipaserver
     become: true
     tasks:
     - name: Ensure privilege "CA administrator" is absent
       ipaprivilege:
   ```

6. Save the file.

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

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

### 24.6. ADDITIONAL RESOURCES

- See [Privileges in IdM](#).
- See [Permissions in IdM](#).
- See the `README-privilege` file available in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/ipaprivilege` directory.
CHAPTER 25. 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.

25.1. 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 controller.
- 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:
   ```bash
   $ cd ~/MyPlaybooks/
   ```

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

   ```yaml
   ---
   - 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:
   ```bash
   $ ansible-playbook -v -i inventory permission-present-copy.yml
   ```
25.2. 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 controller.
- 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:

   ```bash
   $ cd ~/MyPlaybooks/
   ```
2. Make a copy of the permission-present.yml file located in the /usr/share/doc/ansible-freeipa/playbooks/permission/ directory:

```bash
$ 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:

   ```yaml
   ---
   - 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:

   ```bash
   $ ansible-playbook -v -i inventory permission-present-with-attribute.yml
   ```

Additional resources


### 25.3. 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 controller.
- 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:
   ```bash
   $ cd ~/MyPlaybooks/
   ```
2. Make a copy of the permission-absent.yml file located in the /usr/share/doc/ansible-freeipa/playbooks/permission/ directory:
   ```bash
   $ 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:
   ```yaml
   ---
   - 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:
   ```bash
   $ ansible-playbook -v -i inventory permission-absent-copy.yml
   ```
25.4. 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 controller.
- 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 \texttt{action} variable is set to \texttt{member}.

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

```yaml
---
- 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:

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

25.5. 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 \texttt{MyPermission} permission exists.
- The host entries created by a user with a privilege that contains the \texttt{MyPermission} permission cannot have the \texttt{description} attribute.

\textbf{Prerequisites}

- You know the IdM administrator password.
- You have installed the \texttt{ansible-freeipa} package on the Ansible controller.
- The example assumes that you have created and configured the \texttt{~/MyPlaybooks/} directory as a central location to store copies of sample playbooks.
- The \texttt{MyPermission} permission exists.
**Procedure**

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

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

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

   ```bash
   $ 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:

   ```yaml
   ---
   - 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:

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

**25.6. 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 controller.
- 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 **ipadmin_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:
         ipadmin_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
```

### 25.7. ADDITIONAL RESOURCES

- See [Permissions in IdM](#).
- See [Privileges in IdM](#).
- See the [README-permission](#) file available in the `/usr/share/doc/ansible-freeipa/` directory.
- See the sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/ipapermission` directory.
CHAPTER 26. 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

26.1. 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

- Using ID views for Active Directory users
- SSSD Client-side Views

### 26.2. 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.

### 26.3. 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`)

**Group attributes**

- Group name (`gidName`)
- Group GID (`gidNumber`)
- Group GECOS (`gecos`)
- Group description (`description`)
- Group creation time (`ctime`)

**Group membership attributes**

- User attribute (`memberUid`)
- User attribute (`memberGid`)
- User attribute (`memberUidNumber`)
- User attribute (`memberGidNumber`)
- User attribute (`memberLoginShell`)
- User attribute (`memberLoginShellNumber`)

**Group membership role attributes**

- Group attribute (`memberRole`)
• Home directory (homeDirectory)
• SSH public keys (ipaSshPubkey)
• Certificate (userCertificate)

Group attributes

• Group name (cn)
• Group GiD number (gidNumber)

26.4. 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:

```bash
$ 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:

```bash
$ 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
[...]
```

26.5. 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:
     [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
  ```
26.6. 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 --
   ```
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:
      ```bash
      $ ssh root@host1
      Password:
      
      root@host1 ~]$ sss_cache -E
      
      root@host1 ~]# systemctl restart sssd
      ```
   b. Clear the SSSD cache:
   c. Restart the SSSD daemon:

Verification steps

1. **SSH** to `host1` as `idm_user`:
   ```bash
   [root@r8server ~]# ssh idm_user@host1.idm.example.com
   Password:
   
   [user_1234@host1 ~]$ 
   ```

2. Print the working directory:
   ```bash
   [user_1234@host1 ~]$ pwd
   /home/user_1234/
   ```

Additional resources

- Defining global attributes for an AD user by modifying the Default Trust View

### 26.7. 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**

- Red Hat Enterprise Linux 9 Managing IdM users, groups, hosts, and access control rules
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:

   ```bash
   [root@host1 /]# mkdir /home/user_1234/
   ```

2. Change the ownership of the directory:

   ```bash
   [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`:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```bash
   $ ssh root@host1
   Password:
   ```
   b. Clear the SSSD cache:

   ```bash
   root@host1 ~]# sss_cache -E
   ```
   c. Restart the SSSD daemon:

   ```bash
   root@host1 ~]# systemctl restart sssd
   ```

Verification steps

1. SSH to `host1` as `idm_user`:

   ```bash
   [root@r8server ~]# ssh idm_user@host1.idm.example.com
   Password:
   Activate the web console with: systemctl enable --now cockpit.socket
   [idm_user@host1 /]$ 
   ```

2. Print the working directory:

   ```bash
   [idm_user@host1 /]$ pwd
   /home/user_1234/
   ```

Additional resources

- [Overriding Default Trust View attributes for an AD user on an IdM client with an ID view](#)

### 26.8. 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 the GID for an AD user, see [Overriding Default Trust View attributes for an AD user on an IdM client with an ID view](#).

**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`:

   ```bash
   [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:

