Red Hat Enterprise Linux 9.0 Beta Configuring and managing Identity Management

Configuring, managing, and maintaining Identity Management in Red Hat Enterprise Linux 9
Abstract

This documentation collection provides instructions on how to effectively configure, manage and maintain Identity Management on Red Hat Enterprise Linux 9.
# Table of Contents

**RHEL BETA RELEASE** ........................................................................................................... 19

**MAKING OPEN SOURCE MORE INCLUSIVE** ................................................................. 20

**PROVIDING FEEDBACK ON RED HAT DOCUMENTATION** ........................................... 21

**CHAPTER 1. LOGGING IN TO IDENTITY MANAGEMENT FROM THE COMMAND LINE** .......... 22

1.1. USING KINIT TO LOG IN TO IDM MANUALLY .......................................................... 22

1.2. DESTROYING A USER'S ACTIVE KERBEROS TICKET .............................................. 23

1.3. CONFIGURING AN EXTERNAL SYSTEM FOR KERBEROS AUTHENTICATION ........... 23

**CHAPTER 2. VIEWING, STARTING AND STOPPING THE IDENTITY MANAGEMENT SERVICES** .......... 25

2.1. THE IDM SERVICES ...................................................................................................... 25

2.1.1. List of services hosted by IdM servers ..................................................................... 25

2.1.2. List of services hosted by IdM clients ..................................................................... 27

2.2. VIEWING THE STATUS OF IDM SERVICES ............................................................... 28

2.3. STARTING AND STOPPING THE ENTIRE IDENTITY MANAGEMENT SERVER: THE IPACCTL UTILITY ........................................................................................................... 29

ipactl commands ................................................................................................................. 29

2.4. STARTING AND STOPPING AN INDIVIDUAL IDENTITY MANAGEMENT SERVICE: THE SYSTEMCTL UTILITY ........................................................................................................... 29

Useful systemctl commands .............................................................................................. 30

2.5. METHODS FOR DISPLAYING IDM SOFTWARE VERSION ........................................... 30

**CHAPTER 3. INTRODUCTION TO THE IDM COMMAND-LINE UTILITIES** ................................. 32

3.1. WHAT IS THE IPA COMMAND LINE INTERFACE ..................................................... 32

3.2. WHAT IS THE IPA HELP ............................................................................................. 32

3.3. USING IPA HELP TOPICS ......................................................................................... 33

3.4. USING IPA HELP COMMANDS ................................................................................ 33

3.5. STRUCTURE OF IPA COMMANDS .......................................................................... 34

3.6. USING AN IPA COMMAND TO ADD A USER ACCOUNT TO IDM ......................... 35

3.7. USING AN IPA COMMAND TO MODIFY A USER ACCOUNT IN IDM .................... 36

3.8. HOW TO SUPPLY A LIST OF VALUES TO THE IDM UTILITIES ............................. 37

3.9. HOW TO USE SPECIAL CHARACTERS WITH THE IDM UTILITIES ....................... 38

**CHAPTER 4. SEARCHING IDENTITY MANAGEMENT ENTRIES FROM THE COMMAND LINE** .......... 39

4.1. OVERVIEW OF LISTING IDM ENTRIES ..................................................................... 39

4.2. SHOWING DETAILS FOR A PARTICULAR ENTRY ..................................................... 39

4.3. ADJUSTING THE SEARCH SIZE AND TIME LIMIT ................................................. 40

4.3.1. Adjusting the search size and time limit in the command line ............................. 40

4.3.2. Adjusting the search size and time limit in the Web UI ...................................... 41

**CHAPTER 5. ACCESSING THE IDM WEB UI IN A WEB BROWSER** ........................................... 43

5.1. WHAT IS THE IDM WEB UI ....................................................................................... 43

5.2. WEB BROWSERS SUPPORTED FOR ACCESSING THE WEB UI ............................ 43

5.3. ACCESSING THE WEB UI ......................................................................................... 44

**CHAPTER 6. LOGGING IN TO IDM IN THE WEB UI: USING A KERBEROS TICKET** .................. 46

6.1. KERBEROS AUTHENTICATION IN IDENTITY MANAGEMENT ................................. 46

6.2. USING KINIT TO LOG IN TO IDM MANUALLY ....................................................... 46

6.3. CONFIGURING THE BROWSER FOR KERBEROS AUTHENTICATION .................... 47

6.4. LOGGING IN TO THE WEB UI USING A KERBEROS TICKET .................................. 48

6.5. CONFIGURING AN EXTERNAL SYSTEM FOR KERBEROS AUTHENTICATION ........ 49

6.6. WEB UI LOGIN FOR ACTIVE DIRECTORY USERS .................................................. 50
CHAPTER 7. LOGGING IN TO THE IDENTITY MANAGEMENT WEB UI USING ONE TIME PASSWORDS

7.1. PREREQUISITES
7.2. ONE TIME PASSWORD (OTP) AUTHENTICATION IN IDENTITY MANAGEMENT
7.3. ENABLING THE ONE TIME PASSWORD IN THE WEB UI
7.4. ADDING OTP TOKENS IN THE WEB UI
7.5. LOGGING INTO THE WEB UI WITH A ONE TIME PASSWORD
7.6. SYNCHRONIZING OTP TOKENS USING THE WEB UI
7.7. CHANGING EXPIRED PASSWORDS

CHAPTER 8. TROUBLESHOOTING AUTHENTICATION WITH SSSD IN IDM

8.1. DATA FLOW WHEN RETRIEVING IDM USER INFORMATION WITH SSSD
8.2. DATA FLOW WHEN RETRIEVING AD USER INFORMATION WITH SSSD
8.3. DATA FLOW WHEN AUTHENTICATING AS A USER WITH SSSD IN IDM
8.4. NARROWING THE SCOPE OF AUTHENTICATION ISSUES
8.5. SSSD LOG FILES AND LOGGING LEVELS
8.5.1. SSSD log file purposes
8.5.2. SSSD logging levels
8.6. ENABLING DETAILED LOGGING FOR SSSD IN THE SSSD.conf FILE
8.7. ENABLING DETAILED LOGGING FOR SSSD WITH THE SSSCTL COMMAND
8.8. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM SERVER
8.9. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM CLIENT
8.10. TRACKING CLIENT REQUESTS IN THE SSSD Backend

CHAPTER 9. CONFIGURING GLOBAL IDM SETTINGS USING ANSIBLE PLAYBOOKS

9.1. RETRIEVING IDM CONFIGURATION USING AN ANSIBLE PLAYBOOK
9.2. CONFIGURING THE IDM CA RENEWAL SERVER USING AN ANSIBLE PLAYBOOK
9.3. CONFIGURING THE DEFAULT SHELL FOR IDM USERS USING AN ANSIBLE PLAYBOOK

CHAPTER 10. MANAGING USER ACCOUNTS USING THE COMMAND LINE

10.1. USER LIFE CYCLE
10.2. ADDING USERS USING THE COMMAND LINE
10.3. ACTIVATING USERS USING THE COMMAND LINE
10.4. PRESERVING USERS USING THE COMMAND LINE
10.5. DELETING USERS USING THE COMMAND LINE
10.6. RESTORING USERS USING THE COMMAND LINE

CHAPTER 11. MANAGING USER ACCOUNTS USING THE IDM WEB UI

11.1. USER LIFE CYCLE
11.2. ADDING USERS IN THE WEB UI
11.3. ACTIVATING STAGE USERS IN THE IDM WEB UI
11.4. DISABLING USER ACCOUNTS IN THE WEB UI
11.5. ENABLING USER ACCOUNTS IN THE WEB UI
11.6. PRESERVING ACTIVE USERS IN THE IDM WEB UI
11.7. RESTORING USERS IN THE IDM WEB UI
11.8. DELETING USERS IN THE IDM WEB UI

CHAPTER 12. MANAGING USER ACCOUNTS USING ANSIBLE PLAYBOOKS

12.1. USER LIFE CYCLE
12.2. ENSURING THE PRESENCE OF AN IDM USER USING AN ANSIBLE PLAYBOOK
12.3. ENSURING THE PRESENCE OF MULTIPLE IDM USERS USING ANSIBLE PLAYBOOKS
12.4. ENSURING THE PRESENCE OF MULTIPLE IDM USERS FROM A JSON FILE USING ANSIBLE PLAYBOOKS
12.5. ENSURING THE ABSENCE OF USERS USING ANSIBLE PLAYBOOKS
CHAPTER 13. MANAGING USER GROUPS IN IDM CLI .................................................. 103
13.1. THE DIFFERENT GROUP TYPES IN IDM ......................................................... 103
13.2. DIRECT AND INDIRECT GROUP MEMBERS .................................................. 104
13.3. ADDING A USER GROUP USING IDM CLI ...................................................... 104
13.4. SEARCHING FOR USER GROUPS USING IDM CLI .......................................... 105
13.5. DELETING A USER GROUP USING IDM CLI .................................................... 105
13.6. ADDING A MEMBER TO A USER GROUP USING IDM CLI ................................. 106
13.7. ADDING USERS WITHOUT A USER PRIVATE GROUP ....................................... 107
13.7.1. Users without a user private group .............................................................. 107
13.7.2. Adding a user without a user private group when private groups are globally enabled 107
13.7.3. Disabling user private groups globally for all users ..................................... 108
13.7.4. Adding a user when user private groups are globally disabled ....................... 108
13.8. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE IDM CLI .......................................................... 109
13.9. VIEWING GROUP MEMBERS USING IDM CLI ............................................... 110
13.10. REMOVING A MEMBER FROM A USER GROUP USING IDM CLI ..................... 111
13.11. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE IDM CLI ......................................................... 111

CHAPTER 14. MANAGING USER GROUPS IN IDM WEB UI ........................................ 113
14.1. THE DIFFERENT GROUP TYPES IN IDM ......................................................... 113
14.2. DIRECT AND INDIRECT GROUP MEMBERS .................................................. 114
14.3. ADDING A USER GROUP USING IDM WEB UI ................................................. 114
14.4. DELETING A USER GROUP USING IDM WEB UI ............................................. 115
14.5. ADDING A MEMBER TO A USER GROUP USING IDM WEB UI ........................... 116
14.6. ADDING USERS OR GROUPS AS MEMBER MANAGERS TO AN IDM USER GROUP USING THE WEB UI .......................................................... 117
14.7. VIEWING GROUP MEMBERS USING IDM WEB UI .......................................... 119
14.8. REMOVING A MEMBER FROM A USER GROUP USING IDM WEB UI ................. 119
14.9. REMOVING USERS OR GROUPS AS MEMBER MANAGERS FROM AN IDM USER GROUP USING THE WEB UI ......................................................... 120

CHAPTER 15. MANAGING USER GROUPS USING ANSIBLE PLAYBOOKS ............................ 122
15.1. THE DIFFERENT GROUP TYPES IN IDM ......................................................... 122
15.2. DIRECT AND INDIRECT GROUP MEMBERS .................................................. 123
15.3. ENSURING THE PRESENCE OF IDM GROUPS AND GROUP MEMBERS USING ANSIBLE PLAYBOOKS .......................................................... 124
15.4. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS ..................................................... 125
15.5. ENSURING THE ABSENCE OF MEMBER MANAGERS IN IDM USER GROUPS USING ANSIBLE PLAYBOOKS ..................................................... 126

CHAPTER 16. AUTOMATING GROUP MEMBERSHIP USING IDM CLI ................................. 129
16.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP ........................................... 129
16.2. AUTOMEMBER RULES ..................................................................................... 129
16.3. ADDING AN AUTOMEMBER RULE USING IDM CLI ........................................... 130
16.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM CLI ................. 131
16.5. VIEWING EXISTING AUTOMEMBER RULES USING IDM CLI ............................... 132
16.6. DELETING AN AUTOMEMBER RULE USING IDM CLI ....................................... 133
16.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM CLI ........ 133
16.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM CLI .......... 134
16.9. CONFIGURING A DEFAULT AUTOMEMBER GROUP USING IDM CLI .................... 135

CHAPTER 17. AUTOMATING GROUP MEMBERSHIP USING IDM WEB UI ............................ 137
17.1. BENEFITS OF AUTOMATIC GROUP MEMBERSHIP ........................................... 137
17.2. AUTOMEMBER RULES
17.3. ADDING AN AUTOMEMBER RULE USING IDM WEB UI
17.4. ADDING A CONDITION TO AN AUTOMEMBER RULE USING IDM WEB UI
17.5. VIEWING EXISTING AUTOMEMBER RULES AND CONDITIONS USING IDM WEB UI
17.6. DELETING AN AUTOMEMBER RULE USING IDM WEB UI
17.7. REMOVING A CONDITION FROM AN AUTOMEMBER RULE USING IDM WEB UI
17.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM WEB UI
17.8.1. Rebuilding automatic membership for all users or hosts
17.8.2. Rebuilding automatic membership for a single user or host only
17.9. CONFIGURING A DEFAULT USER GROUP USING IDM WEB UI
17.10. CONFIGURING A DEFAULT HOST GROUP USING IDM WEB UI

CHAPTER 18. MANAGING SELF-SERVICE RULES IN IDM USING THE CLI ............................................. 147
18.1. SELF-SERVICE ACCESS CONTROL IN IDM
18.2. CREATING SELF-SERVICE RULES USING THE CLI
18.3. EDITING SELF-SERVICE RULES USING THE CLI
18.4. DELETING SELF-SERVICE RULES USING THE CLI

CHAPTER 19. MANAGING SELF-SERVICE RULES USING THE IDM WEB UI ........................................... 150
19.1. SELF-SERVICE ACCESS CONTROL IN IDM
19.2. CREATING SELF-SERVICE RULES USING THE IDM WEB UI
19.3. EDITING SELF-SERVICE RULES USING THE IDM WEB UI
19.4. DELETING SELF-SERVICE RULES USING THE IDM WEB UI

CHAPTER 20. USING ANSIBLE PLAYBOOKS TO MANAGE SELF-SERVICE RULES IN IDM .......... 154
20.1. SELF-SERVICE ACCESS CONTROL IN IDM
20.2. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS PRESENT
20.3. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE IS ABSENT
20.4. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE HAS SPECIFIC ATTRIBUTES
20.5. USING ANSIBLE TO ENSURE THAT A SELF-SERVICE RULE DOES NOT HAVE SPECIFIC ATTRIBUTES

CHAPTER 21. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM CLI .... 161
21.1. DELEGATION RULES
21.2. CREATING A DELEGATION RULE USING IDM CLI
21.3. VIEWING EXISTING DELEGATION RULES USING IDM CLI
21.4. MODIFYING A DELEGATION RULE USING IDM CLI
21.5. DELETING A DELEGATION RULE USING IDM CLI

CHAPTER 22. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM WEBUI 164
22.1. DELEGATION RULES
22.2. CREATING A DELEGATION RULE USING IDM WEBUI
22.3. VIEWING EXISTING DELEGATION RULES USING IDM WEBUI
22.4. MODIFYING A DELEGATION RULE USING IDM WEBUI
22.5. DELETING A DELEGATION RULE USING IDM WEBUI

CHAPTER 23. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING ANSIBLE PLAYBOOKS ................................................................. 170
23.1. DELEGATION RULES
23.2. CREATING AN ANSIBLE INVENTORY FILE FOR IDM
23.3. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS PRESENT
23.4. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS ABSENT
23.5. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE HAS SPECIFIC ATTRIBUTES
23.6. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE DOES NOT HAVE SPECIFIC ATTRIBUTES
## Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.1</td>
<td>ROLE-BASED ACCESS CONTROL IN IDM</td>
<td>178</td>
</tr>
<tr>
<td>24.1.1</td>
<td>Permissions in IdM</td>
<td>178</td>
</tr>
<tr>
<td>24.1.2</td>
<td>Default managed permissions</td>
<td>179</td>
</tr>
<tr>
<td>24.1.3</td>
<td>Privileges in IdM</td>
<td>180</td>
</tr>
<tr>
<td>24.1.4</td>
<td>Roles in IdM</td>
<td>181</td>
</tr>
<tr>
<td>24.1.5</td>
<td>Predefined roles in Identity Management</td>
<td>181</td>
</tr>
<tr>
<td>24.2</td>
<td>MANAGING IDM PERMISSIONS IN THE CLI</td>
<td>182</td>
</tr>
<tr>
<td>24.3</td>
<td>COMMAND OPTIONS FOR EXISTING PERMISSIONS</td>
<td>184</td>
</tr>
<tr>
<td>24.4</td>
<td>MANAGING IDM PRIVILEGES IN THE CLI</td>
<td>184</td>
</tr>
<tr>
<td>24.5</td>
<td>COMMAND OPTIONS FOR EXISTING PRIVILEGES</td>
<td>185</td>
</tr>
<tr>
<td>24.6</td>
<td>MANAGING IDM ROLES IN THE CLI</td>
<td>185</td>
</tr>
<tr>
<td>24.7</td>
<td>COMMAND OPTIONS FOR EXISTING ROLES</td>
<td>186</td>
</tr>
<tr>
<td>25.1</td>
<td>ROLE-BASED ACCESS CONTROL IN IDM</td>
<td>187</td>
</tr>
<tr>
<td>25.1.1</td>
<td>Permissions in IdM</td>
<td>187</td>
</tr>
<tr>
<td>25.1.2</td>
<td>Default managed permissions</td>
<td>188</td>
</tr>
<tr>
<td>25.1.3</td>
<td>Privileges in IdM</td>
<td>189</td>
</tr>
<tr>
<td>25.1.4</td>
<td>Roles in IdM</td>
<td>190</td>
</tr>
<tr>
<td>25.1.5</td>
<td>Predefined roles in Identity Management</td>
<td>190</td>
</tr>
<tr>
<td>25.2</td>
<td>MANAGING PERMISSIONS IN THE IDM WEB UI</td>
<td>191</td>
</tr>
<tr>
<td>25.3</td>
<td>MANAGING PRIVILEGES IN THE IDM WEB UI</td>
<td>195</td>
</tr>
<tr>
<td>25.4</td>
<td>MANAGING ROLES IN THE IDM WEB UI</td>
<td>198</td>
</tr>
<tr>
<td>26.</td>
<td>PREPARING YOUR ENVIRONMENT FOR MANAGING IDM USING ANSIBLE PLAYBOOKS</td>
<td>203</td>
</tr>
<tr>
<td>27.</td>
<td>USING ANSIBLE PLAYBOOKS TO MANAGE ROLE-BASED ACCESS CONTROL IN IDM</td>
<td>205</td>
</tr>
<tr>
<td>27.1</td>
<td>PERMISSIONS IN IDM</td>
<td>205</td>
</tr>
<tr>
<td>27.2</td>
<td>DEFAULT MANAGED PERMISSIONS</td>
<td>206</td>
</tr>
<tr>
<td>27.3</td>
<td>PRIVILEGES IN IDM</td>
<td>208</td>
</tr>
<tr>
<td>27.4</td>
<td>ROLES IN IDM</td>
<td>208</td>
</tr>
<tr>
<td>27.5</td>
<td>PREDEFINED ROLES IN IDENTITY MANAGEMENT</td>
<td>208</td>
</tr>
<tr>
<td>27.6</td>
<td>USING ANSIBLE TO ENSURE AN IDM RBAC ROLE WITH PRIVILEGES IS PRESENT</td>
<td>209</td>
</tr>
<tr>
<td>27.7</td>
<td>USING ANSIBLE TO ENSURE AN IDM RBAC ROLE IS ABSENT</td>
<td>211</td>
</tr>
<tr>
<td>27.8</td>
<td>USING ANSIBLE TO ENSURE THAT A GROUP OF USERS IS ASSIGNED TO AN IDM RBAC ROLE</td>
<td>212</td>
</tr>
<tr>
<td>27.9</td>
<td>USING ANSIBLE TO ENSURE THAT SPECIFIC USERS ARE NOT ASSIGNED TO AN IDM RBAC ROLE</td>
<td>214</td>
</tr>
<tr>
<td>27.10</td>
<td>USING ANSIBLE TO ENSURE A SERVICE IS A MEMBER OF AN IDM RBAC ROLE</td>
<td>215</td>
</tr>
<tr>
<td>27.11</td>
<td>USING ANSIBLE TO ENSURE A HOST IS A MEMBER OF AN IDM RBAC ROLE</td>
<td>217</td>
</tr>
<tr>
<td>27.12</td>
<td>USING ANSIBLE TO ENSURE A HOST GROUP IS A MEMBER OF AN IDM RBAC ROLE</td>
<td>218</td>
</tr>
<tr>
<td>28.</td>
<td>USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PRIVILEGES</td>
<td>221</td>
</tr>
<tr>
<td>28.1</td>
<td>USING ANSIBLE TO ENSURE A CUSTOM IDM RBAC PRIVILEGE IS PRESENT</td>
<td>221</td>
</tr>
<tr>
<td>28.2</td>
<td>USING ANSIBLE TO ENSURE MEMBER PERMISSIONS ARE PRESENT IN A CUSTOM IDM RBAC PRIVILEGE</td>
<td>222</td>
</tr>
<tr>
<td>28.3</td>
<td>USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE DOES NOT INCLUDE A PERMISSION</td>
<td>224</td>
</tr>
<tr>
<td>28.4</td>
<td>USING ANSIBLE TO RENAME A CUSTOM IDM RBAC PRIVILEGE</td>
<td>225</td>
</tr>
<tr>
<td>28.5</td>
<td>USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE IS ABSENT</td>
<td>227</td>
</tr>
<tr>
<td>29.</td>
<td>USING ANSIBLE PLAYBOOKS TO MANAGE RBAC PERMISSIONS IN IDM</td>
<td>230</td>
</tr>
<tr>
<td>29.1</td>
<td>USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS PRESENT</td>
<td>230</td>
</tr>
<tr>
<td>29.2</td>
<td>USING ANSIBLE TO ENSURE AN RBAC PERMISSION WITH AN ATTRIBUTE IS PRESENT</td>
<td>232</td>
</tr>
<tr>
<td>29.3</td>
<td>USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS ABSENT</td>
<td>233</td>
</tr>
<tr>
<td>29.4</td>
<td>USING ANSIBLE TO ENSURE AN ATTRIBUTE IS A MEMBER OF AN IDM RBAC PERMISSION</td>
<td>235</td>
</tr>
</tbody>
</table>
CHAPTER 39. MANAGING HOST GROUPS USING THE IDM CLI ................................. 319
39.1. HOST GROUPS IN IDM 319
39.2. VIEWING IDM HOST GROUPS USING THE CLI 319
39.3. CREATING IDM HOST GROUPS USING THE CLI 320
39.4. DELETING IDM HOST GROUPS USING THE CLI 320
39.5. ADDING IDM HOST GROUP MEMBERS USING THE CLI 321
39.6. REMOVING IDM HOST GROUP MEMBERS USING THE CLI 322
39.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE CLI 323
39.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE CLI 324

CHAPTER 40. MANAGING HOST GROUPS USING THE IDM WEB UI ............................ 326
40.1. HOST GROUPS IN IDM 326
40.2. VIEWING HOST GROUPS IN THE IDM WEB UI 326
40.3. CREATING HOST GROUPS IN THE IDM WEB UI 327
40.4. DELETING HOST GROUPS IN THE IDM WEB UI 328
40.5. ADDING HOST GROUP MEMBERS IN THE IDM WEB UI 328
40.6. REMOVING HOST GROUP MEMBERS IN THE IDM WEB UI 329
40.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI 330
40.8. REMOVING IDM HOST GROUP MEMBER MANAGERS USING THE WEB UI 331

CHAPTER 41. MANAGING HOST GROUPS USING ANSIBLE PLAYBOOKS .......................... 334
41.1. HOST GROUPS IN IDM 334
41.2. ENSURING THE PRESENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 334
41.3. ENSURING THE PRESENCE OF HOSTS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 336
41.4. NESTING IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 337
41.5. ENSURING THE PRESENCE OF MEMBER MANAGERS IN IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 339
41.6. ENSURING THE ABSENCE OF HOSTS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 340
41.7. ENSURING THE ABSENCE OF NESTED HOST GROUPS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 342
41.8. ENSURING THE ABSENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 343
41.9. ENSURING THE ABSENCE OF MEMBER MANAGERS FROM IDM HOST GROUPS USING ANSIBLE PLAYBOOKS 344

CHAPTER 42. MANAGING KERBEROS TICKET POLICIES ...................................... 347
42.1. THE ROLE OF THE IDM KDC 347
42.2. IDM KERBEROS TICKET POLICY TYPES 348
42.3. KERBEROS AUTHENTICATION INDICATORS 349
    42.3.1. Authentication indicators and IdM services 349
42.4. ENFORCING AUTHENTICATION INDICATORS FOR AN IDM SERVICE 350
    42.4.1. Creating an IdM service entry and its Kerberos keytab 350
    42.4.2. Associating authentication indicators with an IdM service 351
    42.4.3. Retrieving a Kerberos service ticket for an IdM service 353
    42.4.4. Additional resources 353
42.5. CONFIGURING THE GLOBAL TICKET LIFECYCLE POLICY 354
42.6. CONFIGURING GLOBAL TICKET POLICIES PER AUTHENTICATION INDICATOR 354
42.7. CONFIGURING THE DEFAULT TICKET POLICY FOR A USER 355
42.8. CONFIGURING INDIVIDUAL AUTHENTICATION INDICATOR TICKET POLICIES FOR A USER 356
42.9. AUTHENTICATION INDICATOR OPTIONS FOR THE KRBTPOLICY-MOD COMMAND 357

CHAPTER 43. DEFINING IDM PASSWORD POLICIES ............................................ 358
43.1. WHAT IS A PASSWORD POLICY 358
43.2. PASSWORD POLICIES IN IDM 358
43.3. ENSURING THE PRESENCE OF A PASSWORD POLICY IN IDM USING AN ANSIBLE PLAYBOOK 360
48.3. THE PROS AND CONS OF USING CERTIFICATES TO AUTHENTICATE USERS IN IDM

CHAPTER 49. MANAGING CERTIFICATES FOR USERS, HOSTS, AND SERVICES USING THE INTEGRATED IDM CA

49.1. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE USING IDM WEB UI
49.2. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE FROM IDM CA USING CERTUTIL
49.3. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE FROM IDM CA USING OPENSSL
49.4. ADDITIONAL RESOURCES

CHAPTER 50. CONVERTING CERTIFICATE FORMATS TO WORK WITH IDM

50.1. CERTIFICATE FORMATS AND ENCODINGS IN IDM
50.2. CONVERTING AN EXTERNAL CERTIFICATE TO LOAD INTO AN IDM USER ACCOUNT
50.3. PREPARING TO LOAD A CERTIFICATE INTO THE BROWSER
50.4. CERTIFICATE-RELATED COMMANDS AND FORMATS IN IDM

CHAPTER 51. CREATING AND MANAGING CERTIFICATE PROFILES IN IDENTITY MANAGEMENT

51.1. WHAT IS A CERTIFICATE PROFILE?
51.2. CREATING A CERTIFICATE PROFILE
51.3. WHAT IS A CA ACCESS CONTROL LIST?
51.4. DEFINING A CA ACL TO CONTROL ACCESS TO CERTIFICATE PROFILES
51.5. USING CERTIFICATE PROFILES AND CA ACLS TO ISSUE CERTIFICATES
51.6. MODIFYING A CERTIFICATE PROFILE
51.7. CERTIFICATE PROFILE CONFIGURATION PARAMETERS

CHAPTER 52. MANAGING THE VALIDITY OF CERTIFICATES IN IDM

Managing the validity of an existing certificate that was issued by IdM CA
Managing the validity of future certificates issued by IdM CA
52.1. VIEWING THE EXPIRY DATE OF A CERTIFICATE
52.2. REVOKING CERTIFICATES WITH THE INTEGRATED IDM CAS
52.3. RESTORING CERTIFICATES WITH THE INTEGRATED IDM CAS

CHAPTER 53. CONFIGURING IDENTITY MANAGEMENT FOR SMART CARD AUTHENTICATION

53.1. CONFIGURING THE IDM SERVER FOR SMART CARD AUTHENTICATION
53.2. CONFIGURING THE IDM CLIENT FOR SMART CARD AUTHENTICATION
53.3. ADDING A CERTIFICATE TO A USER ENTRY IN THE IDM WEB UI
53.4. ADDING A CERTIFICATE TO A USER ENTRY IN THE IDM CLI
53.5. INSTALLING TOOLS FOR MANAGING AND USING SMART CARDS
53.6. STORING A CERTIFICATE ON A SMART CARD
53.7. LOGGING IN TO IDM WITH SMART CARDS

53.8. CONFIGURING GDM ACCESS USING SMART CARD AUTHENTICATION

53.9. CONFIGURING SU ACCESS USING SMART CARD AUTHENTICATION

CHAPTER 54. CONFIGURING CERTIFICATES ISSUED BY ADCS FOR SMART CARD AUTHENTICATION IN IDM

54.1. SMART CARD AUTHENTICATION

54.2. WINDOWS SERVER SETTINGS REQUIRED FOR TRUST CONFIGURATION AND CERTIFICATE USAGE

54.3. COPYING CERTIFICATES FROM ACTIVE DIRECTORY USING SFTP

54.4. CONFIGURING THE IDM SERVER AND CLIENTS FOR SMART CARD AUTHENTICATION USING ADCS CERTIFICATES

54.5. CONVERTING THE PFX FILE

54.6. INSTALLING TOOLS FOR MANAGING AND USING SMART CARDS

54.7. STORING A CERTIFICATE ON A SMART CARD

54.8. CONFIGURING TIMEOUTS IN SSSD.CONF

54.9. CREATING CERTIFICATE MAPPING RULES FOR SMART CARD AUTHENTICATION

CHAPTER 55. CONFIGURING CERTIFICATE MAPPING RULES IN IDENTITY MANAGEMENT

55.1. CERTIFICATE MAPPING RULES FOR CONFIGURING AUTHENTICATION ON SMART CARDS

55.1.1. Certificate mapping rules for trusts with Active Directory domains

55.1.2. Components of an identity mapping rule in IdM

55.1.3. Obtaining the issuer from a certificate for use in a matching rule

55.1.4. Additional resources

55.2. CONFIGURING CERTIFICATE MAPPING FOR USERS STORED IN IDM

55.2.1. Adding a certificate mapping rule in IdM

55.2.1.1. Adding a certificate mapping rule in the IdM web UI

55.2.1.2. Adding a certificate mapping rule in the IdM CLI

55.2.2. Adding certificate mapping data to a user entry in IdM

55.2.2.1. Adding certificate mapping data to a user entry in the IdM web UI

55.2.2.2. Adding certificate mapping data to a user entry in the IdM CLI

55.3. CONFIGURING CERTIFICATE MAPPING FOR USERS WHOSE AD USER ENTRY CONTAINS THE WHOLE CERTIFICATE

55.3.1. Adding a certificate mapping rule for users whose AD entry contains whole certificates

55.3.1.1. Adding a certificate mapping rule in the IdM web UI

55.3.1.2. Adding a certificate mapping rule in the IdM CLI

55.4. CONFIGURING CERTIFICATE MAPPING IF AD IS CONFIGURED TO MAP USER CERTIFICATES TO USER ACCOUNTS

55.4.1. Adding a certificate mapping rule if the trusted AD domain is configured to map user certificates

55.4.1.1. Adding a certificate mapping rule in the IdM web UI

55.4.1.2. Adding a certificate mapping rule in the IdM CLI

55.4.2. Checking certificate mapping data on the AD side

55.5. CONFIGURING CERTIFICATE MAPPING IF AD USER ENTRY CONTAINS NO CERTIFICATE OR MAPPING DATA

55.5.1. Adding a certificate mapping rule if the AD user entry contains no certificate or mapping data

55.5.1.1. Adding a certificate mapping rule in the IdM web UI

55.5.1.2. Adding a certificate mapping rule in the IdM CLI

55.5.2. Adding a certificate to an AD user’s ID override if the user entry in AD contains no certificate or mapping data

55.5.2.1. Adding a certificate to an AD user’s ID override in the IdM web UI

55.5.2.2. Adding a certificate to an AD user’s ID override in the IdM CLI

55.6. COMBINING SEVERAL IDENTITY MAPPING RULES INTO ONE

CHAPTER 56. CONFIGURING AUTHENTICATION WITH A CERTIFICATE STORED ON THE DESKTOP OF AN IDM CLIENT
56.1. Configuring the Identity Management Server for Certificate Authentication in the Web UI
56.2. Requesting a New User Certificate and Exporting It to the Client
56.3. Making Sure the Certificate and User Are Linked Together
56.4. Configuring a Browser to Enable Certificate Authentication
56.5. Authenticating to the Identity Management Web UI with a Certificate as an Identity Management User
56.6. Configuring an IDM Client to Enable Authenticating to the CLI Using a Certificate

## Chapter 57. Using IDM CA Renewal Server

### 57.1. Explanation of IDM CA Renewal Server
The role of the CA renewal server
The role of the certmonger service on CA replicas
The correct functioning of IdM CA renewal server