   ```bash
   [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:

   ```bash
   [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
      -------------------------
      ```

   b. 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: ipaIDView, 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: ipaIDView, 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.

26.9. MIGRATING NIS DOMAINS TO IDENTITY MANAGEMENT

You can use ID views to set host specific UIDs and GIDs for existing hosts to prevent changing permissions for files and directories when migrating NIS domains into IdM.

Prerequisites

- You authenticated yourself as an admin using the kinit admin command.

Procedure

1. Add users and groups in the IdM domain.
   a. Create users using the ipa user-add command. For more information see: Adding users to IdM.
   b. Create groups using the ipa group-add command. For more information see: Adding groups to IdM.

2. Override IDs IdM generated during the user creation:
   a. Create a new ID view using ipa idview-add command. For more information see: Getting help for ID view commands.
   b. Add ID overrides for the users and groups to the ID view using ipa idoverrideuser-add and idoverridegroup-add respectively.

3. Assign the ID view to the specific hosts using ipa idview-apply command.

4. Decommission the NIS domains.

Verification

1. To check if all users and groups were added to the ID view correctly, use the ipa idview-show command.

   $ ipa idview-show example-view
   ID View Name: example-view
   User object overrides: example-user1
   Group object overrides: example-group
You can use ID views to specify new values for the POSIX attributes of your Active Directory (AD) users in an IdM-AD Trust environment.

By default, IdM applies the **Default Trust View** to all AD users. You can configure additional ID views on individual IdM clients to further adjust which POSIX attributes specific users receive.

### 27.1. HOW THE DEFAULT TRUST VIEW WORKS

The **Default Trust View** is the default ID view that is always applied to AD users and groups in trust-based setups. It is created automatically when you establish the trust using the `ipa-adtrust-install` command and cannot be deleted.

**NOTE**

The Default Trust View only accepts overrides for AD users and groups, not for IdM users and groups.

Using the Default Trust View, you can define custom POSIX attributes for AD users and groups, thus overriding the values defined in AD.

<table>
<thead>
<tr>
<th>Values in AD</th>
<th>Default Trust View</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>ad_user</td>
<td>ad_user</td>
</tr>
<tr>
<td>UID</td>
<td>111</td>
<td>222</td>
</tr>
<tr>
<td>GID</td>
<td>111</td>
<td>(no value)</td>
</tr>
</tbody>
</table>

You can also configure additional ID Views to override the Default Trust View on IdM clients. IdM applies the values from the host-specific ID view on top of the Default Trust View:

- If an attribute is defined in the host-specific ID view, IdM applies the value from this ID view.
- If an attribute is not defined in the host-specific ID view, IdM applies the value from the Default Trust View.

<table>
<thead>
<tr>
<th>Values in AD</th>
<th>Default Trust View</th>
<th>Host-specific ID view</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login</td>
<td>ad_user</td>
<td>(no value)</td>
<td>ad_user</td>
</tr>
<tr>
<td>UID</td>
<td>111</td>
<td>333</td>
<td>333</td>
</tr>
</tbody>
</table>
NOTE

You can only apply host-specific ID views to override the Default Trust View on IdM clients. IdM servers and replicas always apply the values from the Default Trust View.

Additional resources

- Using an ID view to override a user attribute value on an IdM client

27.2. DEFINING GLOBAL ATTRIBUTES FOR AN AD USER BY MODIFYING THE DEFAULT TRUST VIEW

If you want to override a POSIX attribute for an Active Directory (AD) user throughout your entire IdM deployment, modify the entry for that user in the Default Trust View. This procedure sets the GID for the AD user ad_user@ad.example.com to 732000006.

Prerequisites

- You are working with an IdM server that is a Trust Controller or Trust Agent.
- You have authenticated as an IdM administrator.

Procedure

1. As an IdM administrator, create an ID override for the AD user in the Default Trust View that changes the GID number to 732000006:

   ```
   # ipa idoverrideuser-add 'Default Trust View' ad_user@ad.example.com -- gidnumber=732000006
   ```

2. Clear the entry for the ad_user@ad.example.com user from the SSSD cache on all IdM servers and clients. This removes stale data and allows the new override value to apply.

   ```
   # ssscti cache-expire -u ad_user@ad.example.com
   ```

Verification

- Retrieve information for the ad_user@ad.example.com user to verify the GID reflects the updated value.