### 57.2. Changing and Resetting IDM CA Renewal Server

### 57.3. Switching from an Externally to Self-Signed CA in IDM

### 57.4. Renewing the IDM CA Renewal Server with an Externally-Signed Certificate

## Chapter 58. Renewing Expired System Certificates When IDM is Offline

### 58.1. Renewing Expired System Certificates on a CA Renewal Server

### 58.2. Verifying Other IDM Servers in the IDM Domain After Renewal

### 58.3. Replacing the Web Server and LDAP Server Certificates

## Chapter 59. Generating CRL on the IDM CA Server

### 59.1. Stopping CRL Generation on an IDM Server

### 59.2. Starting CRL Generation on an IDM Replica Server

## Chapter 60. Obtaining an IDM Certificate for a Service Using Certmonger

### 60.1. Certmonger Overview
What certmonger does
Types of certificates certmonger tracks
Certmonger components

### 60.2. Obtaining an IDM Certificate for a Service Using Certmonger

### 60.3. Communication Flow for Certmonger Requesting a Service Certificate

### 60.4. Viewing the Details of a Certificate Request Tracked by Certmonger

### 60.5. Starting and Stopping Certificate Tracking

### 60.6. Renewing a Certificate Manually

### 60.7. Making Certmonger Resume Tracking of IDM Certificates on a CA Replica

## Chapter 61. Requesting Certificates Using RHEL System Roles

### 61.1. The Certificate System Role

### 61.2. Requesting a New Self-Signed Certificate Using the Certificate System Role

### 61.3. Requesting a New Certificate from IDM CA Using the Certificate System Role

### 61.4. Specifying Commands to Run Before or After Certificate Issuance Using the Certificate System Role

## Chapter 62. Restricting an Application to Trust Only a Subset of Certificates

### 62.1. Managing Lightweight Sub-Cas
62.1.1. Creating a sub-CA from the IdM WebUI
62.1.2. Deleting a sub-CA from the IdM WebUI
62.1.3. Creating a sub-CA from the IdM CLI
62.1.4. Disabling a sub-CA from the IdM CLI
62.1.5. Deleting a sub-CA from the IdM CLI

### 62.2. Downloading the Sub-CA Certificate from IDM WebUI
62.3. CREATING CA ACLS FOR WEB SERVER AND CLIENT AUTHENTICATION

62.3.1. Viewing CA ACLs in IdM CLI
62.3.2. Creating a CA ACL for web servers authenticating to web clients using certificates issued by webserver-ca
62.3.3. Creating a CA ACL for user web browsers authenticating to web servers using certificates issued by webclient-ca

62.4. OBTAINING AN IDM CERTIFICATE FOR A SERVICE USING CERTMONGER

62.5. COMMUNICATION FLOW FOR CERTMONGER REQUESTING A SERVICE CERTIFICATE

62.6. SETTING UP A SINGLE-INSTANCE APACHE HTTP SERVER

62.7. ADDING TLS ENCRYPTION TO AN APACHE HTTP SERVER

62.8. SETTING THE SUPPORTED TLS PROTOCOL VERSIONS ON AN APACHE HTTP SERVER

62.9. SETTING THE SUPPORTED CIPHERS ON AN APACHE HTTP SERVER

62.10. CONFIGURING TLS CLIENT CERTIFICATE AUTHENTICATION

62.11. REQUESTING A NEW USER CERTIFICATE AND EXPORTING IT TO THE CLIENT

62.12. CONFIGURING A BROWSER TO ENABLE CERTIFICATE AUTHENTICATION

CHAPTER 63. INVALIDATING A SPECIFIC GROUP OF RELATED CERTIFICATES QUICKLY

63.1. DISABLING CA ACLS IN IDM CLI
63.2. DISABLING AN IDM SUB-CA

CHAPTER 64. VAULTS IN IDM

64.1. VAULTS AND THEIR BENEFITS
64.2. VAULT OWNERS, MEMBERS, AND ADMINISTRATORS
64.3. STANDARD, SYMMETRIC, AND ASYMMETRIC VAULTS
64.4. USER, SERVICE, AND SHARED VAULTS
64.5. VAULT CONTAINERS
64.6. BASIC IDM VAULT COMMANDS
64.7. INSTALLING THE KEY RECOVERY AUTHORITY IN IDM

CHAPTER 65. USING IDM USER VAULTS: STORING AND RETRIEVING SECRETS

65.1. STORING A SECRET IN A USER VAULT
65.2. RETRIEVING A SECRET FROM A USER VAULT

CHAPTER 66. USING ANSIBLE TO MANAGE IDM USER VAULTS: STORING AND RETRIEVING SECRETS

66.1. ENSURING THE PRESENCE OF A STANDARD USER VAULT IN IDM USING ANSIBLE
66.2. ARCHIVING A SECRET IN A STANDARD USER VAULT IN IDM USING ANSIBLE
66.3. RETRIEVING A SECRET FROM A STANDARD USER VAULT IN IDM USING ANSIBLE

CHAPTER 67. MANAGING IDM SERVICE SECRETS: STORING AND RETRIEVING SECRETS

67.1. STORING AN IDM SERVICE SECRET IN AN ASYMMETRIC VAULT
67.2. RETRIEVING A SERVICE SECRET FOR AN IDM SERVICE INSTANCE
67.3. CHANGING AN IDM SERVICE VAULT SECRET WHEN COMPROMISED

CHAPTER 68. USING ANSIBLE TO MANAGE IDM SERVICE VAULTS: STORING AND RETRIEVING SECRETS

68.1. ENSURING THE PRESENCE OF AN ASYMMETRIC SERVICE VAULT IN IDM USING ANSIBLE
68.2. ADDING MEMBER SERVICES TO AN ASYMMETRIC VAULT USING ANSIBLE
68.3. STORING AN IDM SERVICE SECRET IN AN ASYMMETRIC VAULT USING ANSIBLE
68.4. RETRIEVING A SERVICE SECRET FOR AN IDM SERVICE USING ANSIBLE
68.5. CHANGING AN IDM SERVICE VAULT SECRET WHEN COMPROMISED USING ANSIBLE

CHAPTER 69. ENSURING THE PRESENCE AND ABSENCE OF SERVICES IN IDM USING ANSIBLE

69.1. ENSURING THE PRESENCE OF AN HTTP SERVICE IN IDM USING AN ANSIBLE PLAYBOOK
69.2. ENSURING THE PRESENCE OF AN HTTP SERVICE IN IDM ON A NON-IDM CLIENT USING AN ANSIBLE PLAYBOOK
69.3. ENSURING THE PRESENCE OF AN HTTP SERVICE ON AN IDM CLIENT WITHOUT DNS USING AN ANSIBLE PLAYBOOK 579
69.4. ENSURING THE PRESENCE OF AN EXTERNALLY SIGNED CERTIFICATE IN AN IDM SERVICE ENTRY USING AN ANSIBLE PLAYBOOK 581
69.5. USING AN ANSIBLE PLAYBOOK TO ALLOW IDM USERS, GROUPS, HOSTS, OR HOST GROUPS TO CREATE A KEYTAB OF A SERVICE 583
69.6. USING AN ANSIBLE PLAYBOOK TO ALLOW IDM USERS, GROUPS, HOSTS, OR HOST GROUPS TO RETRIEVE A KEYTAB OF A SERVICE 585
69.7. ENSURING THE PRESENCE OF A KERBEROS PRINCIPAL ALIAS OF A SERVICE USING AN ANSIBLE PLAYBOOK 587
69.8. ENSURING THE ABSENCE OF AN HTTP SERVICE IN IDM USING AN ANSIBLE PLAYBOOK 589

CHAPTER 70. ENABLING AD USERS TO ADMINISTER IDM .......................... 591
70.1. ID OVERRIDES FOR AD USERS 591
70.2. USING ID OVERRIDES TO ENABLE AD USERS TO ADMINISTER IDM 591
70.3. MANAGING IDM CLI AS AN AD USER 591

CHAPTER 71. CONFIGURING THE DOMAIN RESOLUTION ORDER TO RESOLVE SHORT AD USER NAMES .......................... 593
71.1. HOW DOMAIN RESOLUTION ORDER WORKS 593
71.2. SETTING THE GLOBAL DOMAIN RESOLUTION ORDER ON AN IDM SERVER 594
71.3. SETTING THE DOMAIN RESOLUTION ORDER FOR AN ID VIEW ON AN IDM SERVER 594
71.4. SETTING THE DOMAIN RESOLUTION ORDER IN SSSD ON AN IDM CLIENT 596

CHAPTER 72. ENABLING AUTHENTICATION USING AD USER PRINCIPAL NAMES IN IDM ..................... 598
72.1. USER PRINCIPAL NAMES IN AN AD FOREST TRUSTED BY IDM 598
72.2. ENSURING THAT AD UPNS ARE UP-TO-DATE IN IDM 598
72.3. GATHERING TROUBLESHOOTING DATA FOR AD UPN AUTHENTICATION ISSUES 599

CHAPTER 73. USING CANONICALIZED DNS HOST NAMES IN IDM .................................. 601
73.1. ADDING AN ALIAS TO A HOST PRINCIPAL 601
73.2. ENABLING CANONICALIZATION OF HOST NAMES IN SERVICE PRINCIPALS ON CLIENTS 601
73.3. OPTIONS FOR USING HOST NAMES WITH DNS HOST NAME CANONICALIZATION ENABLED 602

CHAPTER 74. MANAGING GLOBAL DNS CONFIGURATION IN IDM USING ANSIBLE PLAYBOOKS .......... 603
74.1. HOW IDM ENSURES THAT GLOBAL FORWARDERS FROM /ETC/RESOLV.CONF ARE NOT REMOVED BY NETWORKMANAGER 603
74.2. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE 604
74.3. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE 606
74.4. DNS FORWARD POLICIES IN IDM 607
74.5. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT THE FORWARD FIRST POLICY IS SET IN IDM DNS GLOBAL CONFIGURATION 608
74.6. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT GLOBAL FORWARDERS ARE DISABLED IN IDM DNS 609
74.7. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT SYNCHRONIZATION OF FORWARD AND REVERSE LOOKUP ZONES IS DISABLED IN IDM DNS 611

CHAPTER 75. MANAGING DNS ZONES IN IDM .......................................................... 613
75.1. SUPPORTED DNS ZONE TYPES 613
75.2. ADDING A PRIMARY DNS ZONE IN IDM WEB UI 614
75.3. ADDING A PRIMARY DNS ZONE IN IDM CLI 615
75.4. REMOVING A PRIMARY DNS ZONE IN IDM WEB UI 616
75.5. REMOVING A PRIMARY DNS ZONE IN IDM CLI 616
75.6. DNS CONFIGURATION PRIORITIES 616
75.7. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES 617
75.8. EDITING THE CONFIGURATION OF A PRIMARY DNS ZONE IN IDM WEB UI 619
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>93.1. PREPARING THE IDM DOMAIN FOR INSTALLING SAMBA ON DOMAIN MEMBERS</td>
<td>712</td>
</tr>
<tr>
<td>93</td>
<td>93.2. ENABLING THE AES ENCRYPTION TYPE IN ACTIVE DIRECTORY USING A GPO</td>
<td>714</td>
</tr>
<tr>
<td>93</td>
<td>93.3. INSTALLING AND CONFIGURING A SAMBA SERVER ON AN IDM CLIENT</td>
<td>715</td>
</tr>
<tr>
<td>93</td>
<td>93.4. MANUALLY ADDING AN ID MAPPING CONFIGURATION IF IDM TRUSTS A NEW DOMAIN</td>
<td>716</td>
</tr>
<tr>
<td>93</td>
<td>93.5. ADDITIONAL RESOURCES</td>
<td>718</td>
</tr>
<tr>
<td>94</td>
<td>94.1. SETTING UP A KERBEROS-AWARE NFS SERVER</td>
<td>719</td>
</tr>
<tr>
<td>94</td>
<td>94.2. SETTING UP A KERBEROS-AWARE NFS CLIENT</td>
<td>721</td>
</tr>
<tr>
<td>95</td>
<td>95.1. ENABLING NIS IN IDM</td>
<td>723</td>
</tr>
<tr>
<td>95</td>
<td>95.2. MIGRATING USER ENTRIES FROM NIS TO IDM</td>
<td>724</td>
</tr>
<tr>
<td>95</td>
<td>95.3. MIGRATING USER GROUP FROM NIS TO IDM</td>
<td>725</td>
</tr>
<tr>
<td>95</td>
<td>95.4. MIGRATING HOST ENTRIES FROM NIS TO IDM</td>
<td>726</td>
</tr>
<tr>
<td>95</td>
<td>95.5. MIGRATING NETGROUP ENTRIES FROM NIS TO IDM</td>
<td>727</td>
</tr>
<tr>
<td>95</td>
<td>95.6. MIGRATING AUTOMOUNT MAPS FROM NIS TO IDM</td>
<td>728</td>
</tr>
</tbody>
</table>
RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.

- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.


MAKING OPEN SOURCE MORE INCLUSIVE

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

In Identity Management, planned terminology replacements include:

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

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

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the *Feedback* button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the *Add Feedback* pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the *Bugzilla* website.
  2. As the Component, use *Documentation*.
  3. Fill in the *Description* field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click *Submit Bug*. 
CHAPTER 1. LOGGING IN TO IDENTITY MANAGEMENT FROM THE COMMAND LINE

Identity Management (IdM) uses the Kerberos protocol to support single sign-on. Single sign-on means that the user enters the correct user name and password only once, and then accesses IdM services without the system prompting for the credentials again.

IMPORTANT

In IdM, the System Security Services Daemon (SSSD) automatically obtains a ticket-granting ticket (TGT) for a user after the user successfully logs in to the desktop environment on an IdM client machine with the corresponding Kerberos principal name. This means that after logging in, the user is not required to use the kinit utility to access IdM resources.

If you have cleared your Kerberos credential cache or your Kerberos TGT has expired, you need to request a Kerberos ticket manually to access IdM resources. The following sections present basic user operations when using Kerberos in IdM.

1.1. USING KINIT TO LOG IN TO IDM MANUALLY

This procedure describes using the kinit utility to authenticate to an Identity Management (IdM) environment manually. The kinit utility obtains and caches a Kerberos ticket-granting ticket (TGT) on behalf of an IdM user.

NOTE

Only use this procedure if you have destroyed your initial Kerberos TGT or if it has expired. As an IdM user, when logging onto your local machine you are also automatically logging in to IdM. This means that after logging in, you are not required to use the kinit utility to access IdM resources.

Procedure

1. To log in to IdM
   - under the user name of the user who is currently logged in on the local system, use kinit without specifying a user name. For example, if you are logged in as example_user on the local system:

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

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

   [example_user@server ~]$ kinit
   kinit: Client 'example_user@EXAMPLE.COM' not found in Kerberos database while getting initial credentials

   - using a Kerberos principal that does not correspond to your local user name, pass the required user name to the kinit utility. For example, to log in as the admin user:
2. Optionally, to verify that the login was successful, use the `klist` utility to display the cached TGT. In the following example, the cache contains a ticket for the `example_user` principal, which means that on this particular host, only `example_user` is currently allowed to access IdM services:

```
$ klist
Ticket cache: KEYRING:persistent:0:0
Default principal: example_user@EXAMPLE.COM
Valid starting       Expires             Service principal
11/10/2019 08:35:45   11/10/2019 18:35:45   krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

**1.2. DESTROYING A USER’S ACTIVE KERBEROS TICKET**

This section describes how to clear the credentials cache that contains the user’s active Kerberos ticket.

**Procedure**

1. To destroy your Kerberos ticket:

   ```
   [example_user@server ~]$ kdestroy
   ```

2. Optionally, to check that the Kerberos ticket has been destroyed:

   ```
   [example_user@server ~]$ klist
   klist: Credentials cache keyring 'persistent:0:0' not found
   ```

**1.3. CONFIGURING AN EXTERNAL SYSTEM FOR KERBEROS AUTHENTICATION**

This section describes how to configure an external system so that Identity Management (IdM) users can log in to IdM from the external system using their Kerberos credentials.

Enabling Kerberos authentication on external systems is especially useful when your infrastructure includes multiple realms or overlapping domains. It is also useful if the system has not been enrolled into any IdM domain through `ipa-client-install`.

To enable Kerberos authentication to IdM from a system that is not a member of the IdM domain, define an IdM-specific Kerberos configuration file on the external system.

**Prerequisites**

- The `krb5-workstation` package is installed on the external system.
  To find out whether the package is installed, use the following CLI command:

  ```
  # dnf list installed krb5-workstation
  Installed Packages
  krb5-workstation.x86_64  1.16.1-19.el8  @BaseOS
  ```
Procedure

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

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

   **WARNING**
   Do not overwrite the existing `krb5.conf` file on the external system.

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

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

   The `KRB5_CONFIG` variable exists only temporarily until you log out. To prevent this loss, export the variable with a different file name.

3. Copy the Kerberos configuration snippets from the `/etc/krb5.conf.d/` directory to the external system.

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

Additional resources

- For details on Kerberos, see the `krb5.conf(5)`, `kinit(1)`, `klist(1)`, and `kdestroy(1)` man pages.
CHAPTER 2. VIEWING, STARTING AND STOPPING THE IDENTITY MANAGEMENT SERVICES

Identity Management (IdM) servers are Red Hat Enterprise Linux systems that work as domain controllers (DCs). A number of different services are running on IdM servers, most notably the Directory Server, Certificate Authority (CA), DNS, and Kerberos.

2.1. THE IDM SERVICES

2.1.1. List of services hosted by IdM servers

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

Kerberos

the `krb5kdc` and `kadmin` services

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

Kerberos is divided into two parts:

- The `krb5kdc` service is the Kerberos Authentication service and Key Distribution Center (KDC) daemon.
- The `kadmin` service is the Kerberos database administration program.

For information on how to authenticate using Kerberos in IdM, see Logging in to Identity Management from the command line and Logging in to IdM in the Web UI: Using a Kerberos ticket.

LDAP directory server

the `dirsrv` service

The IdM LDAP directory server instance stores all IdM information, such as information related to Kerberos, user accounts, host entries, services, policies, DNS, and others. The LDAP directory server instance is based on the same technology as Red Hat Directory Server. However, it is tuned to IdM-specific tasks.

Certificate Authority

the `pki-tomcatd` service

The integrated certificate authority (CA) is based on the same technology as Red Hat Certificate System. `pki` is the Command-Line Interface for accessing Certificate System services.

You can also install the server without the integrated CA if you create and provide all required certificates independently.

For more information, see Planning your CA services.

Domain Name System (DNS)

the `named` service
IdM uses **DNS** for dynamic service discovery. The IdM client installation utility can use information from DNS to automatically configure the client machine. After the client is enrolled in the IdM domain, it uses DNS to locate IdM servers and services within the domain. The **BIND** (Berkeley Internet Name Domain) implementation of the DNS (Domain Name System) protocols in Red Hat Enterprise Linux includes the **named** DNS server. **named-pkcs11** is a version of the BIND DNS server built with native support for the PKCS#11 cryptographic standard.

For information, see [Planning your DNS services and host names](#).

**Apache HTTP Server**

the **httpd** service

The **Apache HTTP web server** provides the IdM Web UI, and also manages communication between the Certificate Authority and other IdM services.

**Samba / Winbind**

**smb** and **winbind** services

Samba implements the Server Message Block (SMB) protocol, also known as the Common Internet File System (CIFS) protocol, in Red Hat Enterprise Linux. Via the smb service, the SMB protocol enables you to access resources on a server, such as file shares and shared printers. If you have configured a Trust with an Active Directory (AD) environment, the `Winbind` service manages communication between IdM servers and AD servers.

**One-time password (OTP) authentication**

the **ipa-otpd** service

One-time passwords (OTP) are passwords that are generated by an authentication token for only one session, as part of two-factor authentication. OTP authentication is implemented in Red Hat Enterprise Linux via the **ipa-otpd** service.

For more information, see [Logging in to the Identity Management Web UI using one time passwords](#).

**OpenDNSSEC**

the **ipa-dnskeysyncd** service

**OpenDNSSEC** is a DNS manager that automates the process of keeping track of DNS security extensions (DNSSEC) keys and the signing of zones. The **ipa-dnskeysyncd** service manages synchronization between the IdM Directory Server and OpenDNSSEC.
2.1.2. List of services hosted by IdM clients

- **System Security Services Daemon**: the `sssd` service

  The **System Security Services Daemon** (SSSD) is the client-side application that manages user authentication and caching credentials. Caching enables the local system to continue normal authentication operations if the IdM server becomes unavailable or if the client goes offline.

  For more information, see [Understanding SSSD and its benefits](#).

- **Certmonger**: the `certmonger` service

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

  For more information, see [Obtaining an IdM certificate for a service using certmonger](#).
2.2. VIEWING THE STATUS OF IDM SERVICES

To view the status of the IdM services that are configured on your IdM server, run the `ipactl status` command:

```
[root@server ~]# ipactl status
Directory Service: RUNNING
krb5kdc Service: RUNNING
kadmin Service: RUNNING
named Service: RUNNING
httpd Service: RUNNING
pki-tomcatd Service: RUNNING
smb Service: RUNNING
winbind Service: RUNNING
ipa-otpd Service: RUNNING
ipa-dnskeysyncd Service: RUNNING
ipa: INFO: The ipactl command was successful
```

The output of the `ipactl status` command on your server depends on your IdM configuration. For example, if an IdM deployment does not include a DNS server, the `named` service is not present in the list.

**NOTE**

You cannot use the IdM web UI to view the status of all the IdM services running on a particular IdM server. Kerberized services running on different servers can be viewed in the Identity → Services tab of the IdM web UI.

You can start or stop the entire server, or an individual service only.

To start, stop, or restart the entire IdM server, see:

- Section 2.3, “Starting and stopping the entire Identity Management server: the `ipactl` utility”
To start, stop, or restart an individual IdM service, see:

- Section 2.4, “Starting and stopping an individual Identity Management service: the systemctl utility”

To display the version of IdM software, see:

- Section 2.5, “Methods for displaying IdM software version”

### 2.3. STARTING AND STOPPING THE ENTIRE IDENTITY MANAGEMENT SERVER: THE IPACTL UTILITY

Use the ipactl utility to stop, start, or restart the entire IdM server along with all the installed services. Using the ipactl utility ensures all services are stopped, started, or restarted in the appropriate order. You do not need to have a valid Kerberos ticket to run the ipactl commands.

**ipactl commands**

To start the entire IdM server:

```
# ipactl start
```

To stop the entire IdM server:

```
# ipactl stop
```

To restart the entire IdM server:

```
# ipactl restart
```

To show the status of all the services that make up IdM:

```
# ipactl status
```

**IMPORTANT**

You cannot use the IdM web UI to perform the ipactl commands.

### 2.4. STARTING AND STOPPING AN INDIVIDUAL IDENTITY MANAGEMENT SERVICE: THE SYSTEMCTL UTILITY

Changing IdM configuration files manually is generally not recommended. However, certain situations require that an administrator performs a manual configuration of specific services. In such situations, use the systemctl utility to stop, start, or restart an individual IdM service.

For example, use systemctl after customizing the Directory Server behavior, without modifying the other IdM services:

```
# systemctl restart dirsrv@REALM-NAME.service
```

Also, when initially deploying an IdM trust with Active Directory, modify the /etc/sssd/sssd.conf file, adding:
• specific parameters to tune the timeout configuration options in an environment where remote servers have a high latency

• specific parameters to tune the Active Directory site affinity

• overrides for certain configuration options that are not provided by the global IdM settings

To apply the changes you have made in the /etc/sssd/sssd.conf file:

```
# systemctl restart sssd.service
```

Running `systemctl restart sssd.service` is required because the System Security Services Daemon (SSSD) does not automatically re-read or re-apply its configuration.

Note that for changes that affect IdM identity ranges, a complete server reboot is recommended.

**IMPORTANT**

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

**Useful systemctl commands**

To start a particular IdM service:

```
# systemctl start name.service
```

To stop a particular IdM service:

```
# systemctl stop name.service
```

To restart a particular IdM service:

```
# systemctl restart name.service
```

To view the status of a particular IdM service:

```
# systemctl status name.service
```

**IMPORTANT**

You cannot use the IdM web UI to start or stop the individual services running on IdM servers. You can only use the web UI to modify the settings of a Kerberized service by navigating to **Identity → Services** and selecting the service.

### 2.5. METHODS FOR DISPLAYING IDM SOFTWARE VERSION

You can display the IdM version number with:

• the IdM WebUI
Displaying version through the WebUI
In the IdM WebUI, the software version can be displayed by choosing **About** from the username menu at the top-right.

Displaying version with **ipa** commands
From the command line, use the **ipa --version** command.

```
[root@server ~]# ipa --version
VERSION: 4.8.0, API_VERSION: 2.233
```

Displaying version with **rpm** commands
If IdM services are not operating properly, you can use the **rpm** utility to determine the version number of the **ipa-server** package that is currently installed.

```
[root@server ~]# rpm -q ipa-server
ipa-server-4.8.0-11.module+el8.1.0+4247+9f3fd721.x86_64
```
CHAPTER 3. 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.
  For details about obtaining a valid Kerberos ticket, see Logging in to Identity Management from the command line.

3.1. 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 that are used to manage 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.

3.2. WHAT IS THE IPA HELP

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

IPA command line interface (CLI) generates available help topics from loaded IdM plugin modules. If you want to run the IPA help successfully, you need to:

- Have an IdM server installed and running.

- Be authenticated with a valid Kerberos ticket.

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

Executing help with options has the following syntax:

```
$ ipa help [TOPIC | COMMAND | topics | commands]
```
• [] – Brackets mean that all parameters are optional and you can write just `ipa help` and the command will be executed.

• | – The pipe character means or. Therefore, you can use TOPIC or COMMAND or topics or commands with the basic `ipa help` command.

• topics – You can run the command `ipa help topics` and it will execute correctly. The command displays a list of topics that are covered by IPA help, for example, `user`, `cert`, `server` and many others.

• TOPIC – The TOPIC with capital letters means variable, therefore, you can use the particular topic, for example, `ipa help user`

• commands – You can run the command `ipa help commands` and it will execute correctly. The command displays a list of commands which are covered by the IPA help, for example, `user-add`, `ca-enable`, `server-show` and many others.

• COMMAND – The COMMAND with capital letters means variable, therefore, you can use the particular command, for example, `ipa help user-add`

3.3. USING IPA HELP TOPICS

The following procedure helps you to understand using the IPA help in the command line interface.

**Procedure**

1. Open 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 theIPA help command 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.

3.4. USING IPA HELP COMMANDS

The following procedure helps you to understand creating the IPA help commands in the command line interface.
Procedure

1. Open 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

- For details, see `man ipa` page.

### 3.5. 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

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 includes also object: ipa user-mod USER_NAME [options].

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

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

Prerequisites

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

Procedure

1. Open terminal and connect to the IdM server.
2. Enter the command for adding a new user:

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

The command runs a script where you can add 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 if the suggested user name works for you. User name must be unique for the whole IdM database. If an error occurs, that the user already exists, you need to start from the beginning with the ipa user-add command and try a different user name.

After you successfully added the user name, the user account has been added to the IdM database and the IPA command line interface (CLI) prints on the output the following log:

```
----------------------
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
```
As you can see, a user password is not set to the user account. If you want to add also password, use the `ipa user-add` command in the following syntax:

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

The IPA CLI then asks you for adding or confirming a user name and password.

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

**Additional resources**

For more information about parameters, enter the following help command to the command line:

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

### 3.7. 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 in the IdM server.

**Procedure**

1. Open terminal and connect to the IdM server.
2. Enter the command for adding a password:
   
   ```bash
   $ 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.

After you successfully added the user name, the user account has been added to the IdM database and the IPA CLI prints on the output the following log:

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

Additional resources

For more information about parameters, enter the following help command to the command line:

```
$ ipa help user-mod
```

### 3.8. 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} ...
  ```

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.

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:

```bash
$ ipa permission-mod --right=read --right=write ...
```

or

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

### 3.9. 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 `>`, ampersand (`&`), asterisk (`*`), or 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 4. SEARCHING IDENTITY MANAGEMENT ENTRIES FROM THE COMMAND LINE

The following sections describe how to use IPA commands, which helps you to find or show objects.

4.1. OVERVIEW OF LISTING IDM ENTRIES

This section describes the `ipa *-find` commands, which can help you to search for a particular type of IdM entries.

To list all the `find` commands, use the following `ipa` help command:

```
$ ipa help commands | grep find
```

You may need to check if a particular user is included in the IdM database. You can then list all users with the following command:

```
$ ipa user-find
```

To list user groups whose specified attributes contain a keyword:

```
$ ipa group-find keyword
```

For example, the `ipa group-find admin` command lists all groups whose names or descriptions include the string `admin`:

```
----------------
3 groups matched
----------------
Group name: admins
Description: Account administrators group
GID: 427200002

Group name: editors
Description: Limited admins who can edit other users
GID: 427200002

Group name: trust admins
Description: Trusts administrators group
```

When searching user groups, you can also limit the search results to groups that contain a particular user:

```
$ ipa group-find --user=user_name
```

To search for groups that do not contain a particular user:

```
$ ipa group-find --no-user=user_name
```

4.2. SHOWING DETAILS FOR A PARTICULAR ENTRY
Use the `ipa *-show` command to display details about a particular IdM entry.

**Procedure**

- To display details about a host named `server.example.com`:

  ```
  $ ipa host-show server.example.com
  Host name: server.example.com
  Principal name: host/server.example.com@EXAMPLE.COM
  ...
  ```

### 4.3. ADJUSTING THE SEARCH SIZE AND TIME LIMIT

Some queries, such as requesting a list of IdM users, can return a very large number of entries. By tuning these search operations, you can improve the overall server performance when running the `ipa *-find` commands, such as `ipa user-find`, and when displaying corresponding lists in the Web UI.

**Search size limit**

- Defines the maximum number of entries returned for a request sent to the server from a client’s CLI or from a browser accessing the IdM Web UI.
- Default: 100 entries.

**Search time limit**

- Defines the maximum time (in seconds) that the server waits for searches to run. Once the search reaches this limit, the server stops the search and returns the entries discovered in that time.
- Default: 2 seconds.

If you set the values to `-1`, IdM will not apply any limits when searching.

**IMPORTANT**

- Setting search size or time limits too high can negatively affect server performance.

#### 4.3.1. Adjusting the search size and time limit in the command line

The following text describes adjusting search size and time limits in the command line:

**Procedure**

1. To display current search time and size limits in CLI, use the `ipa config-show` command:

   ```
   $ ipa config-show
   Search time limit: 2
   Search size limit: 100
   ```
2. To adjust the limits globally for all queries, use the `ipa config-mod` command and add the `--searchrecordslimit` and `--searchtimelimit` options. For example:

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

3. To adjust the limits only for a specific query, add the `--sizelimit` or `--timelimit` options to the command. For example:

   ```
   $ ipa user-find --sizelimit=200 --timelimit=120
   ```

4.3.2. Adjusting the search size and time limit in the Web UI

The following text describes adjusting search size and time limits in the IdM Web UI:

- Globally
- For a specific entry

**Procedure**

To adjust the limits globally for all queries:

1. Log in to the IdM Web UI.
2. Click **IPA Server**.
3. On the **IPA Server** tab, click **Configuration**.
4. Set the required values in the **Search Options** area.
   - Default values are:
     - Search size limit: 100 entries
     - Search time limit: 2 seconds
5. Click **Save** at the top of the page.
After saving the values, search an entry and verify the result.
CHAPTER 5. ACCESSING THE IDM WEB UI IN A WEB BROWSER

The following sections provide an overview of the IdM (Identity Management) Web UI and describe how to access it.

5.1. WHAT IS THE IDM WEB UI

The IdM (Identity Management) Web UI is a web application for IdM administration, a graphical alternative to the IdM command line tools.

You can access the IdM Web UI as:

- **IdM users**: A limited set of operations depending on permissions granted to the user in the IdM server. Basically, active IdM users can log in to the IdM server and configure their own account. They cannot change settings of other users or the IdM server settings.
- **Administrators**: Full access rights to the IdM server.
- **Active Directory users**: A set of operations depending on permissions granted to the user. Active Directory users can now be administrators for Identity Management. For details, see Enabling AD users to administer IdM.

5.2. WEB BROWSERS SUPPORTED FOR ACCESSING THE WEB UI

Identity Management (IdM) supports the following browsers for connecting to the Web UI:

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

You might experience problems accessing the IdM Web UI with a smart card if your browser attempts to use TLS v1.3:

```plaintext
ssl:error [pid 125757:tid 140436077168384] [client 999.999.999.999:99999] AH: verify client post handshake
ssl:error [pid 125757:tid 140436077168384] [client 999.999.999.999:99999] AH10158: cannot perform post-handshake authentication
```

This is because the most recent versions of browsers do not have TLS Post-Handshake Authentication (PHA) enabled by default, or they do not support PHA. PHA is necessary to require a TLS client certificate for only a portion of a web site, such as when accessing the IdM Web UI with smart card authentication.

To resolve this issue, disable TLS v1.3 by adding `-TLSv1.3` to the `SSLProtocol` option in the `/etc/httpd/conf.d/ssl.conf` configuration file:

```plaintext
SSLProtocol all -TLSv1 -TLSv1.1 -TLSv1.3
```

Note that IdM manages the `ssl.conf` file and might overwrite its contents during package updates. Verify custom settings after updating IdM packages.

5.3. ACCESSING THE WEB UI

The following procedure describes the first logging in to the IdM (Identity Management) Web UI with a password.

After the first login you can configure your IdM server to authenticate with:

- **Kerberos ticket**
  
  For details, see Section 6.1, “Kerberos authentication in Identity Management”.

- **Smart card**
  
  For details, see Configuring the IdM server for smart card authentication.

- **One time password (OTP)** — this can be combined with password and Kerberos authentication.
  
  For details, see Section 7.2, “One time password (OTP) authentication in Identity Management”.

Procedure

1. Type an IdM server URL into the browser address bar. The name will look similarly to the following example:

   ```plaintext
   https://server.example.com
   ```

   You just need to change `server.example.com` with a DNS name of your IdM server.

   This opens the IdM Web UI login screen in your browser.
If the server does not respond or the login screen does not open, check the DNS settings on the IdM server to which you are connecting.

If you use a self-signed certificate, the browser issues a warning. Check the certificate and accept the security exception to proceed with the login.

To avoid security exceptions, install a certificate signed by a certificate authority.

2. On the Web UI login screen, enter the administrator account credentials you added during the IdM server installation.

For details, see Installing an Identity Management server: With integrated DNS, with an integrated CA.

You can enter your personal account credentials as well if they are already entered in the IdM server.

3. Click Log in.

After the successful login, you can start configuring the IdM server.
CHAPTER 6. LOGGING IN TO IDM IN THE WEB UI: USING A KERBEROS TICKET

The following sections describe the initial configuration of your environment to enable Kerberos login to the IdM Web UI and accessing IdM using Kerberos authentication.

Prerequisites

- Installed IdM server in your network environment
  For details, see Installing Identity Management in Red Hat Enterprise Linux 9

6.1. KERBEROS AUTHENTICATION IN IDENTITY MANAGEMENT

Identity Management (IdM) uses the Kerberos protocol to support single sign-on. Single sign-on authentication allows you to provide the correct user name and password only once, and you can then access Identity Management services without the system prompting for credentials again.

The IdM server provides Kerberos authentication immediately after the installation if the DNS and certificate settings have been configured properly. For details, see Installing Identity Management.

To use Kerberos authentication on hosts, install:

- the IdM client
  For details, see Preparing the system for Identity Management client installation.

- the krb5conf package

6.2. USING KINIT TO LOG IN TO IDM MANUALLY

This procedure describes using the kinit utility to authenticate to an Identity Management (IdM) environment manually. The kinit utility obtains and caches a Kerberos ticket-granting ticket (TGT) on behalf of an IdM user.

**NOTE**

Only use this procedure if you have destroyed your initial Kerberos TGT or if it has expired. As an IdM user, when logging onto your local machine you are also automatically logging in to IdM. This means that after logging in, you are not required to use the kinit utility to access IdM resources.

Procedure

1. To log in to IdM
   - under the user name of the user who is currently logged in on the local system, use kinit without specifying a user name. For example, if you are logged in as example_user on the local system:

     ```
     [example_user@server ~]$ kinit
     Password for example_user@EXAMPLE.COM:
     [example_user@server ~]$
     ```
If the user name of the local user does not match any user entry in IdM, the authentication attempt fails:

```
[example_user@server ~]$ kinit
kinit: Client 'example_user@EXAMPLE.COM' not found in Kerberos database while getting initial credentials
```

- using a Kerberos principal that does not correspond to your local user name, pass the required user name to the `kinit` utility. For example, to log in as the `admin` user:

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

2. Optionally, to verify that the login was successful, use the `klist` utility to display the cached TGT. In the following example, the cache contains a ticket for the `example_user` principal, which means that on this particular host, only `example_user` is currently allowed to access IdM services:

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

Valid starting      Expires             Service principal
11/10/2019 08:35:45   11/10/2019 18:35:45   krbtgt/EXAMPLE.COM@EXAMPLE.COM
```

### 6.3. Configuring the Browser for Kerberos Authentication

To enable authentication with a Kerberos ticket, you may need a browser configuration.