   ```
   # id ad_user@ad.example.com
   uid=702801456(ad_user@ad.example.com) gid=732000006(adadmins)
groups=732000006(adadmins),702800513(domain users@ad.example.com)
```
27.3. OVERRIDING DEFAULT TRUST VIEW ATTRIBUTES FOR AN AD USER ON AN IDM CLIENT WITH AN ID VIEW

You might want to override some POSIX attributes from the Default Trust View for an Active Directory (AD) user. For example, you might need to give an AD user a different GID on one particular IdM client. You can use an ID view to override a value from the Default Trust View for an AD user and apply it to a single host. This procedure explains how to set the GID for the ad_user@ad.example.com AD user on the host1.idm.example.com IdM client to 732001337.

Prerequisites

- You have root access to the host1.idm.example.com IdM client.
- You are logged in as a user with the required privileges, for example the admin user.

Procedure

1. Create an ID view. For example, to create an ID view named example_for_host1:

```bash
$ 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’s GID:

- Enter the `ipa idoverrideuser-add` command

```bash
$ ipa idoverrideuser-add example_for_host1 ad_user@ad.example.com --gidnumber=732001337
---------------------------
Added User ID override "ad_user@ad.example.com"
---------------------------
Anchor to override: ad_user@ad.example.com
GID: 732001337
```

3. Apply example_for_host1 to the host1.idm.example.com IdM client:

```bash
$ 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
---------------------------
```
4. Clear the entry for the `ad_user@ad.example.com` user from the SSSD cache on the `host1.idm.example.com` IdM client. This removes stale data and allows the new override value to apply.

```
[root@host1 ~]# sssctl cache-expire -u ad_user@ad.example.com
```

**Verification Steps**

1. **SSH to host1 as ad_user@ad.example.com:**

```
[root@r8server ~]# ssh ad_user@ad.example.com@host1.idm.example.com
```

2. Retrieve information for the `ad_user@ad.example.com` user to verify the GID reflects the updated value.

```
[ad_user@ad.example.com@host1 ~]$ id ad_user@ad.example.com
uid=702801456(ad_user@ad.example.com) gid=732001337(admins2)
groups=732001337(admins2),702800513(domain users@ad.example.com)
```

### 27.4. 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 the GID for an AD user, see Overriding Default Trust View attributes for an AD user on an IdM client with an ID view

**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
   ```
b. 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: ipaIDView, 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: ipaIDView, 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 28. 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.

28.1. 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, 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 Assigning 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 not assigned automatically when a replica is deleted and you must assign 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
- The range size: the number of ID’s in 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.

28.2. AUTOMATIC ID RANGES ASSIGNMENT

IdM ID ranges

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 Assigning DNA ID ranges manually. A DNA range matching the IdM ID range is automatically created during installation.

DNA ID ranges

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.

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.

28.3. 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
----------------------------

28.4. 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 must assign new DNA ID ranges to replicas manually as needed. For more information on how to do this, see [assigning DNA ID ranges manually](#).

**Procedure**

1. To create a new IdM ID range, use the `ipa idrange-add` command. You must 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
```

2. Optional: Update the ID range immediately:
   a. Clear the System Security Services Daemon (SSSD) cache:

```
# sss_cache -E
```

   b. Restart the SSSD daemon:

```
# systemctl restart sssd
```

**NOTE**

If you do not clear the SSSD cache and restart the service, SSSD only detects the new ID 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
----------------
```

28.5. THE ROLE OF SECURITY AND RELATIVE IDENTIFIERS IN IDM ID RANGES

An Identity Management (IdM) ID range is defined by several parameters:

- The range name
- The first POSIX ID of the range
- The range size: the number of IDs in the range
- The first relative identifier (RID) of the corresponding RID range
- The first RID of the secondary RID range

You can view these values by using the `ipa idrange-show` command:

```
$ ipa idrange-show IDM.EXAMPLE.COM_id_range
Range name: IDM.EXAMPLE.COM_id_range
First Posix ID of the range: 196600000
Number of IDs in the range: 200000
First RID of the corresponding RID range: 1000
First RID of the secondary RID range: 1000000
Range type: local domain range
```

Security identifiers

The data from the ID ranges of the local domain are used by the IdM server internally to assign unique security identifiers (SIDs) to IdM users and groups. The SIDs are stored in the user and group objects. A user’s SID consists of the following:

- The domain SID
The user’s relative identifier (RID), which is a four-digit 32-bit value appended to the domain SID.

For example, if the domain SID is S-1-5-21-123-456-789 and the RID of a user from this domain is 1008, then the user has the SID of S-1-5-21-123-456-789-1008.

Relative identifiers

The RID itself is computed in the following way:

Subtract the first POSIX ID of the range from the user’s POSIX UID, and add the first RID of the corresponding RID range to the result. For example, if the UID of `idmuser` is 196600008, the first POSIX ID is 196600000, and the first RID is 1000, then `idmuser`’s RID is 1008.