The following steps help you to support Kerberos negotiation for accessing the IdM domain.

Each browser supports Kerberos in a different way and needs different set up. The IdM Web UI includes guidelines for the following browsers:

- Firefox
- Chrome

**Procedure**

1. Open the IdM Web UI login dialog in your web browser.
2. Click the link for browser configuration on the Web UI login screen.
3. Follow the steps on the configuration page.

After the setup, turn back to the IdM Web UI and click Log in.

6.4. LOGGING IN TO THE WEB UI USING A KERBEROS TICKET

This procedure describes logging in to the IdM Web UI using a Kerberos ticket-granting ticket (TGT).

The TGT expires at a predefined time. The default time interval is 24 hours and you can change it in the IdM Web UI.

After the time interval expires, you need to renew the ticket:

- Using the kinit command.
- Using IdM login credentials in the Web UI login dialog.

Procedure

- Open the IdM Web UI.
  If Kerberos authentication works correctly and you have a valid ticket, you will be automatically authenticated and the Web UI opens.

  If the ticket is expired, it is necessary to authenticate yourself with credentials first. However, next time the IdM Web UI will open automatically without opening the login dialog.
If you see an error message **Authentication with Kerberos failed**, verify that your browser is configured for Kerberos authentication. See Section 6.3, “Configuring the browser for Kerberos authentication”.

6.5. CONFIGURING AN EXTERNAL SYSTEM FOR KERBEROS AUTHENTICATION

This section describes how to configure an external system so that Identity Management (IdM) users can log in to IdM from the external system using their Kerberos credentials.

Enabling Kerberos authentication on external systems is especially useful when your infrastructure includes multiple realms or overlapping domains. It is also useful if the system has not been enrolled into any IdM domain through `ipa-client-install`.

To enable Kerberos authentication to IdM from a system that is not a member of the IdM domain, define an IdM-specific Kerberos configuration file on the external system.

Prerequisites

- The **krb5-workstation** package is installed on the external system. To find out whether the package is installed, use the following CLI command:

  ```
  # dnf list installed krb5-workstation
  Installed Packages
  krb5-workstation.x86_64  1.16.1-19.el8  @BaseOS
  ```

Procedure

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

  ```
  # scp /etc/krb5.conf root@externalsystem.example.com:/etc/krb5_ipa.conf
  ```
2. On the external system, set the terminal session to use the copied IdM Kerberos configuration file:

   $ export KRB5_CONFIG=/etc/krb5_ipa.conf

   The KRB5_CONFIG variable exists only temporarily until you log out. To prevent this loss, export the variable with a different file name.

3. Copy the Kerberos configuration snippets from the /etc/krb5.conf.d/ directory to the external system.

4. Configure the browser on the external system, as described in Section 6.3, “Configuring the browser for Kerberos authentication”.

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

6.6. WEB UI LOGIN FOR ACTIVE DIRECTORY USERS

To enable Web UI login for Active Directory users, define an ID override for each Active Directory user in the default trust view. For example:

   [admin@server ~]$ ipa idoverrideuser-add 'Default Trust View' ad_user@ad.example.com
CHAPTER 7. LOGGING IN TO THE IDENTITY MANAGEMENT WEB UI USING ONE TIME PASSWORDS

Access to IdM Web UI can be secured using several methods. The basic one is password authentication.

To increase the security of password authentication, you can add a second step and require automatically generated one-time passwords (OTPs). The most common usage is to combine password connected with the user account and a time limited one time password generated by a hardware or software token.

The following sections help you to:

- Understand how the OTP authentication works in IdM.
- Configure OTP authentication on the IdM server.
- Create OTP tokens and synchronize them with the FreeOTP app in your phone.
- Authenticate to the IdM Web UI with the combination of user password and one time password.
- Re-synchronize tokens in the Web UI.

7.1. PREREQUISITES

- Accessing the IdM Web UI in a web browser

7.2. ONE TIME PASSWORD (OTP) AUTHENTICATION IN IDENTITY MANAGEMENT

One-time passwords bring an additional step to your authentication security. The authentication uses your password + an automatically generated one time password.

To generate one time passwords, you can use a hardware or software token. IdM supports both software and hardware tokens.

Identity Management supports the following two standard OTP mechanisms:

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

**IMPORTANT**

IdM does not support OTP logins for Active Directory trust users.

7.3. ENABLING THE ONE TIME PASSWORD IN THE WEB UI

The IdM Web UI allows you to configure hardware or software device to generate one-time passwords.

The one time password is entered just after the usual password in the dedicated field in the login dialog.
Only administrators can enable OTP authentication in the user settings.

**Prerequisites**

- Administration privileges

**Procedure**

1. Log in to the IdM Web UI with your username and password.

2. Open the **Identity → Users → Active userstab**.

3. Click your username to open the user settings.

4. In the **User authentication types**, select **Two factor authentication (password + OTP)**.

5. Click **Save**.

At this point, the OTP authentication is enabled on the IdM server.

Now you or users themselves need to assign a new token ID to the user account.

### 7.4. ADDING OTP TOKENS IN THE WEB UI

The following section helps you to add token to the IdM Web UI and to your software token generator.

**Prerequisites**

- Active user account on the IdM server.

- Administrator has enabled OTP for the particular user account in the IdM Web UI.

- A software device generating OTP tokens, for example FreeOTP.

**Procedure**

1. Log in to the IdM Web UI with your user name and password.

2. To create the token in your mobile phone, open the **Authentication → OTP Tokens** tab.

3. Click **Add**.
4. In the Add OTP token dialog box, leave everything unfilled and click Add. At this stage, the IdM server creates a token with default parameters at the server and opens a page with a QR code.

5. Copy the QR code into your mobile phone.

6. Click OK to close the QR code.

Now you can generate one time passwords and log in with them to the IdM Web UI.

7.5. LOGGING INTO THE WEB UI WITH A ONE TIME PASSWORD

This procedure describes the first login into the IdM Web UI using a one time password (OTP).
Prerequisites

1. OTP configuration enabled on the Identity Management server for the user account you are using for the OTP authentication. Administrators as well as users themselves can enable OTP. To enable the OTP configuration, see Section 7.3, “Enabling the one time password in the Web UI”.

2. A hardware or software device generating OTP tokens configured.

Procedure

1. In the Identity Management login screen, enter your user name or a user name of the IdM server administrator account.

2. Add the password for the user name entered above.

3. Generate a one time password on your device.

4. Enter the one time password right after the password (without space).

5. Click Log in.
   If the authentication fails, synchronize OTP tokens.

   If your CA uses a self-signed certificate, the browser issues a warning. Check the certificate and accept the security exception to proceed with the login.

   If the the IdM Web UI does not open, verify the DNS configuration of your Identity Management server.

After successful login, the IdM Web UI appears.

7.6. SYNCHRONIZING OTP TOKENS USING THE WEB UI

If the login with OTP (One Time Password) fails, OTP tokens are not synchronized correctly.

The following text describes token re-synchronization.

Prerequisites

- A login screen opened.
A device generating OTP tokens configured.

**Procedure**

1. On the IdM Web UI login screen, click **Sync OTP Token**

![Login Screen](image1)

2. In the login screen, enter your username and the Identity Management password.

3. Generate one time password and enter it in the **First OTP** field.

4. Generate another one time password and enter it in the **Second OTP** field.

5. Optionally, enter the token ID.

![Login Screen](image2)

6. Click **Sync OTP Token**

After the successful synchronization, you can log in to the IdM server.

**7.7. Changing Expired Passwords**
Administrators of Identity Management can enforce you having to change your password at the next login. It means that you cannot successfully log in to the IdM Web UI until you change the password.

Password expiration can happen during your first login to the Web UI.

If the expiration password dialog appears, follow the instructions in the procedure.

**Prerequisites**

- A login screen opened.
- Active account to the IdM server.

**Procedure**

1. In the password expiration login screen, enter the user name.
2. Add the password for the user name entered above.
3. In the OTP field, generate a one time password, if you use the one time password authentication.
   If you do not have enabled the OTP authentication, leave the field empty.
4. Enter the new password twice for verification.
5. Click **Reset Password**.

After the successful password change, the usual login dialog displays. Log in with the new password.
CHAPTER 8. TROUBLESHOOTING AUTHENTICATION WITH SSSD IN IDM

Authentication in an Identity Management (IdM) environment involves many components:

On the IdM client:

- The SSSD service.
- The Name Services Switch (NSS).
- Pluggable Authentication Modules (PAM).

On the IdM server:

- The SSSD service.
- The IdM Directory Server.
- The IdM Kerberos Key Distribution Center (KDC).

If you are authenticating as an Active Directory (AD) user:

- The Directory Server on an AD Domain Controller.
- The Kerberos server on an AD Domain Controller.

To authenticate users, you must be able to perform the following functions with the SSSD service:

- Retrieve user information from the authentication server.
- Prompt the user for their credentials, pass those credentials to the authentication server, and process the outcome.

The following sections discuss how information flows between the SSSD service and servers that store user information, so you can troubleshoot failing authentication attempts in your environment:

1. Data flow when retrieving IdM user information with SSSD
2. Data flow when retrieving AD user information with SSSD
3. Data flow when authenticating as a user with SSSD in IdM
4. Narrowing the scope of authentication issues
5. SSSD log files and logging levels
6. Enabling detailed logging for SSSD in the sssd.conf file
7. Enabling detailed logging for SSSD with the sssctl command
8. Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM server
9. Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM client
8.1. DATA FLOW WHEN RETRIEVING IDM USER INFORMATION WITH SSSD

The following diagram is a simplification of the information flow between an IdM client and an IdM server during a request for IdM user information with the command `getent passwd <idm_user_name>`.

1. The `getent` command triggers the `getpwnam` call from the `libc` library.

2. The `libc` library references the `/etc/nsswitch.conf` configuration file to check which service is responsible for providing user information, and discovers the entry `sss` for the SSSD service.

3. The `libc` library opens the `nss_sss` module.

4. The `nss_sss` module checks the memory-mapped cache for the user information. If the data is present in the cache, the `nss_sss` module returns it.

5. If the user information is not in the memory-mapped cache, the request is passed to the SSSD `sssd_nss` responder process.

6. The SSSD service checks its cache. If the data is present in the cache and valid, the `sssd_nss` responder reads the data from the cache and returns it to the application.

7. If the data is not present in the cache or it is expired, the `sssd_nss` responder queries the appropriate back-end process and waits for a reply. The SSSD service uses the IPA backend in an IdM environment, enabled by the setting `id_provider=ipa` in the `sssd.conf` configuration file.
8. The `sssd_be` back-end process connects to the IdM server and requests the information from the IdM LDAP Directory Server.

9. The SSSD back-end on the IdM server responds to the SSSD back-end process on the IdM client.

10. The SSSD back-end on the client stores the resulting data in the SSSD cache and alerts the responder process that the cache has been updated.

11. The `sssd_nss` front-end responder process retrieves the information from the SSSD cache.

12. The `sssd_nss` responder sends the user information to the `nss_sss` responder, completing the request.

13. The `libc` library returns the user information to the application that requested it.

### 8.2. DATA FLOW WHEN RETRIEVING AD USER INFORMATION WITH SSSD

If you have established a cross-forest trust between your IdM environment and an Active Directory (AD) domain, the information flow when retrieving AD user information on an IdM client is very similar to the information flow when retrieving IdM user information, with the additional step of contacting the AD user database.

The following diagram is a simplification of the information flow when a user requests information about an AD user with the command `getent passwd <ad_user_name@ad.example.com>`. This diagram does not include the internal details discussed in the Data flow when retrieving IdM user information with SSSD section. It focuses on the communication between the SSSD service on an IdM client, the SSSD service on an IdM server, and the LDAP database on an AD Domain Controller.
1. The IdM client looks to its local SSSD cache for AD user information.

2. If the IdM client does not have the user information, or the information is stale, the SSSD service on the client contacts the `extdom_extop` plugin on the IdM server to perform an LDAP extended operation and requests the information.

3. The SSSD service on the IdM server looks for the AD user information in its local cache.

4. If the IdM server does not have the user information in its SSSD cache, or its information is stale, it performs an LDAP search to request the user information from an AD Domain Controller.

5. The SSSD service on the IdM server receives the AD user information from the AD domain controller and stores it in its cache.

6. The `extdom_extop` plugin receives the information from the SSSD service on the IdM server, which completes the LDAP extended operation.

7. The SSSD service on the IdM client receives the AD user information from the LDAP extended operation.

8. The IdM client stores the AD user information in its SSSD cache and returns the information to the application that requested it.

### 8.3. DATA FLOW WHEN AUTHENTICATING AS A USER WITH SSSD IN IDM

Authenticating as a user on an IdM server or client involves the following components:

- The service that initiates the authentication request, such as the sshd service.
The Pluggable Authentication Module (PAM) library and its modules.

- The SSSD service, its responders, and back-ends.

- A smart card reader, if smart card authentication is configured.

- The authentication server:
  - IdM users are authenticated against an IdM Kerberos Key Distribution Center (KDC).
  - Active Directory (AD) users are authenticated against an AD Domain Controller (DC).

The following diagram is a simplification of the information flow when a user needs to authenticate during an attempt to log in locally to a host via the SSH service on the command line.

1. The authentication attempt with the `ssh` command triggers the `libpam` library.

2. The `libpam` library references the PAM file in the `/etc/pam.d/` directory that corresponds to the service requesting the authentication attempt. In this example involving authenticating via the SSH service on the local host, the `libpam` library checks the `/etc/pam.d/system-auth` configuration file and discovers the `pam_sss.so` entry for the SSSD PAM:
3. To determine which authentication methods are available, the libpam library opens the pam_sss module and sends an SSS_PAM_PREAUTH request to the sssd_pam PAM responder of the SSSD service.

4. If smart card authentication is configured, the SSSD service spawns a temporary p11_child process to check for a smart card and retrieve certificates from it.

5. If smart card authentication is configured for the user, the sssd_pam responder attempts to match the certificate from the smart card with the user. The sssd_pam responder also performs a search for the groups that the user belongs to, since group membership might affect access control.

6. The sssd_pam responder sends an SSS_PAM_PREAUTH request to the sssd_be back-end responder to see which authentication methods the server supports, such as passwords or 2-factor authentication. In an IdM environment, where the SSSD service uses the IPA responder, the default authentication method is Kerberos. For this example, the user authenticates with a simple Kerberos password.

7. The sssd_be responder spawns a temporary krb5_child process.

8. The krb5_child process contacts the KDC on the IdM server and checks for available authentication methods.

9. The KDC responds to the request:
   a. The krb5_child process evaluates the reply and sends the results back to the sssd_be backend process.
   b. The sssd_be backend process receives the result.
   c. The sssd_pam responder receives the result.
   d. The pam_sss module receives the result.

10. If password authentication is configured for the user, the pam_sss module prompts the user for their password. If smart card authentication is configured, the pam_sss module prompts the user for their smart card PIN.

11. The module sends an SSS_PAM_AUTHENTICATE request with the user name and password, which travels to:
   a. The sssd_pam responder.
   b. The sssd_be back-end process.

12. The sssd_be process spawns a temporary krb5_child process to contact the KDC.

13. The krb5_child process attempts to retrieve a Kerberos Ticket Granting Ticket (TGT) from the KDC with the user name and password the user provided.

14. The krb5_child process receives the result of the authentication attempt.

15. The krb5_child process:
   a. Stores the TGT in a credential cache.
b. Returns the authentication result to the **sssd_be** back-end process.

16. The authentication result travels from the **sssd_be** process to:
   a. The **sssd_pam** responder.
   b. The **pam_sss** module.

17. The **pam_sss** module sets an environment variable with the location of the user’s TGT so other applications can reference it.

### 8.4. NARROWING THE SCOPE OF AUTHENTICATION ISSUES

To successfully authenticate a user, you must be able to retrieve user information with the SSSD service from the database that stores user information. The following procedure describes steps to test different components of the authentication process so you can narrow the scope of authentication issues when a user is unable to log in.

#### Procedure

1. Verify that the SSSD service and its processes are running.

   ```
   [root@client ~]# pstree -a | grep sssd
   |-sssd -i --logger=files
   | | | ssd_be --domain implicit_files --uid 0 --gid 0 --logger=files
   | | | ssd_be --domain example.com --uid 0 --gid 0 --logger=files
   | | | ssd_ifp --uid 0 --gid 0 --logger=files
   | | | ssd_nss --uid 0 --gid 0 --logger=files
   | | | ssd_pac --uid 0 --gid 0 --logger=files
   | | ssd_pam --uid 0 --gid 0 --logger=files
   | | ssd_ssh --uid 0 --gid 0 --logger=files
   | | sssd_sudo --uid 0 --gid 0 --logger=files
   | ssd_kcm --uid 0 --gid 0 --logger=files
   ```

2. Verify that the client can contact the user database server via the IP address.

   ```
   [user@client ~]$ ping <IP_address_of_the_database_server>
   ```
   If this step fails, check that your network and firewall settings allow direct communication between IdM clients and servers. See [Using and configuring firewalld](#).

3. Verify that the client can discover and contact the IdM LDAP server (for IdM users) or AD domain controller (for AD users) via the fully qualified host name.

   ```
   [user@client ~]$ dig -t SRV _ldap._tcp.example.com @<name_server>
   [user@client ~]$ ping <fully_qualified_host_name_of_the_server>
   ```
   If this step fails, check your Dynamic Name Service (DNS) settings, including the `/etc/resolv.conf` file. See [Configuring the order of DNS servers](#).
NOTE

By default, the SSSD service attempts to automatically discover LDAP servers and AD DCs through DNS service (SRV) records. Alternatively, you can restrict the SSSD service to use specific servers by setting the following options in the `sssd.conf` configuration file:

- `ipa_server = <fully_qualified_host_name_of_the_server>`
- `ad_server = <fully_qualified_host_name_of_the_server>`
- `ldap_uri = <fully_qualified_host_name_of_the_server>`

If you use these options, verify you can contact the servers listed in them.

4. Verify that the client can authenticate to the LDAP server and retrieve user information with `ldapsearch` commands.

   a. If your LDAP server is an IdM server, like `server.example.com`, retrieve a Kerberos ticket for the host and perform the database search authenticating with the host Kerberos principal:

      ```bash
      [user@client ~]$ kinit -t 'host/client.example.com@EXAMPLE.COM'
      [user@client ~]$ ldapsearch -LLL -Y GSSAPI -h server.example.com -b "dc=example,dc=com" uid=<user_name>
      ```

   b. If your LDAP server is an Active Directory (AD) Domain Controller (DC), like `server.ad.example.com`, retrieve a Kerberos ticket for the host and perform the database search authenticating with the host Kerberos principal:

      ```bash
      [user@client ~]$ kinit -t 'CLIENT$@AD.EXAMPLE.COM'
      [user@client ~]$ ldapsearch -LLL -Y GSSAPI -h server.ad.example.com -b "dc=example,dc=com" sAMAccountname=<user_name>
      ```

   c. If your LDAP server is a plain LDAP server, and you have set the `ldap_default_bind_dn` and `ldap_default_authtok` options in the `sssd.conf` file, authenticate as the same `ldap_default_bind_dn` account:

      ```bash
      [user@client ~]$ ldapsearch -xLLL -D "cn=ldap_default_bind_dn_value" -W -h ldapserver.example.com -b "dc=example,dc=com" uid=<user_name>
      ```

      If this step fails, verify that your database settings allow your host to search the LDAP server.

5. Since the SSSD service uses Kerberos encryption, verify you can obtain a Kerberos ticket as the user that is unable to log in.

   a. If your LDAP server is an IdM server:

      ```bash
      [user@client ~]$ kinit <user_name>
      ```

   b. If LDAP server database is an AD server:

      ```bash
      [user@client ~]$ kinit <user_name@AD.EXAMPLE.COM>
      ```
If this step fails, verify that your Kerberos server is operating properly, all servers have their times synchronized, and that the user account is not locked.

6. Verify you can retrieve user information on the command line.

```
[user@client ~]$ getent passwd <user_name>
[user@client ~]$ id <user_name>
```

If this step fails, verify that the SSSD service on the client can receive information from the user database:


b. Enable detailed logging in the SSSD service, collect debugging logs, and review the logs for indications to the source of the issue.

c. *(Optional)* Open a Red Hat Technical Support case and provide the troubleshooting information you have gathered.

7. Use the `sssct1` utility to verify the user is allowed to log in.

```
[user@client ~]$ sssct1 user-checks -a auth -s ssh <user_name>
```

If this step fails, verify your authorization settings, such as your PAM configuration, IdM HBAC rules, and IdM RBAC rules:

a. Review authorization errors in the `/var/log/secure` and `/var/log/messages` log files.

b. Enable detailed logging in the SSSD service, collect debugging logs, and review the logs for indications to the source of the issue.

c. *(Optional)* Open a Red Hat Technical Support case and provide the troubleshooting information you have gathered.

**Additional resources**

- Enabling detailed logging for SSSD in the `sssd.conf` file
- Enabling detailed logging for SSSD with the `sssct1` command
- Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM server
- Gathering debugging logs from the SSSD service to troubleshoot authentication issues with an IdM client

**8.5. SSSD LOG FILES AND LOGGING LEVELS**

Each SSSD service logs into its own log file in the `/var/log/sssd/` directory. For an IdM server in the `example.com` IdM domain, its log files might look like this:

```
[root@server ~]# ls -l /var/log/sssd/
total 620
-rw-------.  1 root root      0 Mar 29 09:21 krb5_child.log
-rw-------.  1 root root  14324 Mar 29 09:50 ldap_child.log
-rw-------.  1 root root 212870 Mar 29 09:50 sssd_example.com.log
```
8.5.1. SSSD log file purposes

**krb5_child.log**

Log file for the short-lived helper process involved in Kerberos authentication.

**ldap_child.log**

Log file for the short-lived helper process involved in getting a Kerberos ticket for the communication with the LDAP server.

**sssd_<example.com>.log**

For each domain section in the `sssd.conf` file, the SSSD service logs information about communication with the LDAP server to a separate log file. For example, in an environment with an IdM domain named `example.com`, the SSSD service logs its information in a file named `sssd_example.com.log`. If a host is directly integrated with an AD domain named `ad.example.com`, information is logged to a file named `sssd_ad.example.com.log`.

**NOTE**

If you have an IdM environment and a cross-forest trust with an AD domain, information about the AD domain is still logged to the log file for the IdM domain.

Similarly, if a host is directly integrated to an AD domain, information about any child domains is written in the log file for the primary domain.

**selinux_child.log**

Log file for the short-lived helper process that retrieves and sets SELinux information.

**sssd.log**

Log file for SSSD monitoring and communicating with its responder and backend processes.

**sssd_ifp.log**

Log file for the InfoPipe responder, which provides a public D-Bus interface accessible over the system bus.

**sssd_nss.log**

Log file for the Name Services Switch (NSS) responder that retrieves user and group information.

**sssd_pac.log**

Log file for the Microsoft Privilege Attribute Certificate (PAC) responder, which collects the PAC from AD Kerberos tickets and derives information about AD users from the PAC, which avoids requesting it directly from AD.

**sssd_pam.log**

Log file for the Pluggable Authentication Module (PAM) responder.

**sssd_ssh.log**

Log file for the SSH responder process.
8.5.2. SSSD logging levels

Setting a debug level also enables all debug levels below it. For example, setting the debug level at 6 also enables debug levels 0 through 5.

Table 8.1. SSSD logging levels

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>Fatal failures.</strong> Errors that prevent the SSSD service from starting up or cause it to terminate.</td>
</tr>
<tr>
<td>1</td>
<td><strong>Critical failures.</strong> Errors that do not terminate the SSSD service, but at least one major feature is not working properly.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Serious failures.</strong> Errors announcing that a particular request or operation has failed. <strong>This is the default debug log level.</strong></td>
</tr>
<tr>
<td>3</td>
<td><strong>Minor failures.</strong> Errors that cause the operation failures captured at level 2.</td>
</tr>
<tr>
<td>4</td>
<td><strong>Configuration</strong> settings.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Function</strong> data.</td>
</tr>
<tr>
<td>6</td>
<td>Trace messages for <strong>operation</strong> functions.</td>
</tr>
<tr>
<td>7</td>
<td>Trace messages for <strong>internal control</strong> functions.</td>
</tr>
<tr>
<td>8</td>
<td>Contents of <strong>function-internal</strong> variables.</td>
</tr>
<tr>
<td>9</td>
<td>Extremely <strong>low-level tracing</strong> information.</td>
</tr>
</tbody>
</table>

8.6. ENABLING DETAILED LOGGING FOR SSSD IN THE SSSD.CONF FILE

By default, the SSSD service only logs serious failures (debug level 2), but it does not log at the level of detail necessary to troubleshoot authentication issues.

To enable detailed logging persistently across SSSD service restarts, add the option `debug_level=<integer>` in each section of the `/etc/sssd/sssd.conf` configuration file, where the `<integer>` value is a number between 0 and 9. Debug levels up to 3 log larger failures, and levels 8 and higher provide a large number of detailed log messages. **Level 6** is a good starting point for debugging authentication issues.

**Prerequisites**

- You need the root password to edit the `sssd.conf` configuration file and restart the SSSD service.
**Procedure**

1. Open the `/etc/sssd/sssd.conf` file in a text editor.

2. Add the `debug_level` option to every section of the file, and set the debug level to the verbosity of your choice.

   ```
   [domain/example.com]
   debug_level = 6
   id_provider = ipa
   ...
   
   [sssd]
   debug_level = 6
   services = nss, pam, ifp, ssh, sudo
   domains = example.com
   
   [nss]
   debug_level = 6
   
   [pam]
   debug_level = 6
   
   [sudo]
   debug_level = 6
   
   [ssh]
   debug_level = 6
   
   [pac]
   debug_level = 6
   
   [ifp]
   debug_level = 6
   ```

3. Save and close the `sssd.conf` file.

4. Restart the SSSD service to load the new configuration settings.

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

**Additional resources**

- SSSD log files and logging levels

**8.7. ENABLING DETAILED LOGGING FOR SSSD WITH THE SSSCTL COMMAND**

By default, the SSSD service only logs serious failures (debug level 2), but it does not log at the level of detail necessary to troubleshoot authentication issues.

You can change the debug level of the SSSD service on the command line with the `ssscctl debug-level <integer>` command, where the `<integer>` value is a number between 0 and 9. Debug levels up to 3 log larger failures, and levels 8 and higher provide a large number of detailed log messages. Level 6 is a
good starting point for debugging authentication issues.

**Prerequisites**

- You need the root password to run the `sssctl` command.

**Procedure**

- Use the `sssctl debug-level` command to set the debug level of your choice to your desired verbosity.

  ```bash
  [root@server ~]# sssctl debug-level 6
  ```

**Additional resources**

- SSSD log files and logging levels

---

### 8.8. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM SERVER

If you experience issues when attempting to authenticate as an IdM user to an IdM server, enable detailed debug logging in the SSSD service on the server and gather logs of an attempt to retrieve information about the user.

**Prerequisites**

- You need the root password to run the `sssctl` command and restart the SSSD service.

**Procedure**

1. Enable detailed SSSD debug logging on the IdM server.

   ```bash
   [root@server ~]# sssctl debug-level 6
   ```

2. Invalidate objects in the SSSD cache for the user that is experiencing authentication issues, so you do not bypass the LDAP server and retrieve information SSSD has already cached.

   ```bash
   [root@server ~]# sssctl cache-expire -u idmuser
   ```

3. Minimize the troubleshooting dataset by removing older SSSD logs.

   ```bash
   [root@server ~]# sssctl logs-remove
   ```

4. Attempt to switch to the user experiencing authentication problems, while gathering timestamps before and after the attempt. These timestamps further narrow the scope of the dataset.

   ```bash
   [root@server sssd]# date; su idmuser; date
   Mon Mar 29 15:33:48 EDT 2021
   su: user idmuser does not exist
   Mon Mar 29 15:33:49 EDT 2021
   ```
5. (Optional) Lower the debug level if you do not wish to continue gathering detailed SSSD logs.

   [root@server ~]# sssctl debug-level 2

6. Review SSSD logs for information about the failed request. For example, reviewing the
   /var/log/sssd/sssd_example.com.log file shows that the SSSD service did not find the user in
   the cn=accounts,dc=example,dc=com LDAP subtree. This might indicate that the user does
   not exist, or exists in another location.

   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [dp_get_account_info_send] (0x0200):
   Got request for [0x1][BE_REQ_USER][name=idmuser@example.com] ...
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sdap_get_generic_ext_step] (0x0400):
   calling ldap_search_ext with [(uid=idmuser)(objectclass=posixAccount)(uid=)
   & (uidNumber=)](cn=accounts,dc=example,dc=com).
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sdap_get_generic_op_finished]
   (0x0400): Search result: Success(0), no errmsg set
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sdap_search_user_process] (0x0400):
   Search for users, returned 0 results.
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sysdb_search_by_name] (0x0400):
   No such entry
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sysdb_delete_user] (0x0400): Error: 2
   (No such file or directory)
   (Mon Mar 29 15:33:48 2021) [sssd[be@example.com]] [sysdb_search_by_name] (0x0400):
   No such entry
   (Mon Mar 29 15:33:49 2021) [sssd[be@example.com]]
   [ipa_id_get_account_info_orig_done] (0x0080): Object not found, ending request

7. If you are unable to determine the cause of the authentication issue:
   a. Collect the SSSD logs you recently generated.
      [root@server ~]# sssctl logs-fetch sssd-logs-Mar29.tar
   b. Open a Red Hat Technical Support case and provide:
      i. The SSSD logs: **sssd-logs-Mar29.tar**
      ii. The console output, including the time stamps and user name, of the request that
          corresponds to the logs:
          [root@server sssd]# date; id idmuser; date
          Mon Mar 29 15:33:48 EDT 2021
          id: 'idmuser': no such user
          Mon Mar 29 15:33:49 EDT 2021

8.9. GATHERING DEBUGGING LOGS FROM THE SSSD SERVICE TO
TROUBLESHOOT AUTHENTICATION ISSUES WITH AN IDM CLIENT

If you experience issues when attempting to authenticate as an IdM user to an IdM client, verify that you
 can retrieve user information on the IdM server. If you cannot retrieve the user information on an IdM
 server, you will not be able to retrieve it on an IdM client (which retrieves information from the IdM
 server).
After you have confirmed that authentication issues do not originate from the IdM server, gather SSSD debugging logs from both the IdM server and IdM client.

Prerequisites

- The user only has authentication issues on IdM clients, not IdM servers.
- You need the root password to run the `sssctl` command and restart the SSSD service.

Procedure

1. **On the client:** Open the `/etc/sssd/sssd.conf` file in a text editor.

2. **On the client:** Add the `ipa_server` option to the `[domain]` section of the file and set it to an IdM server. This avoids the IdM client autodiscovering other IdM servers, thus limiting this test to just one client and one server.

   ```
   [domain/example.com]
   ipa_server = server.example.com
   ...
   ```

3. **On the client:** Save and close the `sssd.conf` file.

4. **On the client:** Restart the SSSD service to load the configuration changes.

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

5. **On the server and client:** Enable detailed SSSD debug logging.