NOTE

The algorithm computing the user’s RID checks if a given POSIX ID falls into the ID range allocated before it computes a corresponding RID. For example, if the first ID is 196600000 and the range size is 200000, then the POSIX ID of 1600000 is outside of the ID range and the algorithm does not compute a RID for it.

Secondary relative identifiers

In IdM, a POSIX UID can be identical to a POSIX GID. This means that if `idmuser` already exists with the UID of 196600008, you can still create a new `idmgroup` group with the GID of 196600008.

However, a SID can define only one object, a user or a group. The SID of S-1-5-21-123-456-789-1008 that has already been created for `idmuser` cannot be shared with `idmgroup`. An alternative SID must be generated for `idmgroup`.

IdM uses a secondary relative identifier, or secondary RID, to avoid conflicting SIDs. This secondary RID consists of the following:

- The secondary RID base
- A range size; by default identical with the base range size

In the example above, the secondary RID base is set to 1000000. To compute the RID for the newly created `idmgroup`: subtract the first POSIX ID of the range from the user’s POSIX UID, and add the first RID of the secondary RID range to the result. `idmgroup` is therefore assigned the RID of 1000008. Consequently, the SID of `idmgroup` is S-1-5-21-123-456-789-1000008.

IdM uses the secondary RID to compute a SID only if a user or a group object was previously created with a manually set POSIX ID. Otherwise, automatic assignment prevents assigning the same ID twice.

28.6. 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.
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

- See [Removing the trust using the command line](#).
- See [Removing the trust using the IdM Web UI](#).

### 28.7. 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:
  
  1. `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

## 28.8. MANUAL ID RANGE ASSIGNMENT

In certain situations, it is necessary to manually assign 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 stops functioning and must 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.

  To do this, 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 assign a DNA ID range to a replica using the commands in [Assigning 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.

28.9. ASSIGNING DNA ID RANGES MANUALLY

In some cases, you may need to manually assign Distributed Numeric Assignment (DNA) ID ranges to existing replicas, for example to reassign a DNA ID range assigned to a non-functioning replica. For more information, see Manual ID range assignment.

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 \texttt{ipa idrange-find} command. Otherwise, the command fails.

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

- \textit{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 \texttt{ipa-replica-manage dnarange-set}:

\begin{verbatim}
# ipa-replica-manage dnarange-set serverA.example.com 1250-1499
\end{verbatim}

- To define the next DNA ID range for a specified server, use \texttt{ipa-replica-manage dnanextrange-set}:

\begin{verbatim}
# ipa-replica-manage dnanextrange-set serverB.example.com 1500-5000
\end{verbatim}

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 29. MANAGING SUBID RANGES MANUALLY

In a containerized environment, sometimes an IdM user needs to assign subID ranges manually. The following instructions help you to manage the subID ranges.

29.1. GENERATING SUBID RANGES USING IDM CLI

You can generate a subID range and assign it to a user manually. Assuming, the username *jsmith* exists on an *ipa* server.

Prerequisites

- IdM user exists.
- Valid Kerberos ticket is obtained. See Logging in to IdM in the Web UI: Using a Kerberos ticket for more details.
- *root* privileges.

Procedure

1. Check for existing subID ranges:
   ```
   # ipa subid-find
   ```

2. In case the subID range does not exist, generate and assign the new subID range to a user by entering the following command:
   ```
   # ipa subid-generate --owner=jsmith
   ```
   Added subordinate id "359dfcef-6b76-4911-bd37-bb5b66b8c418"
   Unique ID: 359dfcef-6b76-4911-bd37-bb5b66b8c418
   Description: auto-assigned subid
   Owner: jsmith
   SubUID range start: 2147483648
   SubUID range size: 65536
   SubGID range start: 2147483648
   SubGID range size: 65536

3. Alternatively, generate and assign the new subID ranges to all users:
   ```
   # /usr/libexec/ipa/ipa-subids --all-users
   ```
   Found 2 user(s) without subordinate ids
   Processing user 'user4' (1/2)
   Processing user 'user5' (2/2)
   Updated 2 user(s)
   The ipa-subids command was successful

Note, that to assign subID ranges to the new IdM users by default, enable the following option:

```
# ipa config-mod --user-default-subid=True
```
To verify if the user has the subID range assigned, enter the following command:

```
# ipa subid-find --owner=jsmith
```

1 subordinate id matched

Unique ID: 359dfcef-6b76-4911-bd37-bb5b66b8c418
Owner: jsmith
SubUID range start: 2147483648
SubUID range size: 65536
SubGID range start: 2147483648
SubGID range size: 65536

Number of entries returned 1

29.2. GENERATING SUBID RANGES USING IDM WEBUI INTERFACE

You can generate a subID range and assign it to a user in the IdM WebUI interface.

Prerequisites

- An IdM user exists.
- Valid Kerberos ticket is obtained. See Logging in to IdM in the Web UI: Using a Kerberos ticket for more details.
- root privileges.

Procedure

1. In the IdM WebUI interface expand the Subordinate IDs tab and choose Subordinate IDs option.

2. When the Subordinate IDs interface appears, click the Add button in the top-right corner of the interface. The window “Add subid” appears.

3. In the window “Add subid” choose an owner, which is the user you want to assign a subID range.

4. Click the Add button.

Verification

1. Check the table under the Subordinate IDs tab. A new record should appear and the owner is the user to which you assign the subID range.

29.3. MANAGING EXISTING SUBID RANGES USING IDM CLI

You can search for subID ranges and display information about particular one if needed. Assuming, the username jsmith exists on an ipa server.

Prerequisites

- An IdM user exists.
Procedure

1. To display the details about subID range when you know a unique ID hash, enter the following command:

```bash
# ipa subid-show 359dfcef-6b76-4911-bd37-bb5b66b8c418
```

Unique ID: 359dfcef-6b76-4911-bd37-bb5b66b8c418
Owner: jsmith
SubUID range start: 2147483648
SubUID range size: 65536
SubGID range start: 2147483648
SubGID range size: 65536

2. To find the details for the subID range when you have a subID from that range, you can use the following command:

```bash
# ipa subid-match --subuid=2147483648
```

1 subordinate id matched

Unique ID: 359dfcef-6b76-4911-bd37-bb5b66b8c418
Owner: uid=jsmith
SubUID range start: 2147483648
SubUID range size: 65536
SubGID range start: 2147483648
SubGID range size: 65536

Number of entries returned 1

29.4. LISTING SUBID RANGES USING THE GETSUBID COMMAND

To list the subID ranges, for example, for the `user1` in IdM environment, follow the instruction below.

Prerequisites

- The `user1` exists in IdM.
- The `shadow-utils-subid` package is installed.

Procedure

1. Include `subid: sss` record into `/etc/nsswitch.conf` file.

Note that you can provide only one value for the `subid` field. Setting the `subid` field to the `sss` value tells the utils to use the subID ranges from the IdM settings. The `file` value or no value sets the utils to use the subID ranges from the `/etc/subuid` and `/etc/subgid` files.

2. List the subID range for a user:

```bash
# getsubids user1
0: user1 2147483648 65536
```
CHAPTER 30. 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.

30.1. 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.
30.2. 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 \texttt{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 \texttt{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, \texttt{ipa-client-install} adds DNS records for the host too. The command must be run on the client.

30.3. 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 \texttt{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 \texttt{ipa host-add} CLI command or the IdM Web UI is \textbf{Host Administrators}. The \textbf{Host Administrators} privilege can be obtained through the \textbf{IT Specialist} role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the \texttt{ipa-client-install} command. The level of credentials required for executing the \texttt{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 \textbf{Host Administrators} role. A full administrator is a member of the \textbf{admins} group. The \textbf{Host Administrators} role provides privileges to add hosts and enroll hosts. For details about this scenario, see \texttt{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 \texttt{ipa-client-install} successfully. The limited administrator in this case has the \textbf{Enrollment Administrator} role, which provides the \textbf{Host Enrollment} privilege. For details, \texttt{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.

### 30.4. 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 (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>
<thead>
<tr>
<th>Action</th>
<th>User</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-enrollment</td>
<td><code>$ ipa stageuser-add user_name [-password]</code></td>
<td><code>$ ipa host-add host_name [--random]</code></td>
</tr>
<tr>
<td>Activating the account</td>
<td><code>$ ipa stageuser-activate user_name</code></td>
<td><code>$ ipa-client install [--password]</code></td>
</tr>
<tr>
<td></td>
<td>(must be run on the host itself)</td>
<td></td>
</tr>
</tbody>
</table>

- **The deployment stage (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>
<thead>
<tr>
<th>Action</th>
<th>User</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>$ ipa stageuser-add user_name [-password]</code></td>
<td><code>$ ipa host-add host_name [--random]</code></td>
</tr>
<tr>
<td></td>
<td><code>$ ipa stageuser-activate user_name</code></td>
<td><code>$ ipa-client install [--password]</code></td>
</tr>
<tr>
<td></td>
<td>(must be run on the host itself)</td>
<td>(must be run on the host itself)</td>
</tr>
</tbody>
</table>
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.

**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.

**30.5. 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 30.3. Host operations part 1**

<table>
<thead>
<tr>
<th>Action</th>
<th>What are the prerequisites of the action?</th>
<th>When does it make sense to run the command?</th>
<th>How is the action performed by a system administrator? What command(s) does he run?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Enrolling a client

See [Preparing the system for Identity Management client installation](#). 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.