   ```
   [root@server ~]# sssctl debug-level 6
   [root@client ~]# sssctl debug-level 6
   ```

6. **On the server and client:** Invalidate objects in the SSSD cache for the user experiencing authentication issues, so you do not bypass the LDAP database and retrieve information SSSD has already cached.

   ```
   [root@server ~]# sssctl cache-expire -u idmuser
   [root@client ~]# sssctl cache-expire -u idmuser
   ```

7. **On the server and client:** Minimize the troubleshooting dataset by removing older SSSD logs.

   ```
   [root@server ~]# sssctl logs-remove
   [root@server ~]# sssctl logs-remove
   ```

8. **On the client:** Attempt to switch to the user experiencing authentication problems while gathering timestamps before and after the attempt. These timestamps further narrow the scope of the dataset.

   ```
   [root@client sssd]# date; su idmuser; date
   ```
Mon Mar 29 16:20:13 EDT 2021
su: user idmuser does not exist
Mon Mar 29 16:20:14 EDT 2021

9. *(Optional)* **On the server and client:** Lower the debug level if you do not wish to continue gathering detailed SSSD logs.

```
[root@server ~]# sssctl debug-level 0
[root@client ~]# sssctl debug-level 0
```

10. **On the server and client:** Review SSSD logs for information about the failed request.

   a. Review the request from the client in the client logs.
   b. Review the request from the client in the server logs.
   c. Review the result of the request in the server logs.
   d. Review the outcome of the client receiving the results of the request from the server.

11. If you are unable to determine the cause of the authentication issue:

   a. Collect the SSSD logs you recently generated on the IdM server and IdM client. Label them according to their hostname or role.

```
[root@server ~]# sssctl logs-fetch sssd-logs-server-Mar29.tar
[root@client ~]# sssctl logs-fetch sssd-logs-client-Mar29.tar
```

   b. Open a Red Hat Technical Support case and provide:

      i. The SSSD debug logs:

         A. `sssd-logs-server-Mar29.tar` from the server
         B. `sssd-logs-client-Mar29.tar` from the client

      ii. The console output, including the time stamps and user name, of the request that corresponds to the logs:

```
[root@client sssd]# date; su idmuser; date
Mon Mar 29 16:20:13 EDT 2021
su: user idmuser does not exist
Mon Mar 29 16:20:14 EDT 2021
```

**8.10. TRACKING CLIENT REQUESTS IN THE SSSD BACKEND**

SSSD processes requests asynchronously and as messages from different requests are added to the same log file, you can use the unique request identifier and client ID to track client requests in the backend logs. The unique request identifier is added to the debug logs in the form of `RID#<integer>` and the client ID in the form `[CID #<integer>]`. This allows you to isolate logs pertaining to an individual request, and you can track requests from start to finish across log files from multiple SSSD components.
Prerequisites

- You have enabled debug logging and a request has been submitted from an IdM client.
- You must have root privileges to display the contents of the SSSD log files.

Procedure

1. To review your SSSD log file, open the log file using the **less** utility. For example, to view the `/var/log/sssd/sssd_example.com.log`

   ```bash
   [root@server ~]# less /var/log/sssd/sssd_example.com.log
   ```

2. Review the SSSD logs for information about the client request.

   ```plaintext
   (2021-07-26 18:26:37): [be[testidm.com]] [dp_req_destructor] (0x0400): [RID#3] Number of active DP request: 0
   (2021-07-26 18:26:37): [be[testidm.com]] [dp_attach_req] (0x0400): [RID#4] Number of active DP request: 1
   ``

This sample output from an SSSD log file shows the unique identifiers **RID#3** and **RID#4** for two different requests.

However, a single client request to the SSSD client interface often triggers multiple requests in the backend and as a result it is not a 1-to-1 correlation between client request and requests in the backend. Though the multiple requests in the backend have different RID numbers, each initial backend request includes the unique client ID so an administrator can track the multiple RID numbers to the single client request.

The following example shows one client request **[sssd.nss CID #1]** and the multiple requests generated in the backend, **[RID#5]** to **[RID#13]**:

```plaintext
```
CHAPTER 9. CONFIGURING GLOBAL IDM SETTINGS USING ANSIBLE PLAYBOOKS

Using the Ansible `config` module, you can retrieve and set global configuration parameters for Identity Management (IdM).

This chapter includes the following sections:

- Retrieving IdM configuration using an Ansible playbook
- Configuring the IdM CA renewal server using an Ansible playbook
- Configuring the default shell for IdM users using an Ansible playbook

9.1. RETRIEVING IDM CONFIGURATION USING AN ANSIBLE PLAYBOOK

The following procedure describes how you can use an Ansible playbook to retrieve information about the current global IdM configuration.

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 the IdM server from which you want to retrieve the IdM configuration in the `[ipaserver]` section. For example, to instruct Ansible to retrieve the data from `server.idm.example.com`, enter:

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

2. Open the `/usr/share/doc/ansible-freeipa/playbooks/config/retrieve-config.yml` Ansible playbook file for editing:

   ```yaml
   ---
   - name: Playbook to handle global IdM configuration
     hosts: ipaserver
     become: no
     gather_facts: no

     tasks:
     - name: Query IPA global configuration
       ipaconfig:
         ipaadmin_password: Secret123
         register: serverconfig

         - debug:
             msg: "{{ serverconfig }}"
   ```
3. Adapt the file by changing the following:
   - The password of IdM administrator.
   - Other values, if necessary.

4. Save the file.

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

   ```bash
   $ ansible-playbook -v -i path_to_inventory_directory/inventory.file
   /usr/share/doc/ansible-freeipa/playbooks/config/retrieve-config.yml
   [...]
   TASK [debug]
   ok: [server.idm.example.com] => {
      "msg": {
         "ansible_facts": {
            "discovered_interpreter_
         },
         "changed": false,
         "config": {
            "ca_renewal_master_server": "server.idm.example.com",
            "configstring": [
               "AllowNTHash",
               "KDC:Disable Last Success"
            ],
            "defaultgroup": "ipausers",
            "defaultshell": "/bin/bash",
            "emaildomain": "idm.example.com",
            "enable_migration": false,
            "groupsearch": [  
               "cn",
               "description"
            ],
            "homedirectory": "/home",
            "maxhostname": "64",
            "maxusername": "64",
            "pac_type": [  
               "MS-PAC",
               "nfs:NONE"
            ],
            "pwdexpnotify": "4",
            "searchrecordslimit": "100",
            "searchtimelimit": "2",
            "selinuxusermapdefault": "unconfined_u:s0-s0:c0.c1023",
            "selinuxusermaporder": [  
               "guest_u:s0$xguest_u:s0$user_
            ],
            "usersearch": [  
               "uid",
               "givenname",
               "sn",
               "telephonenumber",
               "ou",
               "title"
            ]
         }
   }
9.2. CONFIGURING THE IDM CA RENEWAL SERVER USING AN ANSIBLE PLAYBOOK

In an Identity Management (IdM) deployment that uses an embedded certificate authority (CA), the CA renewal server maintains and renews IdM system certificates. It ensures robust IdM deployments.

For more details on the role of the IdM CA renewal server, see Using IdM CA renewal server.

The following procedure describes how you can use an Ansible playbook to configure the IdM CA renewal server.

Prerequisites

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible controller.

Procedure

1. Optional: Identify the current IdM CA renewal server:

   ```
   $ ipa config-show | grep 'CA renewal'
   IPA CA renewal master: server.idm.example.com
   ```

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

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

3. Open the /usr/share/doc/ansible-freeipa/playbooks/config/set-ca-renewal-master-server.yml Ansible playbook file for editing:

   ```
   ---
   - name: Playbook to handle global DNS configuration
     hosts: ipaserver
     become: no
     gather_facts: no

     tasks:
     - name: set ca_renewal_master_server
       ipaconfig:
         ipaadmind_password: SomeADMINpassword
         ca_renewal_master_server: carenewal.idm.example.com
   ```

4. Adapt the file by changing:

   - The password of IdM administrator set by the ipaadmind_password variable.
   - The name of the CA renewal server set by the ca_renewal_master_server variable.
5. Save the file.

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

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
/usr/share/doc/ansible-freeipa/playbooks/config/set-ca-renewal-master-server.yml
```

**Verification steps**

You can verify that the CA renewal server has been changed:

1. Log into `ipaserver` as IdM administrator:

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

2. Request the identity of the IdM CA renewal server:

```
$ ipa config-show | grep 'CA renewal'
IPA CA renewal master: carenewal.idm.example.com
```

The output shows the `carenewal.idm.example.com` server is the new CA renewal server.

---

**9.3. CONFIGURING THE DEFAULT SHELL FOR IDM USERS USING AN ANSIBLE PLAYBOOK**

The shell is a program that accepts and interprets commands. Several shells are available in Red Hat Enterprise Linux (RHEL), such as `bash`, `sh`, `ksh`, `zsh`, and others. Bash, or `/bin/bash`, is a popular shell on most Linux systems, and it is normally the default shell for user accounts on RHEL.

The following procedure describes how you can use an Ansible playbook to configure `sh`, an alternative shell, as the default shell for IdM users.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.

**Procedure**

1. Optional: Use the `retrieve-config.yml` Ansible playbook to identify the current shell for IdM users. See Retrieving IdM configuration using an Ansible playbook for details.

2. Create an inventory file, for example `inventory.file`, and define `ipaserver` in it:

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

3. Open the `/usr/share/doc/ansible-freeipa/playbooks/config/ensure-config-options-are-set.yml` Ansible playbook file for editing:
- name: Playbook to ensure some config options are set
  hosts: ipaserver
  become: true

  tasks:
  # Set defaultlogin and maxusername
  - ipaconfig:
    ipaadmin_password: Secret123
    defaultshell: /bin/bash
    maxusername: 64

4. Adapt the file by changing the following:

   - The password of IdM administrator set by the ipaadmin_password variable.
   - The default shell of the IdM users set by the defaultshell variable into /bin/sh.

5. Save the file.

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

   $ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/config/ensure-config-options-are-set.yml

Verification steps

You can verify that the default user shell has been changed by starting a new session in IdM:

1. Log into ipaserver as IdM administrator:

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

2. Display the current shell:

   [admin@server /]$ echo "$SHELL"
   /bin/sh

   The logged-in user is using the sh shell.

Additional resources

   - You can see sample Ansible playbooks for configuring global IdM settings and a list of possible variables in the README-config.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory.

   - You can see sample Ansible playbooks for various IdM configuration-related operations in the /usr/share/doc/ansible-freeipa/playbooks/config directory.
CHAPTER 10. 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

10.1. 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.

### 10.2. Adding Users Using the Command Line

You can add users 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_.$-]?<br>
```

**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.

### 10.3. ACTIVATING USERS USING THE COMMAND LINE

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

**Prerequisites**

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

**Procedure**
1. Open terminal and connect to the IdM server.

2. Activate the user account with the following command:

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

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

   ```
   $ ipa user-find
   ```

   This command lists all user accounts with details.

### 10.4. PRESERVING USERS USING THE COMMAND LINE

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

**Prerequisites**

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

**Procedure**

1. Open terminal and connect to the IdM server.

2. Preserve the user account with the following command:

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

   **NOTE**

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

### 10.5. 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.

### 10.6. RESTORING USERS USING THE COMMAND LINE

You can restore a preserved users to:

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

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

**Prerequisites**

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

**Procedure**

1. Open terminal and connect to the IdM server.

2. Activate the user account with the following command:

   ```
   $ ipa user-undel user_login
   -------------------------------
   Undeleted user account "user_login"
   ```
Alternatively, you can restore user accounts as staged:

```
$ ipa user-stage user_login
```

Staged user account "user_login"

Verification steps

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

```
$ ipa user-find
```

This command lists all user accounts with details.
Identity Management (IdM) provides several stages that can help you to manage various user work life situations:

Creating a user account

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

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

Activating a user account

Activating the account the first working day of the employee.

Disabling a user account

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

Enabling a user account

When the user returns, you will need to re-enable the account.

Preserving a user account

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

Restoring a user account

Two years later, the user is back and you need to restore the preserved account.

Deleting a user account

If the employee the employee is dismissed you will delete the account without a backup.

11.1. 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.

### 11.2. 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.
   For details, see *Accessing the IdM Web UI in a web browser*.

2. Go to **Users → Stage Users** tab.
   Alternatively, you can add the user account in the **Users → Active users** tab, 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.

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

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

### 11.3. ACTIVATING STAGE USERS IN THE IDM WEB UI

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

**Prerequisites**

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

1. Log in to the IdM Web UI. For details, see Accessing the IdM Web UI in a web browser.

2. Go to Users → Stage users tab.

3. Click the check-box of the user account you want to activate.

4. Click on the Activate button.

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.

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

**11.5. 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.

### 11.6. PRESERVING ACTIVE USERS IN THE IDM WEB UI

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

Preserve the user account if the employee leaves the company. If you want to disable user accounts for a couple of weeks or months (parental leave, for example), disable the account. For details, see Section 11.4, “Disabling user accounts in the Web UI”. The preserved accounts are not active and users cannot use them to access your internal network, however, the account stays in the database with all the data.

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

**NOTE**

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

**Prerequisites**

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

1. Log in to the IdM Web UI.
   For details, see Accessing the IdM Web UI in a web browser.

2. Go to Users → Active users tab.

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

4. Click on the Delete button.

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.

11.7. RESTORING USERS IN THE IDM WEB UI

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

Prerequisites

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

Procedure

1. Log in to the IdM Web UI.
   For details, see Accessing the IdM Web UI in a web browser.
2. Go to **Users → Preserved users** tab.

3. Click the check-box at the user accounts you want to restore.

4. Click on the **Restore** button.

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.

### 11.8. DELETING USERS IN THE IDM WEB UI

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

You can delete:

- **Active users** – the IdM Web UI offers you with the options:
  - Preserving users temporarily
    For details, see the **Preserving active users in the IdM Web UI**.
  - Deleting them permanently

- **Stage users** – you can just delete stage users permanently.

- **Preserved users** – you can delete preserved users permanently.

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

- **The Stage users** tab

- **The Preserved users** tab

**Prerequisites**

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

**Procedure**

1. Log in to the IdM Web UI.
   For details, see **Accessing the IdM Web UI in a web browser**.

2. Go to **Users → Active users** tab.
   Alternatively, you can delete the user account in the **Users → Stage users** or **Users → Preserved users**.
3. Click the **Delete** icon.

4. In the **Remove users** dialog box, switch the **Delete mode** radio button to **delete**.

5. Click on the **Delete** button.

The users accounts have been permanently deleted from IdM.
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.

### 12.1. 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.

### 12.2. ENSURING THE PRESENCE OF AN IDM USER USING AN ANSIBLE PLAYBOOK

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

**Prerequisites**

- You know the IdM administrator password.
- The ansible-freeipa package is installed on the Ansible controller.

**Procedure**

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

   ```bash
   [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:

```bash
$ 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:

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

  2. Request a Kerberos ticket for admin:

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

  3. Request information about **idm_user**:
12.3. 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
         - name: idm_user_3
           first: Cindy
           last: Acme
   ```
NOTE

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

3. Run the playbook:

```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.

### 12.4. 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:
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file path_to_playbooks_directory/ensure-users-present-jsonfile.yml

Verification steps

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

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

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

  2. Display information about **idm_user_1**:

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

    The user named **idm_user_1** is present in IdM.

### 12.5. 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 administator password.
- You have installed the **ansible-freeipa** package on the Ansible controller.

**Procedure**

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

   [ipaserver]
   server.idm.example.com

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

   ---
   - name: Playbook to handle users
     hosts: ipaserver
     become: true
Run the Ansible playbook specifying the playbook file and the inventory file:

```
$ ansible-playbook -v -i /path/to/inventory/directory/inventory.file /path/to/playbooks/directory/delete-users.yml
```

Verification steps

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

1. Log into `ipaserver` as administrator:

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

2. Request information about `idm_user_1`:

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

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

Additional resources

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

- You can also see sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/user` directory.
CHAPTER 13. 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

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>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](#).

### 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](image)

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

### 13.3. ADDING A USER GROUP USING IDM CLI

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

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

**Procedure**

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

  ```
  $ ipa group-add group_a
  ---------------------
  Added group "group_a"
  ---------------------
  Group name: group_a
  GID: 1133400009
  ```

  By default, `ipa group-add` adds a POSIX user group. To specify a different group type, add options to `ipa group-add`:
  
  - `--nonposix` to create a non-POSIX group
  - `--external` to create an external group
    
    For details on group types, see The different group types in IdM.

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

  **WARNING**

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

13.4. SEARCHING FOR USER GROUPS USING IDM CLI

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

**Procedure**

- Display all user groups by using the `ipa group-find` command. To specify a group type, add options to `ipa group-find`:
  
  - Display all POSIX groups using the `ipa group-find --posix` command.
  
  - Display all non-POSIX groups using the `ipa group-find --nonposix` command.
  
  - Display all external groups using the `ipa group-find --external` command.
    
    For more information on different group types, see The different group types in IdM.

13.5. DELETING A USER GROUP USING IDM CLI
This section describes how to delete a user group using IdM CLI. Note that deleting a group does not delete the group members from IdM.

**Prerequisites**

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

**Procedure**

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

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

**13.6. ADDING A MEMBER TO A USER GROUP USING IDM CLI**

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

**Prerequisites**

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

**Procedure**

- Add a member to a user group by using the `ipa group-add-member` command. Specify the type of member using these options:

  - **--users** adds an IdM user
  - **--external** adds a user that exists outside the IdM domain, in the format of `DOMAIN\user_name` or `user_name@domain`
  - **--groups** adds an IdM user group

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

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

  Members of `group_b` are now indirect members of `group_a`. 
IMPORTANT

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

NOTE

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

13.7. 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.

13.7.1. 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.

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

You can add a user without creating a user private group (UPG) even when UPGs are enabled on the system. This requires manually setting a GID for the new user. For details on why this is needed, see Section 13.7.1, “Users without a user private group”.
Procedure

- To prevent IdM from creating a UPG, add the `--noprivate` option to the `ipa user-add` command.
  Note that for the command to succeed, you must specify a custom GID. For example, to add a new user with GID 10000:

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

13.7.3. 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
   ```

Verification steps

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

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

13.7.4. Adding a user when user private groups are globally disabled
When user private groups (UPGs) are disabled globally, IdM does not assign a GID to a new user automatically. To successfully add a user, you must assign a GID manually or by using an automember rule. For details on why this is required, see Section 13.7.1, "Users without a user private group".

**Prerequisites**

- UPGs must be disabled globally for all users. For more information, see Disabling user private groups globally for all users

**Procedure**

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

**13.8. 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.

### 13.9. 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.
13.10. 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
   ```

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

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

Prerequisites

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

- You must have the name of the existing member manager user or group you are removing and the name of the group they are managing.

Procedure

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

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

  ```
  $ ipa group-remove-member-manager group_a --users=test
  Group name: group_a
  GID: 1133400009
  Membership managed by groups: group_admins
  ---------------------------
  Number of members removed 1
  ---------------------------
  ```
User **test** can no longer manage members of **group_a**.

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

  ```
  $ ipa group-remove-member-manager group_a --groups=group_admins
  Group name: group_a
  GID: 1133400009
  ---------------------------
  Number of members removed 1
  ---------------------------
  
  Group **group_admins** can no longer manage members of **group_a**.
  
  **NOTE**
  After you remove a member manager from a user group, the update may take some time to spread to all clients in your Identity Management environment.

**Verification steps**

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

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

- See `ipa group-remove-member-manager --help` for more details.
CHAPTER 14. 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

14.1. 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 14.1. 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](#).

### 14.2. 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 14.1. 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.

### 14.3. 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.

14.4. 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.

### 14.5. 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.
14.6. 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:
14.7. 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.

14.8. 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.

### 14.9. 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 15. 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

15.1. 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 15.1. 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.

### 15.2. 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 15.1. 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.
15.3. 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**

- For more information about ensuring the presence of user groups using Ansible, see the `/usr/share/doc/ansible-freeipa/README-group.md` Markdown file.

### 15.4. 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]
servers.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:

```bash
$ 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:

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

2. Display information about `managergroup1`:

   ```bash
   [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.

**15.5. 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:

   ```yaml
   ---
   - 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 16. 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

16.1. 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.

16.2. 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](#).

### 16.3. ADDING AN AUTOMEMBER RULE USING IDM CLI

This section describes adding an automember rule using the IdM CLI. For information about automember rules, see [Automember rules](#).

After adding an automember rule, you can add conditions to it using the procedure described in [Adding a condition to an automember rule](#).

**NOTE**

Existing entries are **not** affected by the new rule. If you want to change existing entries, see [Applying automember rules to existing entries using IdM CLI](#).

**Prerequisites**

- You must be logged in as the administrator. For details, see [Using kinit to log in to IdM manually](#).
- The target group of the new rule must exist in IdM.

**Procedure**

1. Enter the `ipa automember-add` command to add an automember rule.

2. When prompted, specify:

   - **Automember rule.** This is the target group name.
   - **Grouping Type.** This specifies whether the rule targets a user group or a host group. To target a user group, enter `group`. To target a host group, enter `hostgroup`. 


For example, to add an automember rule for a user group named `user_group`:

```
$ ipa automember-add
Automember Rule: user_group
Grouping Type: group

Added automember rule "user_group"
```

Verification steps

- You can display existing automember rules and conditions in IdM using Viewing existing automember rules using IdM CLI.

16.4. 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:

```
$ ipa automember-add-condition
Automember Rule: ad_users
Attribute Key: objectclass
Grouping Type: group
[Inclusive Regex]: ntUser
[Exclusive Regex]:
```

Verification steps

- You can display existing automember rules and conditions in IdM using Viewing existing automember rules using IdM CLI.

16.5. VIEWING EXISTING AUTOMEMBER RULES USING IDM CLI

This section describes how to view existing automember rules using the IdM CLI.

Prerequisites

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

Procedure

1. Enter the `ipa automember-find` command.
2. When prompted, specify the Grouping type:
   - To target a user group, enter `group`.
   - To target a host group, enter `hostgroup`.
   
   For example:

   ```
   $ ipa automember-find
   ```
16.6. 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`.

16.7. 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
-----------------------------
```

### 16.8. APPLYING AUTOMEMBER RULES TO EXISTING ENTRIES USING IDM CLI

Automember rules apply automatically to user and host entries created after the rules were added. They are not applied retroactively to entries that existed before the rules were added.

To apply automember rules to previously added entries, you have to manually rebuild automatic membership. Rebuilding automatic membership re-evaluates all existing automember rules and applies them either to all user or hosts entries, or to specific entries.

**NOTE**

Rebuilding automatic membership **does not** remove user or host entries from groups, even if the entries no longer match the group’s inclusive conditions. To remove them manually, see [Removing a member from a user group using IdM CLI](#) or [Removing IdM host group members using the CLI](#).

**Prerequisites**

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

**Procedure**

- To rebuild automatic membership, enter the `ipa automember-rebuild` command. Use the following options to specify the entries to target:
  
  - To rebuild automatic membership for all users, use the `--type=group` option:

    ```
    $ ipa automember-rebuild --type=group
    Automember rebuild task finished. Processed (9) entries.
    ```

  - To rebuild automatic membership for all hosts, use the `--type=hostgroup` option.
To rebuild automatic membership for a specified user or users, use the `--users=target_user` option:

```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=client.idm.example.com` option.

### 16.9. 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 17. 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

17.1. 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.

17.2. 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.

### 17.3. 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.

17.4. 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.

17.5. 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.

17.6. 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.

### 17.7. 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.

17.8. APPL YING 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.

17.8.1. Rebuilding automatic membership for all users or hosts

This section describes how to rebuild automatic membership for all user or host entries.

**Prerequisites**

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

**Procedure**

1. Select **Identity → Users or Hosts**.

2. Click **Actions → Rebuild auto membership**.

### 17.8.2. Rebuilding automatic membership for a single user or host only

This section describes how to rebuild automatic membership for a specific user or host entry.

**Prerequisites**

- You are logged in to the IdM Web UI.
- You must be a member of the **admins** group.

**Procedure**

1. Select **Identity → Users or Hosts**.

2. Click on the required user or host name.

3. Click **Actions → Rebuild auto membership**.
17.9. 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)

17.10. 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 18. 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).

18.1. 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.

18.2. 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:

```bash
$ 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"
```
18.3. 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
```

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.

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

18.4. DELETING SELF-SERVICE RULES USING THE CLI
This procedure describes deleting self-service access rules in IdM using the command-line interface (CLI).

**Prerequisites**

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

**Procedure**

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

For example:

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

**Verification steps**

- Use the `ipa selfservice-find` command to display all self-service rules. The rule you just deleted should be missing.
CHAPTER 19. 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).

19.1. 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.

19.2. 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.

### 19.3. 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.

![Self Service Permissions](image)

**Self Service Permission: User Self service**

#### General

<table>
<thead>
<tr>
<th>Self-service name</th>
<th>User Self service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes *</td>
<td></td>
</tr>
</tbody>
</table>

- audio
- carlicense
- departmentnumber
- destinationindicator
- employeeenum
- facsimiletelephonenumber
- gidnumber
- homedirectory
- homepostaladdress
- inetuserstatus
- businesscategory
- cn
- description
- displayname
- employeetype
- gecos
- givenname
- homephone
- inetuserhttpurl
- initials
3. The edit page only allows you to edit the list of attributes to you want to add or remove to the self-service rule. Select or deselect the appropriate check boxes.

4. Click the **Save** button to save your changes to the self-service rule.

19.4. 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](91).

**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 20. 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

20.1. 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.

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

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

Prerequisites

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

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

**Procedure**

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

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

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

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

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

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

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

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:
   ```
   $ ansible-playbook -v -i inventory selfservice-present-copy.yml
   ```
Additional resources

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

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

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

Prerequisites

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

Procedure

1. Navigate to the ~/MyPlaybooks/ directory:
   
   $ cd ~/MyPlaybooks/  

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

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

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

4. Adapt the file by setting the following variables in the ipaselfservice task section:
   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the self-service rule.
   - Set the state variable to absent.
This is the modified Ansible playbook file for the current example:

```yaml
---
- 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

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

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

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

**Prerequisites**

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

1. Navigate to the ~/MyPlaybooks/ directory:

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

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

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

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

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

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

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

   ```
   ---
   - name: Self-service member present
     hosts: ipaserver
     become: true
     tasks:
       - name: Ensure selfservice "Users can manage their own name details" member attribute surname is present
         ipaselfservice:
           ipaadmin_password: Secret123
           name: "Users can manage their own name details"
           attribute:
             - surname
           action: member
   ```

5. Save the file.

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

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

Additional resources

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

The /usr/share/doc/ansible-freeipa/playbooks/selfservice directory.

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

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

Prerequisites

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

Procedure

1. Navigate to the ~/MyPlaybooks/ directory:

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

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

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

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

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

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

   This is the modified Ansible playbook file for the current example:
---
- name: Self-service member absent
  hosts: ipaserver
  become: true
  tasks:
  - name: Ensure selfservice "Users can manage their own name details" member attributes
givenname and surname are absent
    ipaselfservice:
      ipaadmin_password: Secret123
      name: "Users can manage their own name details"
      attribute:
      - givenname
      - surname
      action: member
      state: absent

5. Save the file.

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

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

Additional resources

- For more information on the concept of self-service rules, see Self-service access control in IdM.

- For more sample Ansible playbooks that use the ipaselfservice module, see:
  - The README-selfservice.md file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the ipaselfservice variables.
  - The /usr/share/doc/ansible-freeipa/playbooks/selfservice directory.
CHAPTER 21. DELEGATING PERMISSIONS TO USER GROUPS TO MANAGE USERS USING IDM CLI

Delegation is one of the access control methods in IdM, along with self-service rules and role-based access control (RBAC). You can use delegation to assign permissions to one group of users to manage entries for another group of users.

This section covers the following topics:

- Delegation rules
- Creating a delegation rule using IdM CLI
- Viewing existing delegation rules using IdM CLI
- Modifying a delegation rule using IdM CLI
- Deleting a delegation rule using IdM CLI

21.1. 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.

21.2. CREATING A DELEGATION RULE USING IDM CLI

This section describes how to create a delegation rule using the IdM CLI.

Prerequisites

- You are logged in as a member of the admins group.

Procedure

- Enter the `ipa delegation-add` command. Specify the following options:
  - `--group`: the group who is being granted permissions to the entries of users in the user group.
  - `--membergroup`: the group whose entries can be edited by members of the delegation group.
  - `--permissions`: whether users will have the right to view the given attributes (read) and add or change the given attributes (write). If you do not specify permissions, only the write permission will be added.
  - `--attrs`: the attributes which users in the member group are allowed to view or edit.
For example:

```
$ ipa delegation-add "basic manager attributes" --permissions=read --permissions=write --attrs=businesscategory --attrs=departmentnumber --attrs=employeetype --attrs=employeenumber --group=managers --membergroup=employees
```

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

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

21.3. VIEWING EXISTING DELEGATION RULES USING IDM CLI

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

Prerequisites

- You are logged in as a member of the **admins** group.

Procedure

- Enter the `ipa delegation-find` command:

```
$ ipa delegation-find
```

```
1 delegation matched
```

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

```
Number of entries returned 1
```

21.4. 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
```

21.5. 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 22. 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

22.1. 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.

22.2. 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.
f. Click the **Add** button to save the new delegation rule.

### 22.3. VIEWING EXISTING DELEGATION RULES USING IDM WEBUI

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

**Prerequisites**
22.4. 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.

### 22.5. 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 23. 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

23.1. 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.

23.2. 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.

### 23.3. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS PRESENT

The following procedure describes how to use an Ansible playbook to define privileges for a new IdM delegation rule and ensure its presence. In the example, the new `basic manager attributes` delegation rule grants the `managers` group the ability to read and write the following attributes for members of the `employees` group:

- `businesscategory`
- `departmentnumber`
- `employeenumber`
- `employeetype`

**Prerequisites**

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

1. Navigate to the ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks/

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

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

3. Open the delegation-present-copy.yml Ansible playbook file for editing.

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

   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the new delegation rule.
   - Set the permission variable to a comma-separated list of permissions to grant: read and write.
   - Set the attribute variable to a list of attributes the delegated user group can manage: businesscategory, departmentnumber, employeenumber, and employeetype.
   - Set the group variable to the name of the group that is being given access to view or modify attributes.
   - Set the membergroup variable to the name of the group whose attributes can be viewed or modified.

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

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

     tasks:
     - name: Ensure delegation "basic manager attributes" is present
       ipadelegation:
         ipaadmin_password: Secret123
         name: "basic manager attributes"
         permission: read, write
         attribute:
           - businesscategory
           - departmentnumber
           - employeenumber
           - employeetype
         group: managers
         membergroup: employees

5. Save the file.

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

- For more information on the concept of a delegation rule, see Delegation rules.
- For more sample Ansible playbooks that use the ipadelegation module, see:
  - The README-delegation.md file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the ipadelegation variables.
  - The /usr/share/doc/ansible-freeipa/playbooks/ipadelegation directory.

23.4. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE IS ABSENT

The following procedure describes how to use an Ansible playbook to ensure a specified delegation rule is absent from your IdM configuration. The example below describes how to make sure the custom basic manager attributes delegation rule does not exist in IdM.

Prerequisites

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

Procedure

1. Navigate to the ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks

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

   $ cp /usr/share/doc/ansible-freeipa/playbooks/delegation/delegation-present.yml delegation-absent-copy.yml

3. Open the delegation-absent-copy.yml Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the ipadelegation task section:
   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the delegation rule.
- Set the state variable to absent.

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

```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:

   ```bash
   $ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-absent-copy.yml
   ```

Additional resources

- For more information on the concept of a delegation rule, see Delegation rules.
- For more sample Ansible playbooks that use the ipadelegation module, see:
  - The README-delegation.md file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the ipadelegation variables.
  - The /usr/share/doc/ansible-freeipa/playbooks/ipadelegation directory.

23.5. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE HAS SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that a delegation rule has specific settings. You can use this playbook to modify a delegation role you have previously created. In the example, you ensure the basic manager attributes delegation rule only has the departmentnumber member attribute.

Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
  - You are using Ansible version 2.8 or later.
  - You have installed the ansible-freeipa package.
  - You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
  - Your Ansible inventory file is located in the ~/MyPlaybooks/ directory.
• The **basic manager attributes** delegation rule exists in IdM.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:
   ```bash
   $ cd ~/MyPlaybooks/
   ```

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

```yaml
---
- 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:
   ```bash
   $ ansible-playbook -v -i ~/MyPlaybooks/inventory delegation-member-present-copy.yml
   ```

Additional resources

• For more information on the concept of a delegation rule, see [Delegation rules](#).
For more sample Ansible playbooks that use the `ipadelegation` module, see:

- The README-delegation.md file available in the `/usr/share/doc/ansible-freeipa/` directory. This file also contains the definitions of the `ipadelegation` variables.
- The `/usr/share/doc/ansible-freeipa/playbooks/ipadelegation` directory.

## 23.6. USING ANSIBLE TO ENSURE THAT A DELEGATION RULE DOES NOT HAVE SPECIFIC ATTRIBUTES

The following procedure describes how to use an Ansible playbook to ensure that a delegation rule does not have specific settings. You can use this playbook to make sure a delegation role does not grant undesired access. In the example, you ensure the basic manager attributes delegation rule does not have the `employeenumber` and `employeetype` member attributes.

### Prerequisites

- You know the IdM administrator password.
- You have configured an Ansible control node that meets the following requirements:
  - You are using Ansible version 2.8 or later.
  - You have installed the `ansible-freeipa` package.
  - You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server where you are configuring these options.
  - Your Ansible inventory file is located in the `~/.MyPlaybooks/` directory.
- The basic manager attributes delegation rule exists in IdM.

### Procedure

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

   ```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`.

Red Hat Enterprise Linux 9.0 Beta Configuring and managing Identity Management
• Set the state variable to absent.