### 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 30.4. Host operations part 2

<table>
<thead>
<tr>
<th>Action</th>
<th>What are the prerequisites of the action?</th>
<th>When does it make sense to run the command?</th>
<th>How is the action performed by a system administrator? What command(s) does he run?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrolling a client</td>
<td>see <a href="#">Preparing the system for Identity Management client installation</a></td>
<td>When you want the host to join the IdM realm.</td>
<td>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 <code>ipa host-add</code> command is run, and then a keytab is created to provision the client. Both parts are performed automatically by the <code>ipa-client-install</code> 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.</td>
</tr>
<tr>
<td>Disabling a client</td>
<td>The host must have an entry in IdM. The host needs to have an active keytab.</td>
<td>When you want to remove the host from the IdM realm temporarily, perhaps for maintenance purposes.</td>
<td><code>ipa host-disable host_name</code></td>
</tr>
<tr>
<td>Enabling a client</td>
<td>The host must have an entry in IdM.</td>
<td>When you want the temporarily disabled host to become active again.</td>
<td><code>ipa-getkeytab</code></td>
</tr>
<tr>
<td>Re-enrolling a client</td>
<td>The host must have an entry in IdM.</td>
<td>When the original host has been lost but you have installed a host with the same host name.</td>
<td><code>ipa-client-install --keytab</code> or <code>ipa-client-install --force-join</code></td>
</tr>
<tr>
<td>Un-enrolling a client</td>
<td>The host must have an entry in IdM.</td>
<td>When you want to remove the host from the IdM realm permanently.</td>
<td><code>ipa-client-install --uninstall</code></td>
</tr>
<tr>
<td>Action</td>
<td>On which machine can the administrator run the command(s)?</td>
<td>What happens when the action is performed? What are the consequences for the host’s functioning in IdM? What limitations are introduced/removed?</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Enrolling a client</strong></td>
<td>In the case of a two-step enrollment: <code>ipa host-add</code> can be run on any IdM client; the second step of <code>ipa-client-install</code> must be run on the client itself</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td><strong>Disabling a client</strong></td>
<td>Any machine in IdM, even the host itself</td>
<td>The host’s Kerberos key and SSL certificate are invalidated, and all services running on the host are disabled.</td>
<td></td>
</tr>
<tr>
<td><strong>Enabling a client</strong></td>
<td>Any machine in IdM. If run on the disabled host, LDAP credentials need to be supplied.</td>
<td>The host’s Kerberos key and the SSL certificate are made valid again, and all IdM services running on the host are re-enabled.</td>
<td></td>
</tr>
<tr>
<td><strong>Re-enrolling a client</strong></td>
<td>The host to be re-enrolled. LDAP credentials need to be supplied.</td>
<td>A new Kerberos key is generated for the host, replacing the previous one.</td>
<td></td>
</tr>
<tr>
<td><strong>Un-enrolling a client</strong></td>
<td>The host to be un-enrolled.</td>
<td>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 <code>/etc/krb5.keytab (host/&lt;fqdn&gt;@REALM)</code> 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 (<code>ipa host-disable &lt;fqdn&gt;</code>).</td>
<td></td>
</tr>
</tbody>
</table>

### 30.6. 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.

**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 30.5. Host Configuration Properties**

<table>
<thead>
<tr>
<th>UI Field</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>--desc=description</td>
<td>A description of the host.</td>
</tr>
<tr>
<td>Locality</td>
<td>--locality=locality</td>
<td>The geographic location of the host.</td>
</tr>
<tr>
<td>Location</td>
<td>--location=location</td>
<td>The physical location of the host, such as its data center rack.</td>
</tr>
<tr>
<td>Platform</td>
<td>--platform=string</td>
<td>The host hardware or architecture.</td>
</tr>
<tr>
<td>Operating system</td>
<td>--os=string</td>
<td>The operating system and version for the host.</td>
</tr>
<tr>
<td>MAC address</td>
<td>--macaddress=address</td>
<td>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.</td>
</tr>
<tr>
<td>SSH public keys</td>
<td>--sshpubkey=string</td>
<td>The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.</td>
</tr>
</tbody>
</table>
### UI Field | Command-Line Option | Description
--- | --- | ---
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.

---

### 30.7. 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 commands 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 30.1. 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 30.2. 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.

30.8. 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

30.9. RE-ENROLLING AN IDENTITY MANAGEMENT CLIENT

This section describes the different way you can re-enroll an Identity Management client.

30.9.1. 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 Renaming Identity
Management client systems.

What happens during client re-enrollment
During re-enrollment, Identity Management:

- Revokes the original host certificate
- Creates new SSH keys
- Generates a new keytab

30.9.2. 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:
   ```bash
   # 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:
   ```bash
   User authorized to enroll computers: hostadmin
   Password for hostadmin@EXAMPLE.COM:
   ```

Additional resources

- See Installing a client by using user credentials: Interactive installation in Installing Identity Management.

30.9.3. 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:
   ```bash
   # ipa-client-install --keytab /etc/krb5.keytab
   ```
30.9.4. 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 ~]$
```

30.10. 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 [Uninstalling an Identity Management client](#).
3. Renaming the host. For details, see [Renaming the host system](#).
4. Installing the IdM client on the host with the new name. For details, see [Installing an Identity Management client: Basic scenario](#) in *Installing Identity Management*.
5. Configuring the host after the IdM client installation. For details, see [Re-adding services, re-generating certificates, and re-adding host groups](#).

### 30.10.1. 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
```

30.10.2. 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:

```sh
[root@server ~]# ipa host-del client.example.com
```

### 30.10.3. Renaming the host system

Rename the machine as required. For example:

```sh
[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.

### 30.10.4. Re-adding services, re-generating certificates, and re-adding host groups

On the Identity Management (IdM) server, add a new keytab for every service identified in the Preparing an IdM client for its renaming.

```sh
[root@server ~]# ipa service-add service_name/new-client-name
```

1. 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

2. Re-add the client to the host groups identified in the Preparing an IdM client for its renaming.

### 30.11. DISABLING AND RE-ENABLING HOST ENTRIES

This section describes how to disable and re-enable hosts in Identity Management (IdM).

#### 30.11.1. 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.

**Procedure**

- Disable a host using the `host-disable` command. Disabling a host kills the host’s current, active keytabs. For example:
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.

### 30.11.2. 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.

**Procedure**

- 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:

```bash
$ 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 31. 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.

31.1. 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.

31.2. 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.

### 31.3. 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.

### 31.4. 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 (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 31.1. User and host enrollment

<table>
<thead>
<tr>
<th>Action</th>
<th>User</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-enrollment</td>
<td>$ ipa stageuser-add <code>user_name</code> [- -password]</td>
<td>$ ipa host-add <code>host_name</code> [--random]</td>
</tr>
<tr>
<td>Activating the account</td>
<td>$ ipa stageuser-activate <code>user_name</code></td>
<td>$ ipa-client-install [--password] (must be run on the host itself)</td>
</tr>
</tbody>
</table>

• The deployment stage (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 31.2. User and host session authentication

<table>
<thead>
<tr>
<th>User</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default means of authentication</td>
<td>Password</td>
</tr>
<tr>
<td>Starting a session (ordinary user)</td>
<td>$ kinit <code>user_name</code></td>
</tr>
<tr>
<td>The result of successful authentication</td>
<td>TGT to be used to obtain access to specific services</td>
</tr>
</tbody>
</table>

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.
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.

31.5. 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.

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>
<thead>
<tr>
<th>UI Field</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>--desc=description</td>
<td>A description of the host.</td>
</tr>
<tr>
<td>Locality</td>
<td>--locality=locality</td>
<td>The geographic location of the host.</td>
</tr>
<tr>
<td>Location</td>
<td>--location=location</td>
<td>The physical location of the host, such as its data center rack.</td>
</tr>
<tr>
<td>UI Field</td>
<td>Command-Line Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Platform</td>
<td>--platform=string</td>
<td>The host hardware or architecture.</td>
</tr>
<tr>
<td>Operating system</td>
<td>--os=string</td>
<td>The operating system and version for the host.</td>
</tr>
<tr>
<td>MAC address</td>
<td>--macaddress=address</td>
<td>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.</td>
</tr>
<tr>
<td>SSH public keys</td>
<td>--sshpubkey=string</td>
<td>The full SSH public key for the host. This is a multi-valued attribute, so multiple keys can be set.</td>
</tr>
<tr>
<td>Principal name (not editable)</td>
<td>--principalname=principal</td>
<td>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.</td>
</tr>
<tr>
<td>Set One-Time Password</td>
<td>--password=string</td>
<td>This option sets a password for the host which can be used in bulk enrollment.</td>
</tr>
<tr>
<td>-</td>
<td>--random</td>
<td>This option generates a random password to be used in bulk enrollment.</td>
</tr>
<tr>
<td>-</td>
<td>--certificate=string</td>
<td>A certificate blob for the host.</td>
</tr>
<tr>
<td>-</td>
<td>--updatedns</td>
<td>This sets whether the host can dynamically update its DNS entries if its IP address changes.</td>
</tr>
</tbody>
</table>

31.6. 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 31.1. Adding Host 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 31.2. Add Host Wizard

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.
Host: server.zone.example.com

Host Settings

- **Host name**: server.zone.example.com
- **Principal name**: host/server.zone.example.com@EXAMPLE.COM
- **Description**: 
- **Class**: 
- **Locality**: 

Figure 31.3. Expanded Entry Page
CHAPTER 32. 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:

- 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

32.1. 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.
- 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:
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:

```yaml
---
- 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:
```

2. 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.
32.2. 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.
- 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 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:
         ipadmin_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:
   ```
3. Run the playbook:

```bash
$ 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:

```bash
$ ssh admin@server.idm.example.com
Password:
```

2. Enter the `ipa host-show` command and specify the name of the host:

```bash
$ 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.