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

```yaml
---
- 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:

```bash
$ ansible-playbook -v -i ~/.MyPlaybooks/inventory delegation-member-absent-copy.yml
```

Additional resources

• For more information on the concept of a delegation rule, see Delegation rules.

• For more sample Ansible playbooks that use the ipadelegation module, see:
  
  - The README-delegation.md file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the ipadelegation variables.
  
  - The /usr/share/doc/ansible-freeipa/playbooks/ipadelegation directory.
CHAPTER 24. MANAGING ROLE-BASED ACCESS CONTROLS IN IDM USING THE CLI

This chapter introduces role-based access control in Identity Management (IdM) and describes the following operations in the command-line interface (CLI):

- Managing permissions
- Managing privileges
- Managing roles

24.1. 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.

24.1.1. 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.

### 24.1.2. 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.

### 24.1.3. 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.
24.1.4. 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.

24.1.5. Predefined roles in Identity Management

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

**Table 24.1. Predefined Roles in Identity Management**

<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>
24.2. 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:

```
$ ipa permission-add "dns admin" --attrs=description --attrs=automountKey
```

```
$ ipa permission-add "dns admin" --attrs={description,automountKey}
```

The attributes provided with `--attrs` must exist and be allowed attributes for the given object type, otherwise the command fails with schema syntax errors.

- `--type` defines the entry object type to which the permission applies, such as user, host, or service. Each type has its own set of allowed attributes. For example:

```
$ ipa permission-add "manage service" --right=all --type=service --attrs=krbprincipalkey -
-attrs=krbprincipalname --attrs=managedby
```

- `--subtree` gives a subtree entry; the filter then targets every entry beneath this subtree entry. Provide an existing subtree entry; `--subtree` does not accept wildcards or non-existent domain names (DNs). Include a DN within the directory. Because IdM uses a simplified, flat directory tree structure, `--subtree` can be used to target some types of entries, like automount locations, which are containers or parent entries for other configuration. For example:

```
$ ipa permission-add "manage automount locations" --
subtree="ldap://ldap.example.com:389/cn=automount,dc=example,dc=com" --right=write
--attrs=automountmapname --attrs=automountkey --attrs=automountInformation
```

- `--filter` uses an LDAP filter to identify which entries the permission applies to. IdM automatically checks the validity of the given filter. The filter can be any valid LDAP filter, for example:

```
$ ipa permission-add "manage Windows groups" --filter="(!(objectclass=posixgroup))" --
right=write --attrs=description
```

- `--memberof` sets the target filter to members of the given group after checking that the group exists. For example, to let the users with this permission modify the login shell of members of the engineers group:

```
$ ipa permission-add ManageShell --right="write" --type=user --attr=loginshell --
memberof=engineers
```

- `--targetgroup` sets target to the specified user group after checking that the group exists. For example, to let those with the permission write the member attribute in the engineers group (so they can add or remove members):
$ ipa permission-add ManageMembers --right="write" --
  subtree=cn=groups,cn=accounts,dc=example,dc=test --attr=member --
  targetgroup=engineers

• Optionally, you can specify a target domain name (DN):
  • --target specifies the DN to apply the permission to. Wildcards are accepted.
  • --targetto specifies the DN subtree where an entry can be moved to.
  • --targetfrom specifies the DN subtree from where an entry can be moved.

24.3. COMMAND OPTIONS FOR EXISTING PERMISSIONS

Use the following variants to modify existing permissions as needed:

• To edit existing permissions, use the ipa permission-mod command. You can use the same command options as for adding permissions.

• To find existing permissions, use the ipa permission-find command. You can use the same command options as for adding permissions.

• To view a specific permission, use the ipa permission-show command.
  The --raw argument shows the raw 389-ds ACI that is generated. For example:

  $ ipa permission-show <permission> --raw

• The ipa permission-del command deletes a permission completely.

Additional resources

For further details about the ipa permission commands, refer to the ipa man page and the ipa help command.

24.4. MANAGING IDM PRIVILEGES IN THE CLI

This section describes how to manage Identity Management (IdM) privileges using the command-line interface (CLI).

Prerequisites

• Administrator privileges for managing IdM or the User Administrator role.

• An active Kerberos ticket. For details, see Using kinit to log in to IdM manually.

• Existing permissions. For details about permissions, see Managing IdM permissions in the CLI.

Procedure

1. Add privilege entries using the ipa privilege-add command
   For example, to add a privilege named managing filesystems with a description:

   $ ipa privilege-add "managing filesystems" --desc="for filesystems"
2. Assign the required permissions to the privilege group with the `privilege-add-permission` command. For example, to add the permissions named `managing automount` and `managing ftp services` to the `managing filesystems` privilege:

```
$ ipa privilege-add-permission "managing filesystems" --permissions="managing automount"
--permissions="managing ftp services"
```

### 24.5. COMMAND OPTIONS FOR EXISTING PRIVILEGES

Use the following variants to modify existing privileges as needed:

- To modify existing privileges, use the `ipa privilege-mod` command.
- To find existing privileges, use the `ipa privilege-find` command.
- To view a specific privilege, use the `ipa privilege-show` command.
- The `ipa privilege-remove-permission` command removes one or more permissions from a privilege.
- The `ipa privilege-del` command deletes a privilege completely.

Additional resources

For further details about the `ipa privilege` commands, refer to the `ipa man` page and the `ipa help` command.

### 24.6. MANAGING IDM ROLES IN THE CLI

This section describes how to manage Identity Management (IdM) roles using the command-line interface (CLI).

**Prerequisites**

- Administrator privileges for managing IdM or the User Administrator role.
- An active Kerberos ticket. For details, see Using kinit to log in to IdM manually.
- Existing privileges. For details about privileges, see Managing IdM privileges in the CLI.

**Procedure**

1. Add new role entries using the `ipa role-add` command:

   ```
   $ ipa role-add --desc="User Administrator" useradmin
   ------------------------
   Added role "useradmin"
   ------------------------
   Role name: useradmin
   Description: User Administrator
   ```

2. Add the required privileges to the role using the `ipa role-add-privilege` command:
$ ipa role-add-privilege --privileges="user administrators" useradmin
Role name: useradmin
Description: User Administrator
Privileges: user administrators
-----------------------------
Number of privileges added 1
-----------------------------

3. Add the required members to the role using the `ipa role-add-member` command. Allowed member types are: users, groups, hosts and hostgroups. For example, to add the group named `useradmins` to the previously created `useradmin` role:

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

**24.7. COMMAND OPTIONS FOR EXISTING ROLES**

Use the following variants to modify existing roles as needed:

- To modify existing roles, use the `ipa role-mod` command.
- To find existing roles, use the `ipa role-find` command.
- To view a specific role, use the `ipa role-show` command.
- To remove a member from the role, use the `ipa role-remove-member` command.
- The `ipa role-remove-privilege` command removes one or more privileges from a role.
- The `ipa role-del` command deletes a role completely.

**Additional resources**

For further details about the `ipa role` commands, refer to the `ipa` man page and the `ipa help` command.
CHAPTER 25. 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

25.1. 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.

25.1.1. 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.

### 25.1.2. 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.

25.1.3. 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.
25.1.4. 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.

25.1.5. Predefined roles in Identity Management

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

Table 25.1. Predefined Roles in Identity Management

<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>
25.2. 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:
4. Select the appropriate Bind rule type:

- **permission** is the default permission type, granting access through privileges and roles
- **all** specifies that the permission applies to all authenticated users
- **anonymous** specifies that the permission applies to all users, including unauthenticated users
NOTE

It is not possible to add permissions with a non-default bind rule type to privileges. You also cannot set a permission that is already present in a privilege to a non-default bind rule type.

5. Choose the rights to grant with this permission in Granted rights.

6. Define the method to identify the target entries for the permission:

- **Type** specifies an entry type, such as user, host, or service. If you choose a value for the Type setting, a list of all possible attributes which will be accessible through this ACI for that entry type appears under Effective Attributes. Defining Type sets Subtree and Target DN to one of the predefined values.

- **Subtree** (required) specifies a subtree entry; every entry beneath this subtree entry is then targeted. Provide an existing subtree entry, as Subtree does not accept wildcards or non-existent domain names (DNs). For example: `cn=automount,dc=example,dc=com`

- **Extra target filter** uses an LDAP filter to identify which entries the permission applies to. The filter can be any valid LDAP filter, for example: `(!(objectclass=posixgroup))` IdM automatically checks the validity of the given filter. If you enter an invalid filter, IdM warns you about this when you attempt to save the permission.

- **Target DN** specifies the domain name (DN) and accepts wildcards. For example: `uid=*,cn=users,cn=accounts,dc=com`

- **Member of group** sets the target filter to members of the given group. After you specify the filter settings and click Add, IdM validates the filter. If all the permission settings are correct, IdM will perform the search. If some of the permissions settings are incorrect, IdM will display a message informing you about which setting is set incorrectly.

7. Add attributes to the permission:

- If you set Type, choose the Effective attributes from the list of available ACI attributes.

- If you did not use Type, add the attributes manually by writing them into the Effective attributes field. Add a single attribute at a time; to add multiple attributes, click Add to add another input field.

    **IMPORTANT**

    If you do not set any attributes for the permission, then the permissions includes all attributes by default.

8. Finish adding the permissions with the Add buttons at the bottom of the form:

- Click the Add button to save the permission and go back to the list of permissions.

- Alternatively, you can save the permission and continue adding additional permissions in the same form by clicking the Add and Add another button

- The Add and Edit button enables you to save and continue editing the newly created permission.
9. *Optional.* You can also edit the properties of an existing permission by clicking its name from the list of permissions to display the **Permission settings** page.

10. *Optional.* If you need to remove an existing permission, click the **Delete** button once you ticked the check box next to its name in the list, to display The **Remove permissions** dialog.

**NOTE**

Operations on default managed permissions are restricted: the attributes you cannot modify are disabled in the IdM Web UI and you cannot delete the managed permissions completely.

However, you can effectively disable a managed permission that has a bind type set to permission, by removing the managed permission from all privileges.
For example, to let those with the permission write the member attribute in the engineers group (so they can add or remove members):

![Add permission form](image)

### 25.3. 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](#).

---

195
- 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:

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

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

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:

![Privilege: New Privilege](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.

25.4. 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:

   ![Role-Based Access Control Menu]

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 26. PREPARING YOUR ENVIRONMENT FOR MANAGING IDM USING ANSIBLE PLAYBOOKS

As a system administrator managing Identity Management (IdM), when working with Red Hat Ansible Engine, it is good practice to do the following:

- Create a subdirectory dedicated to Ansible playbooks in your home directory, for example ~/MyPlaybooks.
- Copy and adapt sample Ansible playbooks from the /usr/share/doc/ansible-freeipa/* and /usr/share/doc/rhel-system-roles/* directories and subdirectories into your ~/MyPlaybooks directory.
- Include your inventory file in your ~/MyPlaybooks directory.

Using this practice, you can find all your playbooks in one place and you can run your playbooks without invoking root privileges.

NOTE

You only need root privileges on the managed nodes to execute the ipaserver, ipareplica, ipaclient and ipabackup ansible-freeipa roles. These roles require privileged access to directories and the dnf software package manager.

This section describes how to create the ~/MyPlaybooks directory and configure it so that you can use it to store and run Ansible playbooks.

Prerequisites

- You have installed an IdM server on your managed nodes, server.idm.example.com and replica.idm.example.com.
- You have configured DNS and networking so you can log in to the managed nodes, server.idm.example.com and replica.idm.example.com, directly from the control node.
- You know the IdM admin password.

Procedure

1. Create a directory for your Ansible configuration and playbooks in your home directory:

   $ mkdir ~/MyPlaybooks/

2. Change into the ~/MyPlaybooks/ directory:

   $ cd ~/MyPlaybooks

3. Create the ~/MyPlaybooks/ansible.cfg file with the following content:

   [defaults]
   inventory = /home/your_username/MyPlaybooks/inventory
4. Create the `/MyPlaybooks/inventory` file with the following content:

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

[us]
replica.idm.example.com

[ipaserver:children]
  eu
  us
```

This configuration defines two host groups, eu and us, for hosts in these locations. Additionally, this configuration defines the ipaserver host group, which contains all hosts from the eu and us groups.

5. [Optional] Create an SSH public and private key. To simplify access in your test environment, do not set a password on the private key:

```
$ ssh-keygen
```

6. Copy the SSH public key to the IdM admin account on each managed node:

```
$ ssh-copy-id admin@server.idm.example.com
$ ssh-copy-id admin@replica.idm.example.com
```

These commands require that you enter the IdM admin password.

**Additional resources**

- For more information on installing an IdM server using an Ansible playbook, see [Installing an Identity Management server using an Ansible playbook](#).
- For an overview of available formats for an Ansible inventory file including examples, see [How to build your inventory](#).
CHAPTER 27. 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

### 27.1. 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.

### 27.2. 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.
27.3. 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.

27.4. 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.

27.5. PREDEFINED ROLES IN IDENTITY MANAGEMENT

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

**Table 27.1. Predefined Roles in Identity Management**

<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>
</tbody>
</table>
### Role | Privilege | Description
--- | --- | ---
**IT Security Specialist** | Netgroups Administrators, HBAC Administrator, Sudo Administrator | Responsible for managing security policy such as host-based access controls, sudo rules

**IT Specialist** | Host Administrators, Host Group Administrators, Service Administrators, Automount Administrators | Responsible for managing hosts

**Security Architect** | Delegation Administrator, Replication Administrators, Write IPA Configuration, Password Policy Administrator | Responsible for managing the Identity Management environment, creating trusts, creating replication agreements

**User Administrator** | User Administrators, Group Administrators, Stage User Administrators | Responsible for creating users and groups

---

**27.6. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE WITH PRIVILEGES IS PRESENT**

To exercise more granular control over role-based access (RBAC) to resources in Identity Management (IdM) than the default roles provide, create a custom role.

The following procedure describes how to use an Ansible playbook to define privileges for a new IdM custom role and ensure its presence. In the example, the new `user_and_host_administrator` role contains a unique combination of the following privileges that are present in IdM by default:

- Group Administrators
- User Administrators
- Stage User Administrators
- Group Administrators

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks` directory.

**Procedure**

1. Navigate to the `~/MyPlaybooks` directory:
2. Make a copy of the `role-member-user-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-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:

   ```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
           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:

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

Additional resources
For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see Encrypting content with Ansible Vault.

For more information on the concept of role in IdM, see Roles in IdM.

For more sample Ansible playbooks that use the iparole module, see:
- The README-role file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the iparole variables.
- The /usr/share/doc/ansible-freeipa/playbooks/iparole directory.

27.7. USING ANSIBLE TO ENSURE AN IDM RBAC ROLE IS ABSENT

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure the absence of an obsolete role so that no administrator assigns it to any user accidentally.

The following procedure describes how to use an Ansible playbook to ensure a role is absent. The example below describes how to make sure the custom user_and_host_administrator role does not exist in IdM.

Prerequisites

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the ~/<MyPlaybooks> directory.

Procedure

1. Navigate to the ~/<MyPlaybooks> directory:

   ```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 ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the role.
   - Ensure that the state variable is set to absent.

   This is the modified Ansible playbook file for the current example:
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
  - iparole:
      ipaadmin_password: Secret123
      name: user_and_host_administrator
      state: absent

5. Save the file.

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

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

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see Encrypting content with Ansible Vault.

- For more information on the concept of role in IdM, see Roles in IdM.

- For more sample Ansible playbooks that use the iparole module, see:
  - The README-role Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the iparole variables.
  - The /usr/share/doc/ansible-freeipa/playbooks/iparole directory.

27.8. USING ANSIBLE TO ENSURE THAT A GROUP OF USERS IS ASSIGNED TO AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to assign a role to a specific group of users, for example junior administrators.

The following example describes how to use an Ansible playbook to ensure the built-in IdM RBAC helpdesk role is assigned to junior_sysadmins.

Prerequisites

- You know the IdM administrator password.

- You have installed the ansible-freeipa package on the Ansible control node.

- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.

- Your Ansible inventory file is located in the ~/.<MyPlaybooks>/ directory.

Procedure
1. Navigate to the `~/<MyPlaybooks>` directory:

   ```
   $ cd ~/<MyPlaybooks/>
   ```

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

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

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

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

   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the role you want to assign.
   - Set the `group` variable to the name of the group.
   - Set the `action` variable to `member`.

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

   ```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:

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

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the `iparole` module, see the README-role Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `iparole` variables.
For more sample Ansible playbooks that use the `iparole` module, see the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

27.9. USING ANSIBLE TO ENSURE THAT SPECIFIC USERS ARE NOT ASSIGNED TO AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure that an RBAC role is not assigned to specific users after they have, for example, moved to different positions within the company.

The following procedure describes how to use an Ansible playbook to ensure that the users named `user_01` and `user_02` are not assigned to the `helpdesk` role.

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>/` directory.

Procedure

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

   ```
   $ cd ~/<MyPlaybooks>/
   ```

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

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

3. Open the `role-member-user-absent-copy.yml` Ansible playbook file for editing.

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

   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the role you want to assign.
   - Set the `user` list to the names of the users.
   - Set the `action` variable to `member`.
   - Set the `state` variable to `absent`.

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

   ```yaml
   ---
   - name: Playbook to manage IPA role with members.
     hosts: ipaserver
   ```
become: yes
gather_facts: no

tasks:
  - iparole:
      ipaadmin_password: Secret123
      name: helpdesk
      user
      - user_01
      - user_02
      action: member
      state: absent

5. Save the file.

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

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

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).

- For more information on the concept of role in IdM, see [Roles in IdM](#).

- For more sample Ansible playbooks that use the iparole module, see the README-role Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the iparole variables.

- For more sample Ansible playbooks that use the iparole module, see the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

### 27.10. USING ANSIBLE TO ENSURE A SERVICE IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control (RBAC) in Identity Management (IdM), you may want to ensure that a specific service that is enrolled into IdM is a member of a particular role. The following example describes how to ensure that the custom web_administrator role can manage the HTTP service that is running on the `client01.idm.example.com` server.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/<MyPlaybooks>` directory.
- The web_administrator role exists in IdM.
The HTTP/client01.idm.example.com@IDM.EXAMPLE.COM service exists in IdM.

Procedure

1. Navigate to the ~/<MyPlaybooks>/ directory:

   $ cd ~/<MyPlaybooks>/

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

   $ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-service-present-absent.yml role-member-service-present-copy.yml

3. Open the role-member-service-present-copy.yml Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the iparole task section:
   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the role you want to assign.
   - Set the service list to the name of the service.
   - Set the action variable to member.

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

   ```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:

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

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see Encrypting content with Ansible Vault.
For more information on the concept of role in IdM, see Roles in IdM.

For more sample Ansible playbooks that use the iparole module, see the README-role Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the iparole variables.

For more sample Ansible playbooks that use the iparole module, see the /usr/share/doc/ansible-freeipa/playbooks/iparole directory.

27.11. USING ANSIBLE TO ENSURE A HOST IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control in Identity Management (IdM), you may want to ensure that a specific host or host group is associated with a specific role. The following example describes how to ensure that the custom web_administrator role can manage the client01.idm.example.com IdM host on which the HTTP service is running.

Prerequisites

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the ~/<MyPlaybooks>/ directory.
- The web_administrator role exists in IdM.
- The client01.idm.example.com host exists in IdM.

Procedure

1. Navigate to the ~/<MyPlaybooks>/ directory:

   $ cd ~/<MyPlaybooks>/

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

   $ cp /usr/share/doc/ansible-freeipa/playbooks/role/role-member-host-present.yml role-member-host-present-copy.yml

3. Open the role-member-host-present-copy.yml Ansible playbook file for editing.

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

   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the role you want to assign.
   - Set the host list to the name of the host.

This is the modified Ansible playbook file for the current example:
---
- name: Playbook to manage IPA role with members.
  hosts: ipaserver
  become: yes
  gather_facts: no

  tasks:
  - iparole:
      ipaadmin_password: Secret123
      name: web_administrator
      host:
        - client01.idm.example.com
      action: member

5. Save the file.

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

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

Additional resources

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
- For more information on the concept of role in IdM, see [Roles in IdM](#).
- For more sample Ansible playbooks that use the iparole module, see the README-role Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the iparole variables.
- For more sample Ansible playbooks that use the iparole module, see the `/usr/share/doc/ansible-freeipa/playbooks/iparole` directory.

### 27.12. USING ANSIBLE TO ENSURE A HOST GROUP IS A MEMBER OF AN IDM RBAC ROLE

As a system administrator managing role-based access control in Identity Management (IdM), you may want to ensure that a specific host or host group is associated with a specific role. The following example describes how to ensure that the custom web_administrator role can manage the web_servers group of IdM hosts on which the HTTP service is running.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the ~/<MyPlaybooks>/ directory.
The web_administrator role exists in IdM.

The web_servers host group exists in IdM.

### Procedure

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

   ```bash
   $ 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:

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

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

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

   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the role you want to assign.
   - Set the `hostgroup` list to the name of the hostgroup.

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

   ```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

- For more information on how to use Ansible Vault to store a password in a separate file or to encrypt it as a variable in the playbook file, see [Encrypting content with Ansible Vault](#).
● For more information on the concept of role in IdM, see Roles in IdM.

● For more sample Ansible playbooks that use the iparole module, see the README-role Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the iparole variables.

● For more sample Ansible playbooks that use the iparole module, see the /usr/share/doc/ansible-freeipa/playbooks/iparole directory.
CHAPTER 28. 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.

28.1. USING ANSIBLE TO ENSURE A CUSTOM IDM RBAC PRIVILEGE IS PRESENT

To have a fully-functioning custom privilege in Identity Management (IdM) role-based access control (RBAC), you need to proceed in stages:

1. Create a privilege with no permissions attached.
2. Add permissions of your choice to the privilege.

The following procedure describes how to create an empty privilege using an Ansible playbook so that you can later add permissions to it. The example describes how to create a privilege named `full_host_administration` that is meant to combine all IdM permissions related to host administration.

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.

Procedure

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

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

   ```sh
   $ 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:

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

28.2. USING ANSIBLE TO ENSURE MEMBER PERMISSIONS ARE PRESENT IN A CUSTOM IDM RBAC PRIVILEGE

To have a fully-functioning custom privilege in Identity Management (IdM) role-based access control (RBAC), you need to proceed in stages:

1. Create a privilege with no permissions attached.

2. Add permissions of your choice to the privilege.

The following procedure describes how to use an Ansible playbook to add permissions to a privilege created in the previous step. The example describes how to add all IdM permissions related to host administration to a privilege named `full_host_administration`. By default, the permissions are distributed between the `Host Enrollment`, `Host Administrators` and `Host Group Administrator` privileges.

---

**Prerequisites**
You know the IdM administrator password.

You have installed the **ansible-freeipa** package on the Ansible control node.

You have created an **Ansible inventory file** with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.

Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.

The **full_host_administration** privilege exists. For information on how to create a privilege using Ansible, see Using Ansible to ensure a custom IdM RBAC privilege is present.

**Procedure**

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

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

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

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

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

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

   - Adapt the `name` of the task to correspond to your use case.
   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the privilege.
   - Set the `permission` list to the names of the permissions that you want to include in the privilege.
   - Make sure that the `action` variable is set to `member`.

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

```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"
```
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
```

28.3. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE DOES NOT INCLUDE A PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control. The following procedure describes how to use an Ansible playbook to remove a permission from a privilege. The example describes how to remove the Request Certificates ignoring CA ACLs permission from the default Certificate Administrators privilege because, for example, the administrator considers it a security risk.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks` directory.

**Procedure**

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

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

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

3. Open the `privilege-member-absent-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipaprivilege` task section:
   - Adapt the `name` of the task to correspond to your use case.
   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the privilege.
   - Set the `permission` list to the names of the permissions that you want to remove from the privilege.
   - Make sure that the `action` variable is set to `member`.
   - Make sure that the `state` variable is set to `absent`.

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

   ```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:

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

---

28.4. USING ANSIBLE TO RENAME A CUSTOM IDM RBAC PRIVILEGE

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control.
The following procedure describes how to rename a privilege because, for example, you have removed a few permissions from it. As a result, the name of the privilege is no longer accurate. In the example, the administrator renames a `full_host_administration` privilege to `limited_host_administration`.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the `~/MyPlaybooks/` directory.
- The `full_host_administration` privilege exists. For more information on how to add a privilege, see Using Ansible to ensure a custom IdM RBAC privilege is present.

**Procedure**

1. Navigate to the `~/MyPlaybooks/` directory:
   ```bash
   $ 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 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 `ipadmin_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:
   ```yaml
   ---
   - name: Rename a privilege
     hosts: ipaserver
     become: true
   ```
6. Rename the task in the playbook, for example:
   ```yaml
   [...]
   tasks:
   - name: Ensure the full_host_administration privilege is renamed to
     ```
limited_host_administration
ipaprivilege:
[...]

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

---
- name: Rename a privilege
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure the full_host_administration privilege is renamed to limited_host_administration
    ipaprivilege:
      ipaadmin_password: Secret123
      name: full_host_administration
      rename: limited_host_administration
      state: renamed

7. Save the file.

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

   $ ansible-playbook -v -i inventory rename-privilege.yml

28.5. USING ANSIBLE TO ENSURE AN IDM RBAC PRIVILEGE IS ABSENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control. The following procedure describes how to use an Ansible playbook to ensure that an RBAC privilege is absent. The example describes how to ensure that the CA administrator privilege is absent. As a result of the procedure, the admin administrator becomes the only user capable of managing certificate authorities in IdM.

Prerequisites

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible control node.
- You have created an Ansible inventory file with the fully-qualified domain name (FQDN) of the IdM server on which you want to do the configuring.
- Your Ansible inventory file is located in the ~/.MyPlaybooks/ directory.

Procedure

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

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

   ```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:

   ```yaml
   [...]
   tasks:
   - 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:
         ipaadmin_password: Secret123
         name: CA administrator
         state: absent
   
   6. Save the file.

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

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

Additional resources

- For more information on the concept of a privilege in IdM RBAC, see [Privileges in IdM](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/multipleservers/rhel_9_multipleservers(idm_rbac))
- For more information on the concept of a permission in IdM RBAC, see [Permissions in IdM](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/multipleservers/rhel_9_multipleservers(idm_rbac))
- For more sample Ansible playbooks that use the `ipaprivilege` module, see the `README-privilege` file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `ipaprivilege` variables.
For more sample Ansible playbooks that use the \texttt{ipaprivilege} module, see the \\
\texttt{/usr/share/doc/ansible-freeipa/playbooks/ipaprivilege} directory.
CHAPTER 29. 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.

29.1. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS PRESENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure a permission is present in IdM so that it can be added to a privilege. The example describes how to ensure the following target state:

- The MyPermission permission exists.
- The MyPermission permission can only be applied to hosts.
- A user granted a privilege that contains the permission can do all of the following possible operations on an entry:
  - Write
  - Read
  - Search
  - Compare
  - Add
  - Delete

Prerequisites
• You know the IdM administrator password.

• You have installed the ansible-freeipa package on the Ansible control node.

• The example assumes that you have created and configured the ~/MyPlaybooks/ directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the ~/MyPlaybooks/ directory:

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

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

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

3. Open the permission-present-copy.yml Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the ipapermission task section:
   - Adapt the name of the task to correspond to your use case.
   - Set the ipaadmin_password variable to the password of the IdM administrator.
   - Set the name variable to the name of the permission.
   - Set the object_type variable to host.
   - Set the right variable to all.

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

```
---
- name: Permission present example
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure that the "MyPermission" permission is present
      ipapermission:
        ipaadmin_password: Secret123
        name: MyPermission
        object_type: host
        right: all
```

5. Save the file.

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

```
$ ansible-playbook -v -i inventory permission-present-copy.yml
```
29.2. USING ANSIBLE TO ENSURE AN RBAC PERMISSION WITH AN ATTRIBUTE IS PRESENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure a permission is present in IdM so that it can be added to a privilege. The example describes how to ensure the following target state:

- The **MyPermission** permission exists.
- The **MyPermission** permission can only be used to add hosts.
- A user granted a privilege that contains the permission can do all of the following possible operations on a host entry:
  - Write
  - Read
  - Search
  - Compare
  - Add
  - Delete
- The host entries created by a user that is granted a privilege that contains the **MyPermission** permission can have a **description** value.

**NOTE**

The type of attribute that you can specify when creating or modifying a permission is not constrained by the IdM LDAP schema. However, specifying, for example, **attrs:** `car_licence` if the **object_type** is **host** later results in the **ipa: ERROR: attribute "car-license" not allowed** error message when you try to exercise the permission and add a specific car licence value to a host.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the **ansible-freeipa** package on the Ansible control node.
- The example assumes that you have created and configured the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

**Procedure**

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

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

```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

- For more information on the IdM schema, see User and group schema in Linux Domain Identity, Authentication and Policy Guide in RHEL 7.

## 29.3. USING ANSIBLE TO ENSURE AN RBAC PERMISSION IS ABSENT

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).
The following procedure describes how to use an Ansible playbook to ensure a permission is absent in IdM so that it cannot be added to a privilege.

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- The example assumes that you have created and configured the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:
   ```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
   ```
29.4. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS A MEMBER OF AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure that an attribute is a member of an RBAC permission in IdM. As a result, a user with the permission can create entries that have the attribute.

The example describes how to ensure that the host entries created by a user with a privilege that contains the `MyPermission` permission can have `gecos` and `description` values.

**NOTE**

The type of attribute that you can specify when creating or modifying a permission is not constrained by the IdM LDAP schema. However, specifying, for example, `attrs: car_licence` if the `object_type` is `host` later results in the `ipa: ERROR: attribute "car-license" not allowed` error message when you try to exercise the permission and add a specific car licence value to a host.

**Prerequisites**

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

**Procedure**

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

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

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

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

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

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

   - Adapt the `name` of the task to correspond to your use case.
   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the permission.
   - Set the `attrs` list to the `description` and `gecos` variables.
Make sure the action variable is set to member.

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

```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:

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

29.5. USING ANSIBLE TO ENSURE AN ATTRIBUTE IS NOT A MEMBER OF AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control (RBAC).

The following procedure describes how to use an Ansible playbook to ensure that an attribute is not a member of an RBAC permission in IdM. As a result, when a user with the permission creates an entry in IdM LDAP, that entry cannot have a value associated with the attribute.

The example describes how to ensure the following target state:

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

Prerequisites

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

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

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

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

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

3. Open the `permission-member-absent-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipapermission` task section:
   - Adapt the `name` of the task to correspond to your use case.
   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the permission.
   - Set the `attrs` variable to `description`.
   - Set the `action` variable to `member`.
   - Make sure the `state` variable is set to `absent`

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

   ```
   ---
   - name: Permission absent example
     hosts: ipaserver
     become: true

     tasks:
       - name: Ensure that an attribute is not a member of "MyPermission"
         ipapermission:
           ipaadmin_password: Secret123
           name: MyPermission
           attrs: description
           action: member
           state: absent
   ```

5. Save the file.

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

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

29.6. USING ANSIBLE TO RENAME AN IDM RBAC PERMISSION

As a system administrator of Identity Management (IdM), you can customize the IdM role-based access control.
The following procedure describes how to use an Ansible playbook to rename a permission. The example describes how to rename `MyPermission` to `MyNewPermission`.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- The example assumes that you have created and configured the `~/.MyPlaybooks/` directory as a central location to store copies of sample playbooks.
- The `MyPermission` exists in IdM.
- The `MyNewPermission` does not exist in IdM.

**Procedure**

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

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

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

   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/permission/permission-renamed.yml permission-renamed-copy.yml
   ```

3. Open the `permission-renamed-copy.yml` Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the `ipapermission` task section:
   
   - Adapt the `name` of the task to correspond to your use case.
   - Set the `ipaadmin_password` variable to the password of the IdM administrator.
   - Set the `name` variable to the name of the permission.

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

   ```
   ---
   - name: Permission present example
     hosts: ipaserver
     become: true

     tasks:
     - name: Rename the "MyPermission" permission
       ipapermission:
         ipaadmin_password: Secret123
         name: MyPermission
         rename: MyNewPermission
         state: renamed
   ```

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

   ```
   $ ansible-playbook -v -i inventory permission-renamed-copy.yml
   ```

29.7. ADDITIONAL RESOURCES

- For more information on the concept of a permission in IdM RBAC, see Permissions in IdM.

- For more information on the concept of a privilege in IdM RBAC, see Privileges in IdM.

- For more sample Ansible playbooks that use the `ipapermission` module, see the README-permission file available in the `/usr/share/doc/ansible-freeipa` directory. The file also contains the definitions of the `ipapermission` variables.

- For more sample Ansible playbooks that use the `ipapermission` module, see the `/usr/share/doc/ansible-freeipa/playbooks/ipapermission` directory.
CHAPTER 30. 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

30.1. ID VIEWS

An ID view in Identity Management (IdM) is an IdM client-side view specifying the following information:

- New values for centrally defined POSIX user or group attributes
- The client host or hosts on which the new values apply.

An ID view contains one or more overrides. An override is a specific replacement of a centrally defined POSIX attribute value.

You can only define an ID view for an IdM client centrally on IdM servers. You cannot configure client-side overrides for an IdM client locally.

For example, you can use ID views to achieve the following goals:

- Define different attribute values for different environments. For example, you can allow the IdM administrator or another IdM user to have different home directories on different IdM clients: you can configure /home/encrypted/username to be this user’s home directory on one IdM client and /dropbox/username on another client. Using ID views in this situation is convenient as alternatively, for example, changing fallback_homedir, override_homedir or other home directory variables in the client’s /etc/sssd/sssd.conf file would affect all users. See Adding an ID view to override an IdM user home directory on an IdM client for an example procedure.