**32.3. 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.
- 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 **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
   ```

3. 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]
   ```
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 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.

### 32.4. 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.
- 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. 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:

```yaml
---
- 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-addresses-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.
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.

### 32.5. 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.

**32.6. ADDITIONAL RESOURCES**

- See the `/usr/share/doc/ansible-freeipa/README-host.md` Markdown file.

- See the additional playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/host` directory.
CHAPTER 33. 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

33.1. 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.

33.2. 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
-------------------
```

### 33.3. 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
   ---------------------
   ```

### 33.4. 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.
   ```

33.5. 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
     ```
You can add multiple hosts and multiple host groups to an IdM host group in one single command using the following syntax:

```bash
$ 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.

### 33.6. 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`:

     ```bash
     $ 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`:

     ```bash
     $ ipa hostgroup-remove-member group_name --hostgroups nested_group
     ```
### 33.7. 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` command. 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: projectadmins
   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`:
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.

Additional resources

- See `ipa hostgroup-add-member-manager --help` for more details.
- See `ipa hostgroup-show --help` for more details.

### 33.8. 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**

1. Optional. Use the `ipa hostgroup-find` command to find hosts and host groups.
2. 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
   ----------------------------------
   ```

3. 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 34. 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

34.1. 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.

34.2. 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

1. 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.

2. 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.

3. 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.

34.3. 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.

### 34.4. 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.

### 34.5. 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.

34.6. 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.
34.7. 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

1. Click **Identity → Groups** and select the **Host Groups** tab.

2. Click the name of the group to which you want to add member managers.

3. Click the member managers tab **User Groups** or **Users** depending on the type of member managers you want to add. The corresponding dialog appears.

4. Click **Add**.

6. Click Delete to confirm. The selected members are deleted.
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.

### 34.8. 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

1. Click Identity → Groups and select the Host Groups tab.

2. Click the name of the group from which you want to remove member managers.

3. Click the member managers tab User Groups or Users depending on the type of member managers you want to remove. The corresponding dialog appears.

4. Select the user or user groups to remove and click Delete.

5. Click Delete to confirm.

   Remove groups from member managers for host group ‘test_hostgroup’

   Are you sure you want to delete selected entries?

   ● testgroup

   

   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.
Host Group: test_hostgroup

- Host Group: test_hostgroup
- test_hostgroup is a member of:
- test_hostgroup member managers:

No entries.
CHAPTER 35. 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

35.1. 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.

35.2. 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:

   ```bash
   [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.

   ```yaml
   ---
   - 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:

   ```bash
   $ 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:

   ```bash
   $ 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.

### 35.3. 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.

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 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.

35.4. 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 host group `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:

```yaml
---
- 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.

### 35.5. 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.

35.6. 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:

```yaml
---
- 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.
35.7. 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.

### 35.8. 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.

### 35.9. 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:

   ```yaml
   ---
   - 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 36. 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).

36.1. 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.

36.2. 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

- You have installed the `ansible-freeipa` package 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:

```yaml
---
- 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
        hbacsvc:
        - 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

- See the README-hbacsvc.md, README-hbacsvcgroup.md, and README-hbacrule.md files in the `/usr/share/doc/ansible-freeipa` directory.

- See the playbooks in the subdirectories of the `/usr/share/doc/ansible-freeipa/playbooks` directory.
CHAPTER 37. 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

37.1. 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.
37.2. 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)
```

37.3. 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.

   ```bash
   [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.

   ```bash
   [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.

   ```bash
   [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.
37.4. 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.

```plaintext
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.

```plaintext
[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.

```plaintext
[root@client2 ~]# id <user_from_subdomain1>
uid=1916901106(user_from_subdomain1) gid=1916900513(domain users) groups=1916900513(domain users)
```

37.5. ADDITIONAL RESOURCES
• Using an ID view to override a user attribute value on an IdM client
CHAPTER 38. ENABLING AUTHENTICATION USING AD USER PRINCIPAL NAMES IN IDM

38.1. 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

38.2. 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.

### 38.3. 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**

1. Open the `/usr/share/ipa/smb.conf.empty` configuration file in a text editor.

2. Add the following contents to the file.

   ```
   [global]
   log level = 10
   ```

3. 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

- See [Using rpcclient to gather troubleshooting data for AD UPN authentication issues](#).
CHAPTER 39. ENABLING AD USERS TO ADMINISTER IDM

39.1. ID OVERRIDES FOR AD USERS

You can centrally manage access of Active Directory (AD) users and groups to Identity Management (IdM) resources in a POSIX environment by 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. With this feature, the IdM LDAP server is able to apply access control rules for the IdM group to the AD user.

AD users can 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.

Additional resources
- Using ID views for Active Directory users

39.2. USING ID OVERRIDES TO ENABLE AD USERS TO ADMINISTER IDM

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.

Prerequisites
- 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

1. 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
   ```

2. 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
   ```
39.3. 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.

1. 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.

2. 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:

3. 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