- Replace a previously generated attribute value with a different value, such as overriding a user’s
UID. This ability can be useful when you want to achieve a system-wide change that would otherwise be difficult to do on the LDAP side, for example make 1009 the UID of an IdM user. IdM ID ranges, which are used to generate an IdM user UID, never start as low as 1000 or even 10000. If a reason exists for an IdM user to impersonate a local user with UID 1009 on all IdM clients, you can use ID views to override the UID of this IdM user that was generated when the user was created in IdM.

**IMPORTANT**

You can only apply ID views to IdM clients, not to IdM servers.

Additional resources

- You can also use ID views in environments involving Active Directory (AD). For details, see the ID Views and Migrating Existing Environments to Trust chapter in the Windows integration guide.
- You can also configure ID views for hosts that are not part of a centralized identity management domain. For details, see the SSSD Client-side Views chapter in the System-level authentication guide.

### 30.2. 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.

### 30.3. ATTRIBUTES AN ID VIEW CAN OVERRIDE

ID views consist of user and group ID overrides. The overrides define the new POSIX attribute values.

User and group ID overrides can define new values for the following POSIX attributes:

**User attributes**

- Login name (uid)
- GECOS entry (gecos)
- UID number (uidNumber)
- GID number (gidNumber)
- Login shell (`loginShell`)
- Home directory (`homeDirectory`)
- SSH public keys (`ipaSshPubkey`)
- Certificate (`userCertificate`)

Group attributes
- Group name (`cn`)
- Group GID number (`gidNumber`)

30.4. GETTING HELP FOR ID VIEW COMMANDS

You can get help for commands involving Identity Management (IdM) ID views on the IdM command-line interface (CLI).

Prerequisites
- You have obtained a Kerberos ticket for an IdM user.

Procedure
- To display all commands used to manage ID views and overrides:
  
  ```
  $ ipa help idviews
  ID Views
  Manage ID Views
  IPA allows to override certain properties of users and groups[...]
  [...] Topic commands:
  idoverridegroup-add Add a new Group ID override
  idoverridegroup-del Delete a Group ID override
  [...]```

- To display detailed help for a particular command, add the `--help` option to the command:
  
  ```
  $ ipa idview-add --help
  Usage: ipa [global-options] idview-add NAME [options]
  Add a new ID View.
  Options:
  -h, --help show this help message and exit
  --desc=STR Description
  [...]```

30.5. 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`:

   ```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 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:

   ```bash
   $ 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:

   ```bash
   $ 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:

   ```bash
   $ ipa idview-apply example_for_host1 --hosts=host1.idm.example.com
   ---------------------------
   ```

   CHAPTER 30. USING AN ID VIEW TO OVERRIDE A USER ATTRIBUTE VALUE ON AN IDM CLIENT

   243
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`:
30.6. MODIFYING AN IDM ID VIEW

An ID view in Identity Management (IdM) overrides a POSIX attribute value associated with a specific IdM user. This section describes how to modify an existing ID view. Specifically, it describes how to modify an ID view to enable the user named idm_user to use the /home/user_1234/ directory as the user home directory instead of /home/idm_user/ on the host1.idm.example.com IdM client.

Prerequisites

- You have root access to host1.idm.example.com.
- You are logged in as a user with the required privileges, for example admin.
- You have an ID view configured for idm_user that applies to the host1 IdM client.

Procedure

1. As root, create the directory that you want idm_user to use on host1.idm.example.com as the user home directory:

   [root@host1 ~]# mkdir /home/user_1234/

2. Change the ownership of the directory:

   [root@host1 ~]# chown idm_user:idm_user /home/user_1234/

3. Display the ID view, including the hosts to which the ID view is currently applied. To display the ID view named example_for_host1:

   $ ipa idview-show example_for_host1 --all
   dn: cn=example_for_host1,cn=views,cn=accounts,dc=idm,dc=example,dc=com
   ID View Name: example_for_host1
   User object override: idm_user
   Hosts the view applies to: host1.idm.example.com
   objectclass: ipaIDView, top, nsContainer

   The output shows that the ID view currently applies to host1.idm.example.com.

4. Modify the user override of the example_for_host1 ID view. To override the user home directory:

   - Enter the ipa idoverrideuser-add command
   - Add the name of the ID view
   - Add the user name, also called the anchor
   - Add the --homedir option:
$ ipa idoverrideuser-mod example_for_host1 idm_user --
homedir=/home/user_1234

-----------------------------
Modified an User ID override "idm_user"
-----------------------------
Anchor to override: idm_user
User login: user_1234
Home directory: /home/user_1234/

For a list of the available options, run ipa idoverrideuser-mod --help.

5. To apply the new configuration to the host1.idm.example.com system immediately:
   a. SSH to the system as root:

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

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

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

Verification steps

1. SSH to host1 as idm_user:

   ```
   [root@r8server ~]# ssh idm_user@host1.idm.example.com
   Password:
   ```
   [user_1234@host1 ~]$

2. Print the working directory:

   ```
   [user_1234@host1 ~]# pwd
   /home/user_1234/
   ```

30.7. ADDING AN ID VIEW TO OVERRIDE AN IDM USER HOME DIRECTORY ON AN IDM CLIENT

An ID view in Identity Management (IdM) overrides a POSIX attribute value associated with a specific IdM user. This section describes how to create an ID view that applies to idm_user on an IdM client named host1 to enable the user to use the /home/user_1234/ directory as the user home directory instead of /home/idm_user/.

Prerequisites

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

Procedure

1. As root, create the directory that you want idm_user to use on host1.idm.example.com as the user home directory:

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

2. Change the ownership of the directory:

   [root@host1 /]# chown idm_user:idm_user /home/user_1234/

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

   $ ipa idview-add example_for_host1
   ------------
   Added ID View "example_for_host1"
   ------------
   ID View Name: example_for_host1

4. Add a user override to the example_for_host1 ID view. To override the user home directory:

   • Enter the ipa idoverrideuser-add command
   • Add the name of the ID view
   • Add the user name, also called the anchor
   • Add the --homedir option:

   $ ipa idoverrideuser-add example_for_host1 idm_user --homedir=/home/user_1234
   ------------
   Added User ID override "idm_user"
   ------------
   Anchor to override: idm_user
   Home directory: /home/user_1234/

5. Apply example_for_host1 to the host1.idm.example.com host:

   $ ipa idview-apply example_for_host1 --hosts=host1.idm.example.com
   ------------
   Applied ID View "example_for_host1"
   ------------
   hosts: host1.idm.example.com
   ------------
   Number of hosts the ID View was applied to: 1
The `ipa idview-apply` command also accepts the `--hostgroups` option. The option applies the ID view to hosts that belong to the specified host group, but does not associate the ID view with the host group itself. Instead, the `--hostgroups` option expands the members of the specified host group and applies the `--hosts` option individually to every one of them.

This means that if a host is added to the host group in the future, the ID view does not apply to the new host.

6. To apply the new configuration to the `host1.idm.example.com` system immediately:
   a. SSH to the system as root:
      ```
      $ ssh root@host1
      Password:
      ``
   b. Clear the SSSD cache:
      ```
      root@host1 ~]# sss_cache -E
      ```
   c. Restart the SSSD daemon:
      ```
      root@host1 ~]# systemctl restart sssd
      ```

Verification steps

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

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

30.8. APPLYING AN ID VIEW TO AN IDM HOST GROUP

The `ipa idview-apply` command accepts the `--hostgroups` option. However, the option acts as a one-time operation that applies the ID view to hosts that currently belong to the specified host group, but does not dynamically associate the ID view with the host group itself. The `--hostgroups` option expands the members of the specified host group and applies the `--hosts` option individually to every one of them.

If you add a new host to the host group later, you must apply the ID view to the new host manually, using the `ipa idview-apply` command with the `--hosts` option.
Similarly, if you remove a host from a host group, the ID view is still assigned to the host after the removal. To unapply the ID view from the removed host, you must run the `ipa idview-unapply id_view_name --hosts=name_of_the_removed_host` command.

This section describes how to achieve the following goals:

1. How to create a host group and add hosts to it.
2. How to apply an ID view to the host group.
3. How to add a new host to the host group and apply the ID view to the new host.

**Prerequisites**

- Ensure that the ID view you want to apply to the host group exists in IdM. For example, to create an ID view to override an IdM user login name on a specific IdM client, see Using an ID view to override the login name of an IdM user on a specific host.

**Procedure**

1. Create a host group and add hosts to it:
   a. Create a host group. For example, to create a host group named `baltimore`:

   ```
   [root@server ~]# ipa hostgroup-add --desc="Baltimore hosts" baltimore
   ----------------------------------
   Added hostgroup "baltimore"
   ----------------------------------
   Host-group: baltimore
   Description: Baltimore hosts
   ```

   b. Add hosts to the host group. For example, to add the `host102` and `host103` to the `baltimore` host group:

   ```
   [root@server ~]# ipa hostgroup-add-member --hosts={host102,host103} baltimore
   Host-group: baltimore
   Description: Baltimore hosts
   Member hosts: host102.idm.example.com, host103.idm.example.com
   -------------------------
   Number of members added 2
   -------------------------
   ```

2. Apply an ID view to the hosts in the host group. For example, to apply the `example_for_host1` ID view to the `baltimore` host group:

   ```
   [root@server ~]# ipa idview-apply --hostgroups=baltimore
   ID View Name: example_for_host1
   ----------------------------------
   Applied ID View "example_for_host1"
   ----------------------------------
   hosts: host102.idm.example.com, host103.idm.example.com
   ---------------------------------------------
   Number of hosts the ID View was applied to: 2
   ---------------------------------------------
   ```
3. Add a new host to the host group and apply the ID view to the new host:

   a. Add a new host to the host group. For example, to add the `somehost.idm.example.com` host to the `baltimore` host group:

      ```
      [root@server ~]# ipa hostgroup-add-member --hosts=somehost.idm.example.com baltimore
      Host-group: baltimore
      Description: Baltimore hosts
      Member hosts: host102.idm.example.com,
      host103.idm.example.com,somehost.idm.example.com
      -------------------------
      Number of members added 1
      -------------------------
      ```

   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 31. 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.

31.1. ID RANGES

ID numbers are divided into ID ranges. Keeping separate numeric ranges for individual servers and replicas eliminates the chance that an ID number issued for an entry is already used by another entry on another server or replica.

Note that there are two distinct types of ID ranges:

- The IdM ID range, which is assigned during the installation of the first server. This range cannot be modified after it is created. However, if you need to, you can create a new IdM ID range in addition to the original one. For more information, see Automatic ID ranges assignment and Adding a new IdM ID range.

- The Distributed Numeric Assignment (DNA) ID ranges, which can be modified by the user. These have to fit within an existing IdM ID range. For more information, see Adjusting DNA ID ranges manually.

Replicas can also have a next DNA ID range assigned. A replica uses its next range when it runs out of IDs in its current range. Next ranges are assigned automatically when a replica is deleted or you can set them manually.

The ranges are updated and shared between the server and replicas by the DNA plug-in, as part of the back end 389 Directory Server instance for the domain.

The DNA range definition is set by two attributes: the server’s next available number (the low end of the DNA range) and its maximum value (the top end of the DNA range). The initial bottom range is set during the plug-in instance configuration. After that, the plug-in updates the bottom value. Breaking the available numbers into ranges allows the servers to continually assign numbers without overlapping with each other.

31.2. AUTOMATIC ID RANGES ASSIGNMENT

By default, an IdM ID range is automatically assigned during the IdM server installation. The ipa-server-install command randomly selects and assigns a range of 200,000 IDs from a total of 10,000 possible ranges. Selecting a random range in this way significantly reduces the probability of conflicting IDs in case you decide to merge two separate IdM domains in the future.

NOTE

This IdM ID range cannot be modified after it is created. You can only manually adjust the Distributed Numeric Assignment (DNA) ID ranges, using the commands described in Adjusting DNA ID ranges manually. A DNA range matching the IdM ID range is automatically created during installation.

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

**IMPORTANT**

When you install a replica, it **does not** immediately receive an ID range. A replica receives an ID range the first time the DNA plug-in is used, for example when you first add a user. Until then, the replica has no ID range defined.

If the initial server stops functioning before the replica requests a DNA ID range from it, the replica is unable to contact the server to request the ID range. Attempting to add a new user on the replica then fails. In such situations, you can find out what ID range is assigned to the disabled server, and **assign an ID range to the replica manually**.

### 31.3. 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:

```bash
# 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
----------
```
31.4. ADDING A NEW IDM ID RANGE

In some cases, you may want to create a new IdM ID range in addition to the original one; for example, when a replica has run out of IDs and the original IdM ID range is depleted.

**IMPORTANT**

Adding a new IdM ID range does not create new DNA ID ranges automatically. You need to assign new DNA ID ranges manually as needed. For more information on how to do this, see Adjusting DNA ID ranges manually.

**Procedure**

1. To create a new IdM ID range, use the `ipa idrange-add` command. You need to specify the new range name, the first ID number of the range and the range size:

   ```
   # ipa idrange-add IDM.EXAMPLE.COM_new_range --base-id=1000000 --range-size=200000
   ------------------------------------------
   Added ID range "IDM.EXAMPLE.COM_new_range"
   ------------------------------------------
   Range name: IDM.EXAMPLE.COM_new_range
   First Posix ID of the range: 1000000
   Number of IDs in the range: 200000
   Range type: local domain range
   ```

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, it takes some time for SSSD to notice the new ID range. More specifically, it notices the range when it updates the domain list and other configuration data stored on the IdM server.

**Verification steps**

- You can check if the new range is set correctly by using the `ipa idrange-find` command:

  ```
  # ipa idrange-find
  2 ranges matched
  ----------------
  Range name: IDM.EXAMPLE.COM_id_range
  First Posix ID of the range: 882200000
  Number of IDs in the range: 200000
  ```
31.5. REMOVING AN ID RANGE AFTER REMOVING A TRUST TO AD

If you have removed a trust between your IdM and Active Directory (AD) environments, you might want to remove the ID range associated with it.

**WARNING**

IDs allocated to ID ranges associated with trusted domains might still be used for ownership of files and directories on systems enrolled into IdM.

If you remove the ID range that corresponds to an AD trust that you have removed, you will not be able to resolve the ownership of any files and directories owned by AD users.

**Prerequisites**

- You have removed a trust to an AD environment.

**Procedure**

1. Display all the ID ranges that are currently in use:

   ```bash
   [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:

   ```bash
   [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.

   ```bash
   [root@server ~]# systemctl restart sssd
   ```

**Additional resources**

- For steps on removing a trust to AD from the command line, see [Removing the trust using the command line](#).
For steps on removing a trust to AD from the IdM Web UI, see Removing the trust using the IdM Web UI.

### 31.6. DISPLAYING CURRENTLY ASSIGNED DNA ID RANGES

You can display both the currently active Distributed Numeric Assignment (DNA) ID range on a server, as well as its next DNA range if it has one assigned.

**Procedure**

- To display which DNA ID ranges are configured for the servers in the topology, use the following commands:
  - `ipa-replica-manage dnarange-show` displays the current DNA ID range that is set on all servers or, if you specify a server, only on the specified server, for example:
    ```
    # ipa-replica-manage dnarange-show
    serverA.example.com: 1001-1500
    serverB.example.com: 1501-2000
    serverC.example.com: No range set
    # ipa-replica-manage dnarange-show serverA.example.com
    serverA.example.com: 1001-1500
    ```
  - `ipa-replica-manage dnanextrange-show` displays the next DNA ID range currently set on all servers or, if you specify a server, only on the specified server, for example:
    ```
    # ipa-replica-manage dnanextrange-show
    serverA.example.com: 2001-2500
    serverB.example.com: No on-deck range set
    serverC.example.com: No on-deck range set
    # ipa-replica-manage dnanextrange-show serverA.example.com
    serverA.example.com: 2001-2500
    ```

### 31.7. AUTOMATIC DNA ID RANGE EXTENSION

When you are deleting a RHEL 8 Identity Management (IdM) replica, the ID range that was assigned to the replica is not, by default, transferred to any other remaining replica in the topology. If you want to use the ID range, you must transfer it to another replica manually.

**Procedure**

In this example we want to delete `serverB.example.com` and preserve its ID range by transferring it to `serverC.example.com`.

1. To check what range is currently assigned to `serverB.example.com`, use the `ipa-replica-manage dnarange-show` command.
    ```
    # ipa-replica-manage dnarange-show
    serverA.example.com: 1001-1500
    serverB.example.com: 1501-2000
    serverC.example.com: No range set
    ```
The output shows that serverB.example.com has the 1501-2000 ID range assigned to it.

**NOTE**
If you want to check the next ID range set for a server, use `ipa-replica-manage dnanextrange-show` instead.

2. Assign the ID range to serverC.example.com using the `ipa-replica-manage dnarange-set` command.

```bash
# ipa-replica-manage dnarange-set serverC.example.com 1501-2000
```

**Verification steps**

1. Delete serverB.example.com:

```bash
# ipa-server-del serverB.example.com
```

2. Verify which DNA ID range is set for the serverC.example.com:

```bash
# ipa-replica-manage dnarange-show serverC.example.com
serverC.example.com: 1501-2000
```

### 31.8. MANUAL DNA ID RANGE ADJUSTMENT

In certain situations, it is necessary to manually adjust a Distributed Numeric Assignment (DNA) ID range, for example when:

- A replica has run out of IDs and the IdM ID range is depleted
  A replica has exhausted the DNA ID range that was assigned to it, and requesting additional IDs failed because no more free IDs are available in the IdM range.

  To solve this situation, extend the DNA ID range assigned to the replica. You can do this in two ways:
  - Shorten the DNA ID range assigned to a different replica, then assign the newly available values to the depleted replica.
  - Create a new IdM ID range, then set a new DNA ID range for the replica within this created IdM range.
    For information on how to create a new IdM ID range, see [Adding a new IdM ID range](#).

- A replica stopped functioning
  A replica's DNA ID range is not automatically retrieved when the replica dies and needs to be deleted, which means the DNA ID range previously assigned to the replica becomes unavailable.
  You want to recover the DNA ID range and make it available for other replicas.

  If you want to recover a DNA ID range belonging to a replica that stopped functioning and assign it to another server, you first need to find out what the ID range values are, before manually assigning that range to a different server. Also, to avoid duplicate UIDs or GIDs, make sure that no ID value from the recovered range was previously assigned to a user or group; you can do this by examining the UIDs and GIDs of existing users and groups.
You can manually adjust a DNA ID range for a replica using the commands in Adjusting DNA ID ranges manually.

**NOTE**

If you assign a new DNA ID range, the UIDs of the already existing entries on the server or replica stay the same. This does not pose a problem because even if you change the current DNA ID range, IdM keeps a record of what ranges were assigned in the past.

### 31.9. ADJUSTING DNA ID RANGES MANUALLY

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

When adjusting a DNA ID range manually, make sure that the newly adjusted range is included in the IdM ID range; you can check this using the `ipa idrange-find` command. Otherwise, the command will fail.

**IMPORTANT**

Be careful not to create overlapping ID ranges. If any of the ID ranges you assign to servers or replicas overlap, it could result in two different servers assigning the same ID value to different entries.

**Prerequisites**

- Optional. If you are recovering a DNA ID range from a non-functioning replica, first find the ID range using the commands described in Displaying currently assigned DNA ID ranges.

**Procedure**

- To define the current DNA ID range for a specified server, use the `ipa-replica-manage dnarange-set`:

  ```bash
  # ipa-replica-manage dnarange-set serverA.example.com 1250-1499
  ```

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

  ```bash
  # ipa-replica-manage dnanextrange-set serverB.example.com 1500-5000
  ```

**Verification steps**

- You can check that the new DNA ranges are set correctly by using the commands described in Displaying the currently assigned DNA ID ranges.
CHAPTER 32. 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.

32.1. 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

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

Verification

1. 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

32.2. 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

**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.

### 32.3. 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,cn=users,cn=accounts,dc=laptop,dc=example,dc=org
SubUID range start: 2147483648
SubUID range size: 65536
SubGID range start: 2147483648
SubGID range size: 65536

Number of entries returned 1
You can maintain multiple Identity Management (IdM) servers and let them replicate each other for redundancy purposes to mitigate or prevent server loss. For example, if one server fails, the other servers keep providing services to the domain. You can also recover the lost server by creating a new replica based on one of the remaining servers.

Data stored on an IdM server is replicated based on replication agreements: when two servers have a replication agreement configured, they share their data. The data that is replicated is stored in the topology suffixes. When two replicas have a replication agreement between their suffixes, the suffixes form a topology segment.

This chapter describes how to use Red Hat Ansible Engine to manage IdM replication agreements, topology segments, and topology suffixes. The chapter contains the following sections:

- Using Ansible to ensure a replication agreement exists in IdM
- Using Ansible to ensure replication agreements exist between multiple IdM replicas
- Using Ansible to check if a replication agreement exists between two replicas
- Using Ansible to verify that a topology suffix exists in IdM
- Using Ansible to re-initialize an IdM replica
- Using Ansible to ensure a replication agreement is absent in IdM

### 33.1. USING ANSIBLE TO ENSURE A REPLICATION AGREEMENT EXISTS IN IDM

Data stored on an Identity Management (IdM) server is replicated based on replication agreements: when two servers have a replication agreement configured, they share their data. Replication agreements are always bilateral: the data is replicated from the first replica to the other one as well as from the other replica to the first one.

This section describes how to use an Ansible playbook to ensure that a replication agreement of the domain type exists between server.idm.example.com and replica.idm.example.com.

**Prerequisites**

- Ensure that you understand the recommendations for designing your IdM topology listed in Connecting the replicas in a topology.
- 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.
  - 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 `add-topologysegment.yml` Ansible playbook file located in the `/usr/share/doc/ansible-freeipa/playbooks/topology/` directory:

   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/topology/add-topologysegment.yml add-topologysegment-copy.yml
   ```

3. Open the `add-topologysegment-copy.yml` file for editing.

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

   - Set the `ipaadmin_password` variable to the password of the IdM admin.
   - Set the `suffix` variable to either `domain` or `ca`, depending on what type of segment you want to add.
   - Set the `left` variable to the name of the IdM server that you want to be the left node of the replication agreement.
   - Set the `right` variable to the name of the IdM server that you want to be the right node of the replication agreement.
   - Ensure that the `state` variable is set to `present`.

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

   ```yaml
   ---
   - name: Playbook to handle topologysegment
     hosts: ipaserver
     become: true
     tasks:
     - name: Add topology segment
       ipatopologysegment:
         ipaadmin_password: Secret123
         suffix: domain
         left: server.idm.example.com
         right: replica.idm.example.com
         state: present
   ```

5. Save the file.

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

   ```
   $ ansible-playbook -v -i inventory add-topologysegment-copy.yml
   ```

Additional resources

- For more information on the concept of topology agreements, suffixes, and segments, see Explaining Replication Agreements, Topology Suffixes, and Topology Segments.
For more sample Ansible playbooks that use the `ipatopologysegment` module, see:

- The README-topology.md file in the `/usr/share/doc/ansible-freeipa/` directory. This file also contains the definitions of the `ipatopologysegment` variables.
- The `/usr/share/doc/ansible-freeipa/playbooks/topology` directory.

### 33.2. USING ANSIBLE TO ENSURE REPLICATION AGREEMENTS EXIST BETWEEN MULTIPLE IDM REPLICAS

Data stored on an Identity Management (IdM) server is replicated based on replication agreements: when two servers have a replication agreement configured, they share their data. Replication agreements are always bilateral: the data is replicated from the first replica to the other one as well as from the other replica to the first one.

This section describes how to ensure replication agreements exist between multiple pairs of replicas in IdM.

#### Prerequisites

- Ensure that you understand the recommendations for designing your IdM topology listed in Connecting the replicas in a topology
- 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.
  - 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 `add-topologysegments.yml` Ansible playbook file located in the `/usr/share/doc/ansible-freeipa/playbooks/topology/` directory:

   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/topology/add-topologysegments.yml add-topologysegments-copy.yml
   ```

3. Open the `add-topologysegments-copy.yml` file for editing.

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

   - Set the `ipaadmin_password` variable to the password of the IdM `admin`.
   - For every topology segment, add a line in the `ipatopology_segments` section and set the following variables:
Red Hat Enterprise Linux 9.0 Beta Configuring and managing Identity Management

Set the suffix variable to either domain or ca, depending on what type of segment you
want to add.
Set the left variable to the name of the IdM server that you want to be the left node of
the replication agreement.
Set the right variable to the name of the IdM server that you want to be the right node
of the replication agreement.
5. In the tasks section of the add-topologysegments-copy.yml file, ensure that the state
variable is set to present.
This is the modified Ansible playbook file for the current example:
--- name: Add topology segments
hosts: ipaserver
become: true
gather_facts: false
vars:
ipaadmin_password: Secret123
ipatopology_segments:
- {suffix: domain, left: replica1.idm.example.com , right: replica2.idm.example.com }
- {suffix: domain, left: replica2.idm.example.com , right: replica3.idm.example.com }
- {suffix: domain, left: replica3.idm.example.com , right: replica4.idm.example.com }
- {suffix: domain+ca, left: replica4.idm.example.com , right: replica1.idm.example.com }
tasks:
- name: Add topology segment
ipatopologysegment:
ipaadmin_password: "{{ ipaadmin_password }}"
suffix: "{{ item.suffix }}"
name: "{{ item.name | default(omit) }}"
left: "{{ item.left }}"
right: "{{ item.right }}"
state: present
#state: absent
#state: checked
#state: reinitialized
loop: "{{ ipatopology_segments | default([]) }}"
6. Save the file.
7. Run the Ansible playbook specifying the playbook file and the inventory file:
$ ansible-playbook -v -i inventory add-topologysegments-copy.yml
Additional resources
For more information on the concept of topology agreements, suffixes, and segments, see
Explaining Replication Agreements, Topology Suffixes, and Topology Segments .
For more sample Ansible playbooks that use the ipatopologysegment module, see:

The README-topology.md file in the /usr/share/doc/ansible-freeipa/ directory. This file

264


The README-topology.md file in the /usr/share/doc/ansible-freeipa/ directory. This file also contains the definitions of the ipatopologysegment variables.

The /usr/share/doc/ansible-freeipa/playbooks/topology directory.

33.3. USING ANSIBLE TO CHECK IF A REPLICATION AGREEMENT EXISTS BETWEEN TWO REPLICAS

Data stored on an Identity Management (IdM) server is replicated based on replication agreements: when two servers have a replication agreement configured, they share their data. Replication agreements are always bilateral: the data is replicated from the first replica to the other one as well as from the other replica to the first one.

This section describes how to verify that replication agreements exist between multiple pairs of replicas in IdM.

Prerequisites

- Ensure that you understand the recommendations for designing your Identity Management (IdM) topology listed in Connecting the replicas in a topology.
- 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.
  - 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 check-topologysegments.yml Ansible playbook file located in the /usr/share/doc/ansible-freeipa/playbooks/topology/ directory:

   $ cp /usr/share/doc/ansible-freeipa/playbooks/topology/check-topologysegments.yml check-topologysegments-copy.yml

3. Open the check-topologysegments-copy.yml file for editing.

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

   - Set the ipaadmin_password variable to the password of the IdM admin.
   - For every topology segment, add a line in the ipatopology_segments section and set the following variables:
     - Set the suffix variable to either domain or ca, depending on the type of segment you are adding.
Red Hat Enterprise Linux 9.0 Beta Configuring and managing Identity Management

Set the left variable to the name of the IdM server that you want to be the left node of
the replication agreement.
Set the right variable to the name of the IdM server that you want to be the right node
of the replication agreement.
5. In the tasks section of the check-topologysegments-copy.yml file, ensure that the state
variable is set to present.
This is the modified Ansible playbook file for the current example:
--- name: Add topology segments
hosts: ipaserver
become: true
gather_facts: false
vars:
ipaadmin_password: Secret123
ipatopology_segments:
- {suffix: domain, left: replica1.idm.example.com, right: replica2.idm.example.com }
- {suffix: domain, left: replica2.idm.example.com , right: replica3.idm.example.com }
- {suffix: domain, left: replica3.idm.example.com , right: replica4.idm.example.com }
- {suffix: domain+ca, left: replica4.idm.example.com , right:
replica1.idm.example.com }
tasks:
- name: Check topology segment
ipatopologysegment:
ipaadmin_password: "{{ ipaadmin_password }}"
suffix: "{{ item.suffix }}"
name: "{{ item.name | default(omit) }}"
left: "{{ item.left }}"
right: "{{ item.right }}"
state: checked
loop: "{{ ipatopology_segments | default([]) }}"
6. Save the file.
7. Run the Ansible playbook specifying the playbook file and the inventory file:
$ ansible-playbook -v -i inventory check-topologysegments-copy.yml
Additional resources
For more information on the concept of topology agreements, suffixes, and segments, see
Explaining Replication Agreements, Topology Suffixes, and Topology Segments .
For more sample Ansible playbooks that use the ipatopologysegment module, see:
The README-topology.md file in the /usr/share/doc/ansible-freeipa/ directory. This file
also contains the definitions of the ipatopologysegment variables.
The /usr/share/doc/ansible-freeipa/playbooks/topology directory.

33.4. USING ANSIBLE TO VERIFY THAT A TOPOLOGY SUFFIX EXISTS

266


### USING ANSIBLE TO VERIFY THAT A TOPOLOGY SUFFIX EXISTS IN IDM

In the context of replication agreements in Identity Management (IdM), topology suffixes store the data that is replicated. IdM supports two types of topology suffixes: **domain** and **ca**. Each suffix represents a separate back end, a separate replication topology. When a replication agreement is configured, it joins two topology suffixes of the same type on two different servers.

The **domain** suffix contains all domain-related data, such as users, groups, and policies. The **ca** suffix contains data for the Certificate System component. It is only present on servers with a certificate authority (CA) installed.

This section describes how to use an Ansible playbook to ensure that a topology suffix exists in IdM. The example describes how to ensure that the **domain** suffix exists in IdM.

**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.
  - 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 **verify-topologysuffix.yml** Ansible playbook file located in the ```/usr/share/doc/ansible-freeipa/playbooks/topology/``` directory:
   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/topology/verify-topologysuffix.yml verify-topologysuffix-copy.yml
   ```

3. Open the **verify-topologysuffix-copy.yml** Ansible playbook file for editing.

4. Adapt the file by setting the following variables in the **ipatopologysuffix** section:
   - Set the **ipaadmin_password** variable to the password of the IdM **admin**.
   - Set the **suffix** variable to **domain**. If you are verifying the presence of the **ca** suffix, set the variable to **ca**.
   - Ensure that the **state** variable is set to **verified**. No other option is possible.

   This is the modified Ansible playbook file for the current example:
   ```
   ---
   - name: Playbook to handle topologysuffix
   ```
hosts: ipaserver
become: true

tasks:
- name: Verify topology suffix
  ipatopologysuffix:
    ipaadmin_password: Secret123
    suffix: domain
    state: verified

5. Save the file.

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

```
$ ansible-playbook -v -i inventory verify-topologysuffix-copy.yml
```

Additional resources

- For more information on the concept of topology agreements, suffixes, and segments, see [Explaining Replication Agreements, Topology Suffixes, and Topology Segments](#).

- For more sample Ansible playbooks that use the `ipatopologysegment` module, see:
  - The `README-topology.md` file in the `/usr/share/doc/ansible-freeipa/` directory. This file also contains the definitions of the `ipatopologysuffix` variables.
  - The `/usr/share/doc/ansible-freeipa/playbooks/topology` directory.

### 33.5. USING ANSIBLE TO REINITIALIZE AN IDM REPLICA

If a replica has been offline for a long period of time or its database has been corrupted, you can reinitialize it. reinitialization refreshes the replica with an updated set of data. reinitialization can, for example, be used if an authoritative restore from backup is required.

**NOTE**

In contrast to replication updates, during which replicas only send changed entries to each other, reinitialization refreshes the whole database.

The local host on which you run the command is the reinitialized replica. To specify the replica from which the data is obtained, use the `direction` option.

This section describes how to use an Ansible playbook to reinitialize the `domain` data on `replica.idm.example.com` from `server.idm.example.com`.

**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.
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 `reinitialize-topologysegment.yml` Ansible playbook file located in the `/usr/share/doc/ansible-freeipa/playbooks/topology/` directory:
   ```bash
   $ cp /usr/share/doc/ansible-freeipa/playbooks/topology/reinitialize-topologysegment.yml reinitialize-topologysegment-copy.yml
   ```

3. Open the `reinitialize-topologysegment-copy.yml` file for editing.

4. Adapt the file by setting the following variables in the `ipatopologysegment` section:
   - Set the `ipaadmin_password` variable to the password of the IdM admin.
   - Set the `suffix` variable to `domain`. If you are reinitializing the `ca` data, set the variable to `ca`.
   - Set the `left` variable to the left node of the replication agreement.
   - Set the `right` variable to the right node of the replication agreement.
   - Set the `direction` variable to the direction of the reinitializing data. The `left-to-right` direction means that data flows from the left node to the right node.
   - Ensure that the `state` variable is set to `reinitialized`. This is the modified Ansible playbook file for the current example:
     ```yaml
     ---
     - name: Playbook to handle topologysegment
       hosts: ipaserver
       become: true

       tasks:
       - name: Reinitialize topology segment
         ipatopologysegment:
           ipaadmin_password: Secret123
           suffix: domain
           left: server.idm.example.com
           right: replica.idm.example.com
           direction: left-to-right
           state: reinitialized
     ```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:
   ```bash
   $ ansible-playbook -v -i inventory reinitialize-topologysegment-copy.yml
   ```

**Additional resources**

CHAPTER 33. USING ANSIBLE TO MANAGE THE REPLICATION TOPOLOGY IN IDM
33.6. USING ANSIBLE TO ENSURE A REPLICATION AGREEMENT IS ABSENT IN IDM

Data stored on an Identity Management (IdM) server is replicated based on replication agreements: when two servers have a replication agreement configured, they share their data. Replication agreements are always bilateral: the data is replicated from the first replica to the other one as well as from the other replica to the first one.

This section describes how to ensure a replication agreement between two replicas does not exist in IdM. The example describes how to ensure a replication agreement of the domain type does not exist between the replica01.idm.example.com and replica02.idm.example.com IdM servers.

Prerequisites

- Ensure that you understand the recommendations for designing your IdM topology listed in Connecting the replicas in a topology
- 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.
  - 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 delete-topologysegment.yml Ansible playbook file located in the /usr/share/doc ansible-freeipa/playbooks/topology/ directory:

   ```
   $ cp /usr/share/doc ansible-freeipa/playbooks/topology/delete-topologysegment.yml delete-topologysegment-copy.yml
   ```

3. Open the delete-topologysegment-copy.yml file for editing.

4. Adapt the file by setting the following variables in the ipatopologysegment task section:
- Set the `ipaadmin_password` variable to the password of the IdM admin.
- Set the `suffix` variable to `domain`. Alternatively, if you are ensuring that the `ca` data are not replicated between the left and right nodes, set the variable to `ca`.
- Set the `left` variable to the name of the IdM server that is the left node of the replication agreement.
- Set the `right` variable to the name of the IdM server that is the right node of the replication agreement.
- Ensure that the `state` variable is set to `absent`.

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

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

  tasks:
  - name: Delete topology segment
    ipatopologysegment:
      ipaadmin_password: Secret123
      suffix: domain
      left: replica01.idm.example.com
      right: replica02.idm.example.com:
      state: absent

5. Save the file.

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

   ```shell
   $ ansible-playbook -v -i inventory delete-topologysegment-copy.yml
   ```

**Additional resources**

- For more information on the concept of topology agreements, suffixes, and segments, see Explaining Replication Agreements, Topology Suffixes, and Topology Segments.
- For more sample Ansible playbooks that use the `ipatopologysegment` module, see:
  - The `README-topology.md` file in the `/usr/share/doc/ansible-freeipa/` directory. This file also contains the definitions of the `ipatopologysegment` variables.
  - The `/usr/share/doc/ansible-freeipa/playbooks/topology` directory.

**33.7. ADDITIONAL RESOURCES**

- For more information on how to plan the topology of your IdM deployment, see Planning the replica topology.
- For more information on how to install an IdM replica, see Installing an IdM replica.
CHAPTER 34. CONFIGURING IDM FOR EXTERNAL PROVISIONING OF USERS

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

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

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

You can configure a script to automatically move the staged users created by the external provisioning system to active users automatically.

This chapter contains these sections:

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

Additional materials

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

34.1. 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:
2. Create a user named **provisionator** with the privileges to add stage users.
   a. Add the provisionator user account:
      ```bash
      $ 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:
         ```bash
         $ 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:
         ```bash
         $ ipa role-add-privilege "System Provisioning" --privileges="Stage User Provisioning"
         ```
      iii. Add the provisionator user to the role:
         ```bash
         $ ipa role-add-member --users=provisionator "System Provisioning"
         ```
      iv. Verify that the provisionator exists in IdM:
         ```bash
         $ ipa user-find provisionator --all --raw
         1 user matched
         dn: uid=provisionator, cn=users, cn=accounts, dc=idm, dc=example, dc=com
         uid: provisionator
         [...] 
      3. Create a user, **activator**, with the privileges to manage user accounts.
         a. Add the activator user account:
            ```bash
            $ 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:
            ```bash
            $ ipa role-add-member --users=activator "User Administrator"
            ```
        4. Create a user group for application accounts:
           ```bash
           $ 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:
6. (Optional) Verify that the password policy exists in IdM:

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

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

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

8. Change the passwords for the user accounts:

$ kpasswd provisionator
$ kpasswd activator

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

Additional resources:

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

34.2. 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:

   ```bash
   #!/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:

   ```bash
   # 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:

   ```ini
   [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:

   ```ini
   [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:

   ```bash
   # 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.
   ```

34.3. 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

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:

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

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

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

**Prerequisites**

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

**Procedure**

1. Use the **SSH** protocol to connect to the IdM server using your IdM identity and credentials:
   
   ```
   $ ssh provisionator@server.idm.example.com
   Password:
   [provisionator@server ~]$ 
   ```

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

3. Enter the **ldapmodify** command and specify Generic Security Services API (GSSAPI) as the Simple Authentication and Security Layer (SASL) mechanism to use for authentication. Specify the name of the IdM server and the port:
   
   ```
   # ldapmodify -h server.idm.example.com -p 389 -Y GSSAPI
   SASL/GSSAPI authentication started
   SASL username: provisionator@IDM.EXAMPLE.COM
   SASL SSF: 56
   SASL data security layer installed.
   ```

4. Enter the **dn** of the user you are adding:
   
   ```
   dn: uid=stageuser,cn=staged
   users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com
   ```

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

6. Specify the LDAP object class categories required to allow the correct processing of the user life cycle:
   
   ```
   objectClass: top
   objectClass: inetorgperson
   ```
   
   You can specify additional object classes.

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

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

9. Enter the last name of the user:

sn: Jensen

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

    [Enter]

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

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

**Verification steps**

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

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

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

1. Note that the user is explicitly disabled by the `nsaccountlock` attribute.
CHAPTER 35. 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.

35.1. TEMPLATES FOR MANAGING IDM USER ACCOUNTS EXTERNALLY

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

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

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

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

For detailed example procedures, see the following sections:

- Adding an IdM stage user defined in an LDIF file
- Adding an IdM stage user directly from the CLI using ldapmodify
- Preserving an IdM user with ldapmodify

Templates for adding a new stage user

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

  ```
  dn: uid=user_login, cn=staged
  users,cn=accounts,cn=provisioning,dc=idm,dc=example,dc=com
  changetype: add
  objectClass: top
  objectClass: inetorgperson
  uid: user_login
  sn: surname
  givenName: first_name
  cn: full_name
  ```
A template for adding a user with **UID and GID assigned statically**

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

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

**Templates for modifying existing users**

- **Modifying a user’s attribute**

  ```
dn: distinguished_name
changeType: modify
replace: attribute_to_modify
attribute_to_modify: new_value
```

- **Disabling a user**

  ```
dn: distinguished_name
changeType: modify
replace: nsAccountLock
nsAccountLock: TRUE
```

- **Enabling a user**

  ```
dn: distinguished_name
changeType: modify
replace: nsAccountLock
nsAccountLock: FALSE
```

Updating the **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**

  ```
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 entries user is a user allowed to modify user and group information, for example activator or IdM administrator. The password in the example is this user’s password:

[...]

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

35.2. TEMPLATES FOR MANAGING IDM GROUP ACCOUNTS EXTERNALLY

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

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

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

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

Creating a new group

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

Modifying groups
• Deleting an existing group:

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

• Adding a member to a group:

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

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

• Removing a member from a group:

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

**NOTE**

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

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

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

**35.3. PRESERVING AN IDM USER WITH LDAPMODIFY**

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

**Prerequisites**

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

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

   $ kinit admin

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

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

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

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

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

   ```
   changetype: modrdn
   ```

5. Specify the `newrdn` for the user:

   ```
   newrdn: uid=user1
   ```

6. Indicate that you want to preserve the user:

   ```
   deleteoldrdn: 0
   ```

7. Specify the `new superior DN`:

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

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

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

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

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

**Verification steps**

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

   ```
   $ ipa user-find --preserved=true
   --------------
   1 user matched
   --------------
   User login: user1
   ```
First name: First 1
Last name: Last 1
Home directory: /home/user1
Login shell: /bin/sh
Principal name: user1@IDM.EXAMPLE.COM
Principal alias: user1@IDM.EXAMPLE.COM
Email address: user1@idm.example.com
UID: 1997010003
GID: 1997010003
Account disabled: True
Preserved user: True

Number of entries returned 1
CHAPTER 36. 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.

36.1. 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.
36.2. 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.

36.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running `ipa-client-install`
- If a one-time password (OTP) is used for enrollment

User privileges for optionally manually creating a host entry in IdM LDAP

The user privilege required for creating a host entry in IdM LDAP using the `ipa host-add` CLI command or the IdM Web UI is Host Administrators. The Host Administrators privilege can be obtained through the IT Specialist role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the `ipa-client-install` command. The level of credentials required for executing the `ipa-client-install` command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator’s credentials or the Host Administrators role. A full administrator is a member of the admins group. The Host Administrators role provides privileges to add hosts and enroll hosts. For details about this scenario, see Installing a client using user credentials: interactive installation.

- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator’s credentials to execute `ipa-client-install` successfully. The limited administrator in this case has the Enrollment Administrator role, which provides the Host Enrollment privilege. For details, see Installing a client using user credentials: interactive installation.

- The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the `ipa-client-install` command with the `--password` option, supplying the correct OTP. For
details, see Installing a client by using a one-time password: Interactive installation.

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

36.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- The enrollment stage (Table 36.1, "User and host enrollment"):
  - An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is `ipa stageuser-add`; for the host, the command is `ipa host-add`.
  - A file containing a key table or, abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the `ipa-client-install` command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.
  - While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

<table>
<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> (must be run on the host itself)</td>
</tr>
</tbody>
</table>

- The deployment stage (Table 36.2, "User and host session authentication"):
  - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
  - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
  - The TGT is then used to obtain specific tickets for specific services.

<table>
<thead>
<tr>
<th>User</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default means of authentication</td>
<td>Password</td>
</tr>
</tbody>
</table>
Starting a session (ordinary user)

\$ kinit user_name

[switch on the host]

The result of successful authentication

TGT to be used to obtain access to specific services

TGT to be used to obtain access to specific services

TGTs and other Kerberos tickets are generated as part of the Kerberos services and policies defined by the server. The initial granting of a Kerberos ticket, the renewing of the Kerberos credentials, and even the destroying of the Kerberos session are all handled automatically by the IdM services.

### 36.2.3. Alternative authentication options for IdM hosts

Apart from keytabs, IdM supports two other types of machine authentication:

- **SSH keys.** The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.

- **Machine certificates.** In this case, the machine uses an SSL certificate that is issued by the IdM server’s certificate authority and then stored in IdM’s Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called certmonger.

### 36.3. HOST OPERATIONS

This section lists the most common operations related to host enrollment and enablement, and explains the prerequisites, the context, and the consequences of performing them.

#### Table 36.3. 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>Enrolling a client</td>
<td>see Preparing the system for Identity Management client installation in Installing_Identity_Management</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>Action</td>
<td>What are the prerequisites of the action?</td>
<td>When does it make sense to run the command?</td>
<td>How is the action performed by a system administrator? What command(s) does he run?</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------</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>ipa host-disable host_name</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>ipa-getkeytab</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>ipa-client-install --keytab or ipa-client-install --force-join</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>ipa-client-install --uninstall</td>
</tr>
</tbody>
</table>

Table 36.4. Host operations part 2

| Action | On which machine can the administrator run the command(s)? | What happens when the action is performed? What are the consequences for the host’s functioning in IdM? What limitations are introduced/removed? |
|--------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|
### Action

<table>
<thead>
<tr>
<th>Action</th>
<th>On which machine can the administrator run the command(s)?</th>
<th>What happens when the action is performed? What are the consequences for the host’s functioning in IdM? What limitations are introduced/removed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enrolling a client</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>
</tr>
<tr>
<td>Disabling a client</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>
</tr>
<tr>
<td>Enabling a client</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>
</tr>
<tr>
<td>Re-enrolling a client</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>
</tr>
<tr>
<td>Un-enrolling a client</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>
</tr>
</tbody>
</table>

### 36.4. HOST ENTRY IN IDM LDAP

This section describes what a host entry in Identity Management (IdM) looks like and what attributes it can contain.

An LDAP host entry contains all relevant information about the client within IdM:

- Service entries associated with the host
- The host and service principal
- Access control rules
- Machine information, such as its physical location and operating system

**NOTE**

Note that the IdM Web UI **Identity → Hosts** tab does not show all the information about a particular host stored in the IdM LDAP.

### 36.4.1. Host entry configuration properties

A host entry can contain information about the host that is outside its system configuration, such as its physical location, MAC address, keys, and certificates.

This information can be set when the host entry is created if it is created manually. Alternatively, most of this information can be added to the host entry after the host is enrolled in the domain.

#### Table 36.5. 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 <strong>ethers</strong> 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>
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.

This option sets a password for the host which can be used in bulk enrollment.

This option generates a random password to be used in bulk enrollment.

A certificate blob for the host.

This sets whether the host can dynamically update its DNS entries if its IP address changes.

### 36.5. ADDING IDM HOST ENTRIES FROM IDM CLI

This section describes how to add host entries in Identity Management (IdM) using the command-line interface (CLI).

Host entries are created using the `host-add` command. This 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 36.1. 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 36.2. 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.

### 36.6. DELETING HOST ENTRIES FROM IDM CLI

- Use the `host-del` command to delete host records. If your IdM domain has integrated DNS, use the `--updatedns` option to remove the associated records of any kind for the host from the DNS:

  ```
  $ ipa host-del --updatedns client1.example.com
  ```

### 36.7. RE-ENROLLING AN IDENTITY MANAGEMENT CLIENT

#### 36.7.1. Client re-enrollment in IdM

This section describes how to re-enroll an Identity Management (IdM) client. If a client machine has been destroyed and lost connection with the IdM servers, for example due to the client’s hardware failure, and you still have its keytab, you can re-enroll the client. In this scenario, you want to get the client back in the IdM environment with the same hostname.

During the re-enrollment, the client generates a new Kerberos key and SSH keys, but the identity of the client in the LDAP database remains unchanged. After the re-enrollment, the host has its keys and other information in the same LDAP object with the same FQDN as previously, before the machine’s loss of connection with the IdM servers.

**IMPORTANT**

You can only re-enroll clients whose domain entry is still active. If you uninstalled a client (using `ipa-client-install --uninstall`) or disabled its host entry (using `ipa host-disable`), you cannot re-enroll it.

You cannot re-enroll a client after you have renamed it. This is because in Identity Management, the key attribute of the client’s entry in LDAP is the client’s hostname, its FQDN. As opposed to re-enrolling a client, during which the client’s LDAP object remains unchanged, the outcome of renaming a client is that the client has its keys and other information in a different LDAP object with a new FQDN. Thus the only way to rename a client is to uninstall the host from IdM, change the host’s hostname, and install it as an IdM client with a new name. For details on how to rename a client, see Section 36.8, “Renaming Identity Management client systems”.

#### 36.7.1.1. What happens during client re-enrollment

During re-enrollment, Identity Management:
Revokes the original host certificate
Creates new SSH keys
Generates a new keytab

36.7.2. Re-enrolling a client by using user credentials: Interactive re-enrollment

This procedure describes re-enrolling an Identity Management client interactively by using the credentials of an authorized user.

1. Re-create the client machine with the same host name.

2. Run the `ipa-client-install --force-join` command on the client machine:

   ```
   # ipa-client-install --force-join
   ```

3. The script prompts for a user whose identity will be used to re-enroll the client. This could be, for example, a `hostadmin` user with the Enrollment Administrator role:

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

Additional resources

- For a more detailed procedure on enrolling clients by using an authorized user’s credentials, see Installing a client by using user credentials: Interactive installation in Installing Identity Management.

36.7.3. Re-enrolling a client by using the client keytab: Non-interactive re-enrollment

Prerequisites

- Back up the original client keytab file, for example in the `/tmp` or `/root` directory.

Procedure

This procedure describes re-enrolling an Identity Management (IdM) client non-interactively by using the keytab of the client system. For example, re-enrollment using the client keytab is appropriate for an automated installation.

1. Re-create the client machine with the same host name.

2. Copy the keytab file from the backup location to the `/etc/` directory on the re-created client machine.

3. Use the `ipa-client-install` utility to re-enroll the client, and specify the keytab location with the `-keytab` option:

   ```
   # ipa-client-install --keytab /etc/krb5.keytab
   ```
The keytab specified in the `--keytab` option is only used when authenticating to initiate the enrollment. During the re-enrollment, IdM generates a new keytab for the client.

### 36.7.4. Testing an Identity Management client after installation

The Command-Line Interface informs you that the `ipa-client-install` was successful, but you can also do your own test.

To test that the Identity Management client can obtain information about users defined on the server, check that you are able to resolve a user defined on the server. For example, to check the default `admin` user:

```
[user@client1 ~]$ id admin
uid=1254400000(admin) gid=1254400000(admins) groups=1254400000(admins)
```

To test that authentication works correctly, `su -` as another IdM user:

```
[user@client1 ~]$ su - idm_user
Last login: Thu Oct 18 18:39:11 CEST 2018 from 192.168.122.1 on pts/0
[idm_user@client1 ~]$ 
```

### 36.8. RENAMING IDENTITY MANAGEMENT CLIENT SYSTEMS

The following sections describe how to change the host name of an Identity Management client system.

**WARNING**

Renaming a client is a manual procedure. Do not perform it unless changing the host name is absolutely required.

Renaming an Identity Management client involves:

1. Preparing the host. For details, see [Preparing an IdM client for its renaming](#).

2. Uninstalling the IdM client from the host. For details, see [Section 36.8.2, “Uninstalling an Identity Management client”](#).

3. Renaming the host. For details, see [Section 36.8.3, “Renaming the host system”](#).

4. Installing the IdM client on the host with the new name. For details, see [Section 36.8.4, “Re-installing an Identity Management client”](#).

5. Configuring the host after the IdM client installation. For details, see [Section 36.8.5, “Re-adding services, re-generating certificates, and re-adding host groups”](#).

### 36.8.1. Preparing an IdM client for its renaming
Before uninstalling the current client, make note of certain settings for the client. You will apply this configuration after re-enrolling the machine with a new host name.

- Identify which services are running on the machine:
  - Use the `ipa service-find` command, and identify services with certificates in the output:
    ```
    $ ipa service-find old-client-name.example.com
    ```
  - In addition, each host has a default host service which does not appear in the `ipa service-find` output. The service principal for the host service, also called a host principal, is `host/old-client-name.example.com`.

- For all service principals displayed by `ipa service-find old-client-name.example.com`, determine the location of the corresponding keytabs on the `old-client-name.example.com` system:
  ```
  # find / -name "*.keytab"
  ```
  Each service on the client system has a Kerberos principal in the form `service_name/host_name@REALM`, such as `ldap/old-client-name.example.com@EXAMPLE.COM`.

- Identify all host groups to which the machine belongs.
  ```
  # ipa hostgroup-find old-client-name.example.com
  ```

### 36.8.2. Uninstalling an Identity Management client

Uninstalling a client removes the client from the Identity Management domain, along with all of the specific Identity Management configuration of system services, such as System Security Services Daemon (SSSD). This restores the previous configuration of the client system.

**Procedure**

1. Run the `ipa-client-install --uninstall` command:
   ```
   [root@client]# ipa-client-install --uninstall
   ```

2. Remove the DNS entries for the client host manually from the server:
   ```
   [root@server]# ipa dnsrecord-del
   Record name: old-client-client
   Zone name: idm.example.com
   No option to delete specific record provided.
   Delete all? Yes/No (default No): yes
   ------------------------
   Deleted record "old-client-name"
   ```

3. For each identified keytab other than `/etc/krb5.keytab`, remove the old principals:
   ```
   [root@client ~]# ipa-rmkeytab -k /path/to/keytab -r EXAMPLE.COM
   ```
4. On an IdM server, remove the host entry. This removes all services and revokes all certificates issued for that host:

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

### 36.8.3. Renaming the host system

Rename the machine as required. For example:

    [root@client ]# hostnamectl set-hostname new-client-name.example.com

You can now re-install the Identity Management client to the Identity Management domain with the new host name.

### 36.8.4. Re-installing an Identity Management client

Install an client on your renamed host following the procedure described in Installing an Identity Management client: Basic scenario in Installing Identity Management.

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

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

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

2. Generate certificates for services that had a certificate assigned in the Preparing an IdM client for its renaming. You can do this:
   - Using the IdM administration tools
   - Using the certmonger utility

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

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

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

#### 36.9.1. Disabling Hosts

Complete this procedure to disable a host entry in IdM.

Domain services, hosts, and users can access an active host. There can be situations when it is necessary to remove an active host temporarily, for maintenance reasons, for example. Deleting the host in such situations is not desired as it removes the host entry and all the associated configuration permanently. Instead, choose the option of disabling the host.

Disabling a host prevents domain users from accessing it without permanently removing it from the domain. This can be done by using the host disable command. Disabling a host kills the host’s current, active keytabs.

For example:

    -
$ kinit admin
$ ipa host-disable client.example.com

As a result of disabling a host, the host becomes unavailable to all IdM users, hosts and services.

**IMPORTANT**
Disabling a host entry not only disables that host. It disables every configured service on that host as well.

### 36.9.2. Re-enabling Hosts

This section describes how to re-enable a disabled IdM host.

Disabling a host killed its active keytabs, which removed the host from the IdM domain without otherwise touching its configuration entry.

To re-enable a host, use the `ipa-getkeytab` command, adding:

- the `-s` option to specify which IdM server to request the keytab from
- the `-p` option to specify the principal name
- the `-k` option to specify the file to which to save the keytab.

For example, to request a new host keytab from `server.example.com` for `client.example.com`, and store the keytab in the `/etc/krb5.keytab` file:

```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 37. 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.

37.1. 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.

37.2. 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.

37.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running `ipa-client-install`
- If a one-time password (OTP) is used for enrollment

**User privileges for optionally manually creating a host entry in IdM LDAP**
The user privilege required for creating a host entry in IdM LDAP using the `ipa host-add` CLI command or the IdM Web UI is **Host Administrators**. The **Host Administrators** privilege can be obtained through the **IT Specialist** role.

**User privileges for joining the client to the IdM domain**
Hosts are configured as IdM clients during the execution of the `ipa-client-install` command. The level of credentials required for executing the `ipa-client-install` command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator’s credentials or the **Host Administrators** role. A full administrator is a member of the **admins** group. The **Host Administrators** role provides privileges to add hosts and enroll hosts. For details about this scenario, see Installing a client using user credentials: interactive installation.

- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator’s credentials to execute `ipa-client-install` successfully. The limited administrator in this case has the **Enrollment Administrator** role, which provides the **Host Enrollment** privilege. For details, see Installing a client using user credentials: interactive installation.

- The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the `ipa-client-install` command with the `--password` option, supplying the correct OTP. For details, see Installing a client by using a one-time password: Interactive installation.

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

37.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- The enrollment stage (Table 37.1, “User and host enrollment”):
An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is `ipa stageuser-add`; for the host, the command is `ipa host-add`.

A file containing a key table or, abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the `ipa-client-install` command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.

While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

### Table 37.1. 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><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></td>
<td>(must be run on the host itself)</td>
</tr>
</tbody>
</table>

- The deployment stage (Table 37.2, “User and host session authentication”):
  - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
  - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
  - The TGT is then used to obtain specific tickets for specific services.

### Table 37.2. User and host session authentication

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

### 37.2.3. Alternative authentication options for IdM hosts
Apart from keytabs, IdM supports two other types of machine authentication:

- SSH keys. The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.

- Machine certificates. In this case, the machine uses an SSL certificate that is issued by the IdM server’s certificate authority and then stored in IdM’s Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called certmonger.

### 37.3. HOST ENTRY IN IDM LDAP

This section describes what a host entry in Identity Management (IdM) looks like and what attributes it can contain.

An LDAP host entry contains all relevant information about the client within IdM:

- Service entries associated with the host
- The host and service principal
- Access control rules
- Machine information, such as its physical location and operating system

**NOTE**

Note that the IdM Web UI **Identity → Hosts** tab does not show all the information about a particular host stored in the IdM LDAP.

#### 37.3.1. Host entry configuration properties

A host entry can contain information about the host that is outside its system configuration, such as its physical location, MAC address, keys, and certificates.

This information can be set when the host entry is created if it is created manually. Alternatively, most of this information can be added to the host entry after the host is enrolled in the domain.

**Table 37.3. 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><code>--desc=description</code></td>
<td>A description of the host.</td>
</tr>
<tr>
<td>Locality</td>
<td><code>--locality=locality</code></td>
<td>The geographic location of the host.</td>
</tr>
<tr>
<td>Location</td>
<td><code>--location=location</code></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>

37.4. ADDING HOST ENTRIES FROM THE WEB UI

1. Open the Identity tab, and select the Hosts subtab.

2. Click Add at the top of the hosts list.
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.

**NOTE**

Select the **Force** check box if you want to skip checking whether the host is resolvable via DNS.

4. Click the **Add and Edit** button to go directly to the expanded entry page and enter more attribute information. Information about the host hardware and physical location can be included with the host entry.
Figure 37.3. Expanded Entry Page

**Host**: server.zone.example.com

server.zone.example.com is a member of:

<table>
<thead>
<tr>
<th>Settings</th>
<th>Host Groups</th>
<th>Netgroups</th>
<th>Roles</th>
<th>HBAC Rules</th>
<th>Sudo Rules</th>
</tr>
</thead>
</table>

- **Host Settings**
  - **Host name**: server.zone.example.com
  - **Principal name**: host/server.zone.example.com@EXAMPLE.COM
  - **Description**
  - **Class**
  - **Locality**
Ansible is an automation tool used to configure systems, deploy software, and perform rolling updates. Ansible includes support for Identity Management (IdM), and you can use Ansible modules to automate host management.

This chapter describes the following concepts and operations performed when managing hosts and host entries using Ansible playbooks:

- Hosts in IdM
- Host enrollment
- Ensuring the presence of IdM host entries that are only defined by their FQDNs
- Ensuring the presence of IdM host entries with IP addresses
- Ensuring the presence of multiple IdM host entries with random passwords
- Ensuring the presence of an IdM host entry with multiple IP addresses
- Ensuring the absence of IdM host entries

### 38.1. 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.

38.2. 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.

38.2.1. User privileges required for host enrollment

The host enrollment operation requires authentication to prevent an unprivileged user from adding unwanted machines to the IdM domain. The privileges required depend on several factors, for example:

- If a host entry is created separately from running `ipa-client-install`
- If a one-time password (OTP) is used for enrollment

User privileges for optionally manually creating a host entry in IdM LDAP

The user privilege required for creating a host entry in IdM LDAP using the `ipa host-add` CLI command or the IdM Web UI is Host Administrators. The Host Administrators privilege can be obtained through the IT Specialist role.

User privileges for joining the client to the IdM domain

Hosts are configured as IdM clients during the execution of the `ipa-client-install` command. The level of credentials required for executing the `ipa-client-install` command depends on which of the following enrolling scenarios you find yourself in:

- The host entry in IdM LDAP does not exist. For this scenario, you need a full administrator’s credentials or the Host Administrators role. A full administrator is a member of the admins group. The Host Administrators role provides privileges to add hosts and enroll hosts. For details about this scenario, see Installing a client using user credentials: interactive installation.
- The host entry in IdM LDAP exists. For this scenario, you need a limited administrator’s credentials to execute `ipa-client-install` successfully. The limited administrator in this case has the Enrollment Administrator role, which provides the Host Enrollment privilege. For details, see Installing a client using user credentials: interactive installation.
The host entry in IdM LDAP exists, and an OTP has been generated for the host by a full or limited administrator. For this scenario, you can install an IdM client as an ordinary user if you run the `ipa-client-install` command with the `--password` option, supplying the correct OTP. For details, see Installing a client by using a one-time password: Interactive installation.

After enrollment, IdM hosts authenticate every new session to be able to access IdM resources. Machine authentication is required for the IdM server to trust the machine and to accept IdM connections from the client software installed on that machine. After authenticating the client, the IdM server can respond to its requests.

### 38.2.2. Enrollment and authentication of IdM hosts and users: comparison

There are many similarities between users and hosts in IdM. This section describes some of the similarities that can be observed during the enrollment stage as well as those that concern authentication during the deployment stage.

- **The enrollment stage** (Table 38.1, “User and host enrollment”):
  - An administrator can create an LDAP entry for both a user and a host before the user or host actually join IdM: for the stage user, the command is `ipa stageuser-add`; for the host, the command is `ipa host-add`.
  - A file containing a key table, or abbreviated, keytab, a symmetric key resembling to some extent a user password, is created during the execution of the `ipa-client-install` command on the host, resulting in the host joining the IdM realm. Analogically, a user is asked to create a password when they activate their account, thus joining the IdM realm.
  - While the user password is the default authentication method for a user, the keytab is the default authentication method for a host. The keytab is stored in a file on the host.

<table>
<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</td>
<td><code>$ ipa stageuser-activate user_name</code></td>
<td><code>$ ipa-client install [--password]</code></td>
</tr>
<tr>
<td>account</td>
<td>(must be run on the host itself)</td>
<td>(must be run on the host itself)</td>
</tr>
</tbody>
</table>

- **The deployment stage** (Table 38.2, "User and host session authentication"):
  - When a user starts a new session, the user authenticates using a password; similarly, every time it is switched on, the host authenticates by presenting its keytab file. The System Security Services Daemon (SSSD) manages this process in the background.
  - If the authentication is successful, the user or host obtains a Kerberos ticket granting ticket (TGT).
  - The TGT is then used to obtain specific tickets for specific services.

<table>
<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</td>
<td><code>$ ipa stageuser-activate user_name</code></td>
<td><code>$ ipa-client install [--password]</code></td>
</tr>
<tr>
<td>account</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.

### 38.2.3. Alternative authentication options for IdM hosts

Apart from keytabs, IdM supports two other types of machine authentication:

- **SSH keys.** The SSH public key for the host is created and uploaded to the host entry. From there, the System Security Services Daemon (SSSD) uses IdM as an identity provider and can work in conjunction with OpenSSH and other services to reference the public keys located centrally in IdM.

- **Machine certificates.** In this case, the machine uses an SSL certificate that is issued by the IdM server’s certificate authority and then stored in IdM’s Directory Server. The certificate is then sent to the machine to present when it authenticates to the server. On the client, certificates are managed by a service called `certmonger`.

### 38.3. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH FQDN USING ANSIBLE PLAYBOOKS

This section describes ensuring the presence of host entries in Identity Management (IdM) using Ansible playbooks. The host entries are only defined by their [fully-qualified domain names](https://en.wikipedia.org/wiki/FQDN) (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.
Without Ansible, host entries are created in IdM using the `ipa host-add` command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: `state: present`.

**Prerequisites**
- You know the IdM administrator password.
- The `ansible-freeipa` package is installed on the Ansible controller.

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

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

2. Create an Ansible playbook file with the FQDN of the host whose presence in IdM you want to ensure. To simplify this step, you can copy and modify the example in the `~/usr/share/doc/ansible-freeipa/playbooks/host/add-host.yml` file:

   ```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:

   ```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:
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
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.

### 38.4. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH DNS INFORMATION USING ANSIBLE PLAYBOOKS

This section describes ensuring the presence of host entries in Identity Management (IdM) using Ansible playbooks. The host entries are defined by their fully-qualified domain names (FQDNs) and their IP addresses.

**NOTE**

Without Ansible, host entries are created in IdM using the `ipa host-add` command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: `state: present`.

**Prerequisites**

- You know the IdM administrator password.
- The `ansible-freeipa` package is installed on the Ansible controller.

**Procedure**

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:
    ipaadmin_password: MySecret123
    name: host01.idm.example.com
    description: Example host
    ip_address: 192.168.0.123
    locality: Lab
    ns_host_location: Lab
    ns_os_version: CentOS 7
    ns_hardware_platform: Lenovo T61
    mac_address:
    - "08:00:27:E3:B1:2D"
    - "52:54:00:BD:97:1E"
    state: present

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
   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.
ENSURING THE PRESENCE OF MULTIPLE IDM HOST ENTRIES WITH RANDOM PASSWORDS USING ANSIBLE PLAYBOOKS

The ipahost module allows the system administrator to ensure the presence or absence of multiple host entries in IdM using just one Ansible task. This section describes how to ensure the presence of multiple host entries that are only defined by their fully-qualified domain names (FQDNs). Running the Ansible playbook generates random passwords for the hosts.

**NOTE**

Without Ansible, host entries are created in IdM using the ipa host-add command. The result of adding a host to IdM is the state of the host being present in IdM. Because of the Ansible reliance on idempotence, to add a host to IdM using Ansible, you must create a playbook in which you define the state of the host as present: **state: present**.

**Prerequisites**

- You know the IdM administrator password.
- The ansible-freeipa package is installed on the Ansible controller.

**Procedure**

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

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

2. Create an Ansible playbook file with the fully-qualified domain name (FQDN) of the hosts whose presence in IdM you want to ensure. To make the Ansible playbook generate a random password for each host even when the host already exists in IdM and **update_password** is limited to **on_create**, add the **random: yes** and **force: yes** options. To simplify this step, you can copy and modify the example from the /usr/share/doc/ansible-freeipa/README-host.md Markdown file:

   ```
   ---
   - name: Ensure hosts with random password
     hosts: ipaserver
     become: true
     register: ipahost

     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:

```bash
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-are-present.yml
```

[...]

```
TASK [Hosts host01.idm.example.com and host02.idm.example.com present with random passwords]
changed: [r8server.idm.example.com] => { "changed": true, "host":
{ "host01.idm.example.com": { "randompassword": "0HoIRvjUdH0Ycbf6uYdWTxH"},
"host02.idm.example.com": { "randompassword": "5VdLgrf3wvojmACdHC3uA3s"} ]
```

**NOTE**

To deploy the hosts as IdM clients using random, one-time passwords (OTPs), see [Authorization options for IdM client enrollment using an Ansible playbook](#) or [Installing a client by using a one-time password: Interactive installation](#).

**Verification steps**

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 one of the hosts:

   ```bash
   $ 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.

**38.6. ENSURING THE PRESENCE OF AN IDM HOST ENTRY WITH MULTIPLE IP ADDRESSES USING ANSIBLE PLAYBOOKS**

This section describes how to ensure the presence of a host entry in Identity Management (IdM) using Ansible playbooks. The host entry is defined by its **fully-qualified domain name** (FQDN) and its multiple IP addresses.

**NOTE**

In contrast to the *ipa host* utility, the Ansible *ipahost* module can ensure the presence or absence of several IPv4 and IPv6 addresses for a host. The *ipa host-mod* command cannot handle IP addresses.

**Prerequisites**

- You know the IdM administrator password.
- The *ansible-freeipa* package is installed on the Ansible controller.
Procedure

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

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

2. Create an Ansible playbook file. Specify, as the name of the `ipahost` variable, the **fully-qualified domain name** (FQDN) of the host whose presence in ldM 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 host01.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:

```bash
$ 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 ldM LDAP server but does not enroll the host into the ldM Kerberos realm. For that, you must deploy the host as an ldM client. For details, see *Installing an Identity Management client using an Ansible playbook*.

Verification steps

1. Log in to your ldM 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
Principal name: host/host01.idm.example.com@IDM.EXAMPLE.COM
Principal alias: host/host01.idm.example.com@IDM.EXAMPLE.COM
```
The output confirms that `host01.idm.example.com` exists in IdM.

3. To verify that the multiple IP addresses of the host exist in the IdM DNS records, enter the `ipa dnsrecord-show` command and specify the following information:

   - The name of the IdM domain
   - The name of the host

   ```
   $ ipa dnsrecord-show idm.example.com host01
   [...
   Record name: host01
   A record: 192.168.0.123, 192.168.0.124
   AAAA record: fe80::20c:29ff:fe02:a1b3, fe80::20c:29ff:fe02:a1b4
   ```

   The output confirms that all the IPv4 and IPv6 addresses specified in the playbook are correctly associated with the `host01.idm.example.com` host entry.

38.7. ENSURING THE ABSENCE OF AN IDM HOST ENTRY USING ANSIBLE PLAYBOOKS

This section describes how to ensure the absence of host entries in Identity Management (IdM) using Ansible playbooks.

Prerequisites

- IdM administrator credentials

Procedure

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

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

2. Create an Ansible playbook file with the fully-qualified domain name (FQDN) of the host whose absence from IdM you want to ensure. If your IdM domain has integrated DNS, use the `updatedns: yes` option to remove the associated records of any kind for the host from the DNS.

   To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/host/delete-host.yml` file:

   ```
   ---
   - name: Host absent
     hosts: ipaserver
     become: true

     tasks:
     - name: Host host01.idm.example.com absent
       ipahost:
   ```
ipaadmin_password: MySecret123
name: host01.idm.example.com
updatedns: yes
state: absent

3. Run the playbook:

```bash
$ 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:

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

2. Display information about `host01.idm.example.com`:

   ```bash
   $ ipa host-show host01.idm.example.com
   ipa: ERROR: host01.idm.example.com: host not found
   ```

The output confirms that the host does not exist in IdM.

**Additional resources**

- You can see the definitions of the `ipahost` variables as well as sample Ansible playbooks for ensuring the presence, absence, and disablement of hosts in the `/usr/share/doc/ansible-freeipa/README-host.md` Markdown file.

- Additional playbooks are in the `/usr/share/doc/ansible-freeipa/playbooks/host` directory.
CHAPTER 39. 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

39.1. 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.

39.2. 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.

```bash
$ 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:

```bash
$ 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
-------------------
```

### 39.3. 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:

```bash
$ ipa hostgroup-add --desc 'My new host group' group_name
---------------------
Added hostgroup "group_name"
---------------------
Host-group: group_name
Description: My new host group
---------------------
```

### 39.4. 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.

### 39.5. 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:

```
$ 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.

### 39.6. REMOVING IDM HOST GROUP MEMBERS USING THE CLI

You can remove hosts as well as host groups from an IdM host group using a single command.

**Prerequisites**

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see Using kinit to log in to IdM manually.
- Optional. Use the `ipa hostgroup-find` command to confirm that the group includes the member you want to remove.

**Procedure**

1. To remove a host group member, use the `ipa hostgroup-remove-member` command and provide the relevant information. You can specify the type of member to remove using these options:
   - Use the `--hosts` option to remove one or more hosts from an IdM host group. For example, to remove the host named `example_member` from the group named `group_name`:
     ```
     $ ipa hostgroup-remove-member group_name --hosts example_member
     ```
   - Use the `--hostgroups` option to remove one or more host groups from an IdM host group. For example, to remove the host group named `nested_group` from the group named `group_name`:
     ```
     $ ipa hostgroup-remove-member group_name --hostgroups nested_group
     ```
39.7. ADDING IDM HOST GROUP MEMBER MANAGERS USING THE CLI

You can add hosts as well as host groups as member managers to an IdM host group using a single command. Member managers can add hosts or host groups to IdM host groups but cannot change the attributes of a host group.

Prerequisites

- Administrator privileges for managing IdM or User Administrator role.
- An active Kerberos ticket. For details, see Using kinit to log in to IdM manually.
- You must have the name of the host or host group you are adding as member managers and the name of the host group you want them to manage.

Procedure

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

2. To add a member manager to a host group, use the `ipa hostgroup-add-member-manager`. For example, to add the user named `example_member` as a member manager to the group named `group_name`:

   ```
   $ ipa hostgroup-add-member-manager group_name --user example_member
   Host-group: group_name
   Member hosts: server.idm.example.com
   Member host-groups: project_admins
   Member of netgroups: group_name
   Membership managed by users: example_member
   -------------------------
   Number of members added 1
   -------------------------
   ```

3. Use the `--groups` option to add one or more host groups as a member manager to an IdM host group.
   For example, to add the host group named `admin_group` as a member manager to the group named `group_name`:

   ```
   $ ipa hostgroup-add-member-manager group_name --groups admin_group
   Host-group: group_name
   Member hosts: server.idm.example.com
   Member host-groups: project_admins
   Member of netgroups: group_name
   Membership managed by users: example_member
   Membership managed by groups: admin_group
   -------------------------
   Number of members added 1
   -------------------------
   ```
$ ipa hostgroup-add-member-manager group_name --groups admin_group
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Member of netgroups: group_name
Membership managed by groups: admin_group
Membership managed by users: example_member

-------------------------
Number of members added 1
-------------------------

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

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

$ ipa hostgroup-show group_name
Host-group: group_name
Member hosts: server.idm.example.com
Member host-groups: project_admins
Membership managed by groups: admin_group
Membership managed by users: example_member

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

39.8. 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 40. 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

40.1. 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.

40.2. 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.

40.3. 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.

40.4. 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.

40.5. 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.

---

40.6. 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.
40.7. 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.
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.

### 40.8. 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' dialog](image)

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.
CHAPTER 40. MANAGING HOST GROUPS USING THE IDM WEB UI

Host Group: test_hostgroup

No entries.
CHAPTER 41. 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

41.1. 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.

41.2. ENSURING THE PRESENCE OF IDM HOST GROUPS USING ANSIBLE PLAYBOOKS

This section describes how to ensure the presence of host groups in Identity Management (IdM) using Ansible playbooks.
NOTE

Without Ansible, host group entries are created in IdM using the `ipa hostgroup-add` command. The result of adding a host group to IdM is the state of the host group being present in IdM. Because of the Ansible reliance on idempotence, to add a host group to IdM using Ansible, you must create a playbook in which you define the state of the host group as present: `state: present`.

Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.

Procedure

1. Create an inventory file, for example `inventory.file`, and define `ipaserver` in it with the list of IdM servers to target:

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

2. Create an Ansible playbook file with the necessary host group information. For example, to ensure the presence of a host group named `databases`, specify `name: databases` in the `-ipahostgroup` task. To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/user/ensure-hostgroup-is-present.yml` file.

   ```
   ---
   - name: Playbook to handle hostgroups
     hosts: ipaserver
     become: true
     tasks:
     # Ensure host-group databases is present
     - ipahostgroup:
       ipaadmin_password: MySecret123
       name: databases
       state: present
   ```

   In the playbook, `state: present` signifies a request to add the host group to IdM unless it already exists there.

3. Run the playbook:

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

Verification steps

1. Log into `ipaserver` as admin:

   ```
   $ ssh admin@server.idm.example.com
   Password:
   [admin@server /]$
2. Request a Kerberos ticket for admin:

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

3. Display information about the host group whose presence in IdM you wanted to ensure:

```bash
$ ipa hostgroup-show databases
Host-group: databases
```

The **databases** host group exists in IdM.

### 41.3. 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:

   ```ini
   [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:

   ```yaml
   ---
   - 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.

**41.4. NESTING IDM HOST GROUPS USING ANSIBLE PLAYBOOKS**

This section describes ensuring the presence of nested host groups in Identity Management (IdM) host groups using Ansible playbooks.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.
- The host groups you reference from the Ansible playbook file exist in IdM. For details, see `Ensuring the presence of IdM host groups using Ansible playbooks`.

**Procedure**
1. Create an inventory file, for example `inventory.file`, and define `ipaserver` in it with the list of IdM servers to target:

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

2. Create an Ansible playbook file with the necessary host group information. To ensure that a nested host group `A` exists in a host group `B` in the Ansible playbook, specify, among the `-ipahostgroup` variables, the name of the host group `B` using the `name` variable. Specify the name of the nested hostgroup `A` with the `hostgroup` variable. To simplify this step, you can copy and modify the examples in the `/usr/share/doc/ansible-freeipa/playbooks/hostgroup/ensure-hosts-and-hostgroups-are-present-in-hostgroup.yml` file:

```
---
- name: Playbook to handle hostgroups
  hosts: ipaserver
  become: true
  tasks:
    # Ensure hosts and hostgroups are present in existing databases hostgroup
    - ipahostgroup:
        ipaadmin_password: MySecret123
        name: databases
        hostgroup:
          - mysql-server
          - oracle-server
        action: member
```

This Ansible playbook ensures the presence of the `mysql-server` and `oracle-server` host groups in the `databases` host group. The `action: member` line indicates that when the playbook is run, no attempt is made to add the `databases` group itself to IdM.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
path_to_playbooks_directory/ensure-hosts-or-hostgroups-are-present-in-hostgroup.yml
```

Verification steps

1. Log into `ipaserver` as admin:

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

2. Request a Kerberos ticket for admin:

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

3. Display information about the host group in which nested host groups are present:
$ ipa hostgroup-show databases
Host-group: databases
Member hosts: db.idm.example.com
Member host-groups: mysql-server, oracle-server

The mysql-server and oracle-server host groups exist in the databases host group.

41.5. 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:
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.

41.6. 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.
41.7. 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.

41.8. 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.

```yaml
---
- 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.

**41.9. 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 42. MANAGING KERBEROS TICKET POLICIES

Kerberos ticket policies in Identity Management (IdM) set restrictions on Kerberos ticket access, duration, and renewal. You can configure Kerberos ticket policies for the Key Distribution Center (KDC) running on your IdM server.

This chapter presents the following Kerberos ticket management topics and tasks:

- The role of the IdM KDC
- IdM Kerberos ticket policy types
- Kerberos authentication indicators
- Enforcing authentication indicators for an IdM service
- Configuring the global ticket lifecycle policy
- Configuring global ticket policies per authentication indicator
- Configuring the default ticket policy for a user
- Configuring individual authentication indicator ticket policies for a user
- Authentication indicator options for the `krbtpolicy-mod` command

42.1. THE ROLE OF THE IDM KDC

Identity Management’s authentication mechanisms use the Kerberos infrastructure established by the Key Distribution Center (KDC). The KDC is the trusted authority that stores credential information and ensures the authenticity of data originating from entities within the IdM network.

Each IdM user, service, and host acts as a Kerberos client and is identified by a unique Kerberos principal:

- For users: `identifier@REALM`, such as `admin@EXAMPLE.COM`
- For services: `service/fully-qualified-hostname@REALM`, such as `http/server.example.com@EXAMPLE.COM`
- For hosts: `host/fully-qualified-hostname@REALM`, such as `host/client.example.com@EXAMPLE.COM`

The following image is a simplification of the communication between a Kerberos client, the KDC, and a Kerberized application that the client wants to communicate with.
1. A Kerberos client identifies itself to the KDC by authenticating as a Kerberos principal. For example, an IdM user performs `kinit username` and provides their password.

2. The KDC checks for the principal in its database, authenticates the client, and evaluates Kerberos ticket policies to determine whether to grant the request.

3. The KDC issues the client a ticket-granting ticket (TGT) with a lifecycle and authentication indicators according to the appropriate ticket policy.

4. With the TGT, the client requests a service ticket from the KDC to communicate with a Kerberized service on a target host.

5. The KDC checks if the client’s TGT is still valid, and evaluates the service ticket request against ticket policies.

6. The KDC issues the client a service ticket.

7. With the service ticket, the client can initiate encrypted communication with the service on the target host.

### 42.2. IDM KERBEROS TICKET POLICY TYPES

IdM Kerberos ticket policies implement the following ticket policy types:

**Connection policy**

To protect Kerberized services with different levels of security, you can define connection policies to enforce rules based on which pre-authentication mechanism a client used to retrieve a ticket-granting ticket (TGT).

For example, you can require smart card authentication to connect to `client1.example.com`, and require two-factor authentication to access the `testservicel` application on `client2.example.com`. 
To enforce connection policies, associate authentication indicators with services. Only clients that have the required authentication indicators in their service ticket requests are able to access those services. For more information, see Kerberos authentication indicators.

Ticket lifecycle policy
Each Kerberos ticket has a lifetime and a potential renewal age: you can renew a ticket before it reaches its maximum lifetime, but not after it exceeds its maximum renewal age. The default global ticket lifetime is one day (86400 seconds) and the default global maximum renewal age is one week (604800 seconds). To adjust these global values, see Configuring the global ticket lifecycle policy.

You can also define your own ticket lifecycle policies:

- To configure different global ticket lifecycle values for each authentication indicator, see Configuring global ticket policies per authentication indicator.
- To define ticket lifecycle values for a single user that apply regardless of the authentication method used, see Configuring the default ticket policy for a user.
- To define individual ticket lifecycle values for each authentication indicator that only apply to a single user, see Configuring individual authentication indicator ticket policies for a user.

42.3. KERBEROS AUTHENTICATION INDICATORS
The Kerberos Key Distribution Center (KDC) attaches authentication indicators to a ticket-granting ticket (TGT) based on which pre-authentication mechanism the client used to prove its identity:

- otp
two-factor authentication (password + One-Time Password)
- radius
  RADIUS authentication (commonly for 802.1x authentication)
- pkinit
  PKINIT, smart card, or certificate authentication
- hardened
  hardened passwords (SPAKE or FAST)[1]

The KDC then attaches the authentication indicators from the TGT to any service ticket requests that stem from it. The KDC enforces policies such as service access control, maximum ticket lifetime, and maximum renewable age based on the authentication indicators.

42.3.1. Authentication indicators and IdM services
If you associate a service or a host with an authentication indicator, only clients that used the corresponding authentication mechanism to obtain a TGT will be able to access it. The KDC, not the application or service, checks for authentication indicators in service ticket requests, and grants or denies requests based on Kerberos connection policies.

For example, to require two-factor authentication to connect to host secure.example.com, associate the otp authentication indicator with the host/secure.example.com@EXAMPLE.COM Kerberos principal. Only users who used a One-Time password to obtain their initial TGT from the KDC will be able to log in.
If a service or a host has no authentication indicators assigned to it, it will accept tickets authenticated
by any mechanism.

Additional resources

- To associate an IdM service with authentication indicators, see Enforcing authentication
  indicators for an IdM service.

42.4. ENFORCING AUTHENTICATION INDICATORS FOR AN IDM SERVICE

This procedure describes creating an IdM service and configuring it to require particular Kerberos
authentication indicators from incoming service ticket requests.

By associating authentication indicators with an IdM service, only clients who used those specific pre-
authentication mechanisms to obtain their initial ticket-granting ticket (TGT) will be able to access the
service.

42.4.1. Creating an IdM service entry and its Kerberos keytab

Adding an IdM service entry to IdM for a service running on an IdM host creates a corresponding
Kerberos principal, and allows the service to request an SSL certificate, a Kerberos keytab, or both.

The following procedure describes creating an IdM service entry and generating an associated Kerberos
keytab for encrypting communication with that service.

Prerequisites

- Your service can store a Kerberos principal, an SSL certificate, or both.

Procedure

1. Add an IdM service with the `ipa service-add` command to create a Kerberos principal
   associated with it. For example, to create the IdM service entry for the `testservice`
   application that runs on host `client.example.com`:

   ```
   [root@client ~]# ipa service-add testservice/client.example.com
   ------------------------------------------
   Modified service "testservice/client.example.com@EXAMPLE.COM"
   ------------------------------------------
   Principal name: testservice/client.example.com@EXAMPLE.COM
   Principal alias: testservice/client.example.com@EXAMPLE.COM
   Managed by: client.example.com
   ```

2. Generate and store a Kerberos keytab for the service on the client.

   ```
   [root@client ~]# ipa-getkeytab -k /etc/testservice.keytab -p
testservice/client.example.com
   Keytab successfully retrieved and stored in: /etc/testservice.keytab
   ```

Verification steps

1. Display information about an IdM service with the `ipa service-show` command.
2. Display the contents of the service’s Kerberos keytab with the `klist` command.

```
[root@server etc]# klist -ekt /etc/testservice.keytab
Keytab name: FILE:/etc/testservice.keytab
KVNO  Timestamp                     Principal
--- ------------------- ------------------------------------------------------
  2 04/01/2020 17:52:55 testservice/client.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
  2 04/01/2020 17:52:55 testservice/client.example.com@EXAMPLE.COM (aes128-cts-hmac-sha1-96)
  2 04/01/2020 17:52:55 testservice/client.example.com@EXAMPLE.COM (camellia128-cts-cmac)
  2 04/01/2020 17:52:55 testservice/client.example.com@EXAMPLE.COM (camellia256-cts-cmac)
```

### 42.4.2. Associating authentication indicators with an IdM service

This procedure describes configuring a service to require particular Kerberos authentication indicators from incoming service ticket requests.

**Prerequisites**

- You have created an IdM service entry for a service that runs on an IdM host. See Creating an IdM service entry and its Kerberos keytab.

**WARNING**

Do not assign authentication indicators to internal IdM services. The following IdM services cannot perform the interactive authentication steps required by PKINIT and multi-factor authentication methods:

- `host/server.example.com@EXAMPLE.COM`
- `HTTP/server.example.com@EXAMPLE.COM`
- `ldap/server.example.com@EXAMPLE.COM`
- `DNS/server.example.com@EXAMPLE.COM`
- `cifs/server.example.com@EXAMPLE.COM`

**Procedure**

- Use the `ipa service-mod` command to specify one or more required authentication indicators for a service, identified with the `--auth-ind` argument.
<table>
<thead>
<tr>
<th>Authentication method</th>
<th>--auth-ind value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-factor authentication</td>
<td>otp</td>
</tr>
<tr>
<td>RADIUS authentication</td>
<td>radius</td>
</tr>
<tr>
<td>PKINIT, smart card, or certificate authentication</td>
<td>pkinit</td>
</tr>
<tr>
<td>Hardened passwords (SPAKE or FAST)</td>
<td>hardened</td>
</tr>
</tbody>
</table>

For example, to require that a user was authenticated with smart card or OTP authentication to retrieve a service ticket for the `testservicetestservice` principal on host `client.example.com`:

```
[root@server ~]# ipa service-mod testservice/client.example.com@EXAMPLE.COM --auth-ind otp --auth-ind pkinit
```

```
Modified service "testservice/client.example.com@EXAMPLE.COM"
Principal name: testservice/client.example.com@EXAMPLE.COM
Principal alias: testservice/client.example.com@EXAMPLE.COM
Authentication Indicators: otp, pkinit
Managed by: client.example.com
```

**NOTE**

To remove all authentication indicators from a service, provide an empty list of indicators:

```
[root@server ~]# ipa service-mod testservice/client.example.com@EXAMPLE.COM --auth-ind "
```

```
Modified service "testservice/client.example.com@EXAMPLE.COM"
Principal name: testservice/client.example.com@EXAMPLE.COM
Principal alias: testservice/client.example.com@EXAMPLE.COM
Managed by: client.example.com
```

**Verification steps**

- Display information about an IdM service, including the authentication indicators it requires, with the `ipa service-show` command.

```
[root@server ~]# ipa service-show testservice/client.example.com
```

```
Principal name: testservice/client.example.com@EXAMPLE.COM
Principal alias: testservice/client.example.com@EXAMPLE.COM
Authentication Indicators: otp, pkinit
Keytab: True
Managed by: client.example.com
```

**Additional resources**
To test requesting a service ticket for an IdM service, see Retrieving a Kerberos service ticket for an IdM service.

42.4.3. Retrieving a Kerberos service ticket for an IdM service

The following procedure describes retrieving a Kerberos service ticket for an IdM service. You can use this procedure to test Kerberos ticket policies.

Prerequisites

- If the service you are working with is not an internal IdM service, you have created a corresponding IdM service entry for it. See Creating an IdM service entry and its Kerberos keytab.
- You have a Kerberos ticket-granting ticket (TGT).

Procedure

- Use the `kvno` command with the `-S` option to retrieve a service ticket, and specify the name of the IdM service and the fully-qualified domain name of the host that manages it.

```bash
[root@server ~]# kvno -S testservice client.example.com
testservice/client.example.com@EXAMPLE.COM: kvno = 1
```

**NOTE**

If you need to access an IdM service and your current ticket-granting ticket (TGT) does not possess the required authentication indicators associated with it, clear your current Kerberos credentials cache with the `kdestroy` command and retrieve a new TGT:

```bash
[root@server ~]# kdestroy
```

For example, if you initially retrieved a TGT by authenticating with a password, and you need to access an IdM service that has the `pkinit` authentication indicator associated with it, destroy your current credentials cache and re-authenticate with a smart card. See Kerberos authentication indicators.

Verification steps

- Use the `klist` command to verify that the service ticket is in the default Kerberos credentials cache.

```bash
[root@server etc]# klist
Ticket cache: KCM:1000
Default principal: admin@EXAMPLE.COM

Valid starting       Expires              Service principal
04/01/2020 12:52:42  04/02/2020 12:52:39  krbtgt/EXAMPLE.COM@EXAMPLE.COM
04/01/2020 12:54:07 04/02/2020 12:52:39 testservice/client.example.com@EXAMPLE.COM
```

42.4.4. Additional resources
For more information on Kerberos authentication indicators, see Section 42.3, "Kerberos authentication indicators".

### 42.5. CONFIGURING THE GLOBAL TICKET LIFECYCLE POLICY

The global ticket policy applies to all service tickets and to users that do not have any per-user ticket policies defined.

The following procedure describes adjusting the maximum ticket lifetime and maximum ticket renewal age for the global Kerberos ticket policy using the `ipa krbtpolicy-mod` command.

While using the `ipa krbtpolicy-mod` command, specify at least one of the following arguments:

- `--maxlife` for the maximum ticket lifetime in seconds
- `--maxrenew` for the maximum renewable age in seconds

**Procedure**

1. To modify the global ticket policy:

   ```bash
   [root@server ~]# ipa krbtpolicy-mod --maxlife=$((8*60*60)) --maxrenew=$((24*60*60))
   Max life: 28800
   Max renew: 86400
   ```

   In this example, the maximum lifetime is set to eight hours (8 * 60 minutes * 60 seconds) and the maximum renewal age is set to one day (24 * 60 minutes * 60 seconds).

2. Optional: To reset the global Kerberos ticket policy to the default installation values:

   ```bash
   [root@server ~]# ipa krbtpolicy-reset
   Max life: 86400
   Max renew: 604800
   ```

**Verification steps**

- Display the global ticket policy:

  ```bash
  [root@server ~]# ipa krbtpolicy-show
  Max life: 28800
  Max renew: 86640
  ```

**Additional resources**

- To adjust the default ticket policy for a single user, see Configuring the default ticket policy for a user.
- To configure individual ticket policies for each authentication indicator for a single user, see Configuring individual authentication indicator ticket policies for a user.

### 42.6. CONFIGURING GLOBAL TICKET POLICIES PER AUTHENTICATION INDICATOR
This procedure describes adjusting the global maximum ticket lifetime and maximum renewable age for each authentication indicator. These settings apply to users that do not have per-user ticket policies defined.

Use the `ipa krbtpolicy-mod` command to specify the global maximum lifetime or maximum renewable age for Kerberos tickets depending on the authentication indicators attached to them.

**Procedure**

- For example, to set the global two-factor ticket lifetime and renewal age values to one week, and the global smart card ticket lifetime and renewal age values to two weeks:

```
[root@server ~]# ipa krbtpolicy-mod --otp-maxlife=604800 --otp-maxrenew=604800 --pkinit-maxlife=172800 --pkinit-maxrenew=172800
```

**Verification steps**

- Display the global ticket policy:

```
[root@server ~]# ipa krbtpolicy-show
Max life: 86400
OTP max life: 604800
PKINIT max life: 172800
Max renew: 604800
OTP max renew: 604800
PKINIT max renew: 172800
```

Notice that the OTP and PKINIT values are different from the global default Max life and Max renew values.

**Additional resources**

- For a list of authentication indicator options for the `ipa krbtpolicy-mod` command, see *Authentication indicator options for the krbtpolicy-mod command*.

- To adjust the default ticket policy for a single user, see *Configuring the default ticket policy for a user*.

- To configure individual ticket policies for each authentication indicator for a single user, see *Configuring individual authentication indicator ticket policies for a user*.

### 42.7. CONFIGURING THE DEFAULT TICKET POLICY FOR A USER

You can define a unique Kerberos ticket policy that only applies to a single user. These per-user settings override the global ticket policy, for all authentication indicators.

Use the `ipa krbtpolicy-mod username` command, and specify at least one of the following arguments:

- `--maxlife` for the maximum ticket lifetime in seconds

- `--maxrenew` for the maximum renewable age in seconds

**Procedure**
1. For example, to set the IdM `admin` user’s maximum ticket lifetime to two days and maximum renewal age to two weeks:

   ```bash
   [root@server ~]# ipa krbtpolicy-mod admin --maxlife=172800 --maxrenew=1209600
   Max life: 172800
   Max renew: 1209600
   ```

2. Optional: To reset the ticket policy for a user:

   ```bash
   [root@server ~]# ipa krbtpolicy-reset admin
   ```

**Verification steps**

- Display the effective Kerberos ticket policy that applies to a user:

  ```bash
  [root@server ~]# ipa krbtpolicy-show admin
  Max life: 172800
  Max renew: 1209600
  ```

**Additional resources**

- To adjust the global ticket policy for all users, see [Configuring the global ticket lifecycle policy](#).
- To configure different default ticket policies per authentication indicator, see [Configuring global ticket policies per authentication indicator](#).

### 42.8. Configuring Individual Authentication Indicator Ticket Policies for a User

As an administrator, you can define Kerberos ticket policies for a user that differ per authentication indicator. For example, you can configure a policy to allow the IdM `admin` user to renew a ticket for two days if it was obtained with OTP authentication, and a week if it was obtained with smart card authentication.

These per-authentication indicator settings will override the user’s default ticket policy, the `global` default ticket policy, and any `global` authentication indicator ticket policy.

Use the `ipa krbtpolicy-mod username` command to set custom maximum lifetime and maximum renewable age values for a user’s Kerberos tickets depending on the *authentication indicators* attached to them.

**Procedure**

1. For example, to allow the IdM `admin` user to renew a Kerberos ticket for two days if it was obtained with One-Time Password authentication, set the `--otp-maxrenew` option:

   ```bash
   [root@server ~]# ipa krbtpolicy-mod admin --otp-maxrenew=$(2*24*60*60)
   OTP max renew: 172800
   ```

2. Optional: To reset the ticket policy for a user:

   ```bash
   [root@server ~]# ipa krbtpolicy-reset username
   ```
Verification steps

- Display the effective Kerberos ticket policy that applies to a user:

```
[root@server ~]# ipa krbtpolicy-show admin
Max life: 28800
Max renew: 86640
```

Additional resources

- For a list of authentication indicator options for the `ipa krbtpolicy-mod` command, see [Authentication indicator options for the `krbtpolicy-mod` command](#).
- To adjust the default ticket policy for a single user, see [Configuring the default ticket policy for a user](#).
- To adjust the global ticket policy for all users, see [Configuring the global ticket lifecycle policy](#).
- To configure different global ticket policies per authentication indicator, see [Configuring global ticket policies per authentication indicator](#).

### 42.9. AUTHENTICATION INDICATOR OPTIONS FOR THE `KRBTPOLICY-MOD` COMMAND

Specify values for authentication indicators with the following arguments.

<table>
<thead>
<tr>
<th>Authentication indicator</th>
<th>Argument for maximum lifetime</th>
<th>Argument for maximum renewal age</th>
</tr>
</thead>
<tbody>
<tr>
<td>otp</td>
<td><code>--otp-maxlife</code></td>
<td><code>--otp-maxrenew</code></td>
</tr>
<tr>
<td>radius</td>
<td><code>--radius-maxlife</code></td>
<td><code>--radius-maxrenew</code></td>
</tr>
<tr>
<td>pkinit</td>
<td><code>--pkinit-maxlife</code></td>
<td><code>--pkinit-maxrenew</code></td>
</tr>
<tr>
<td>hardened</td>
<td><code>--hardened-maxlife</code></td>
<td><code>--hardened-maxrenew</code></td>
</tr>
</tbody>
</table>

[1] A hardened password is protected against brute-force password dictionary attacks by using Single-Party Public-Key Authenticated Key Exchange (SPAKE) pre-authentication and/or Flexible Authentication via Secure Tunneling (FAST) armoring.
CHAPTER 43. DEFINING IDM PASSWORD POLICIES

This chapter describes Identity Management (IdM) password policies and how to add a new password policy in IdM using an Ansible playbook.

43.1. WHAT IS A PASSWORD POLICY

A password policy is a set of rules that passwords must meet. For example, a password policy can define the minimum password length and the maximum password lifetime. All users affected by this policy are required to set a sufficiently long password and change it frequently enough to meet the specified conditions. In this way, password policies help reduce the risk of someone discovering and misusing a user’s password.

43.2. PASSWORD POLICIES IN IDM

Passwords are the most common way for Identity Management (IdM) users to authenticate to the IdM Kerberos domain. Password policies define the requirements that these IdM user passwords must meet.

NOTE

The IdM password policy is set in the underlying LDAP directory, but the Kerberos Key Distribution Center (KDC) enforces the password policy.

Password policy attributes lists the attributes you can use to define a password policy in IdM.

Table 43.1. Password Policy Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
</table>
| Max lifetime  | The maximum amount of time in days that a password is valid before a user must reset it. | Max lifetime = 90  
User passwords are valid only for 90 days. After that, IdM prompts users to change them. |
| Min lifetime  | The minimum amount of time in hours that must pass between two password change operations. | Min lifetime = 1  
After users change their passwords, they must wait at least 1 hour before changing them again. |
| History size  | The number of previous passwords that are stored. A user cannot reuse a password from their password history but can reuse old passwords that are not stored. | History size = 0  
In this case, the password history is empty and users can reuse any of their previous passwords. |
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
</table>
| Character classes    | The number of different character classes the user must use in the password. The character classes are: * Uppercase characters * Lowercase characters * Digits * Special characters, such as comma (,), period (.), asterisk (*) * Other UTF-8 characters Using a character three or more times in a row decreases the character class by one. For example:  
  * Secret1 has 3 character classes: uppercase, lowercase, digits  
  * Secret111 has 2 character classes: uppercase, lowercase, digits, and a -1 penalty for using 1 repeatedly | Character classes = 0  
The default number of classes required is 0. To configure the number, run the **ipa pwpolicy-mod** command with the **--minclasses** option.  
See also the **Important** note below this table. |
| Min length           | The minimum number of characters in a password. If any of the **additional password policy options** are set, then the minimum length of passwords is 6 regardless of the value to which the Min length option is set. | Min length = 8  
Users cannot use passwords shorter than 8 characters. |
| Max failures         | The maximum number of failed login attempts before IdM locks the user account.                                                                                                                                  | Max failures = 6  
IdM locks the user account when the user enters a wrong password 7 times in a row. |
| Failure reset interval| The amount of time in seconds after which IdM resets the current number of failed login attempts.                                                                                                               | Failure reset interval = 60  
If the user waits for more than 1 minute after the number of failed login attempts defined in **Max failures**, the user can attempt to log in again without risking a user account lock. |
| Lockout duration     | The amount of time in seconds that the user account is locked after the number of failed login attempts defined in **Max failures**.                                                                            | Lockout duration = 600  
Users with locked accounts are unable to log in for 10 minutes. |
IMPORTANT

Use the English alphabet and common symbols for the character classes requirement if you have a diverse set of hardware that may not have access to international characters and symbols. For more information about character class policies in passwords, see What characters are valid in a password? in Red Hat Knowledgebase.

43.3. ENSURING THE PRESENCE OF A PASSWORD POLICY IN IDM USING AN ANSIBLE PLAYBOOK

This section describes how to ensure the presence of a password policy in Identity Management (IdM) using an Ansible playbook.

In the default global_policy password policy in IdM, the number of different character classes in the password is set to 0. The history size is also set to 0.

Complete this procedure to enforce a stronger password policy for an IdM group using an Ansible playbook.

NOTE

You can only define a password policy for an IdM group. You cannot define a password policy for an individual user.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller.
- You know the IdM administrator password.
- The group for which you are ensuring the presence of a password policy exists in IdM.

Procedure

1. Create an inventory file, for example inventory.file, and define the FQDN of your IdM server in the [ipaserver] section:

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

2. Create your Ansible playbook file that defines the password policy whose presence you want to ensure. To simplify this step, copy and modify the example in the /usr/share/doc/ansible-freeipa/playbooks/pwpolicy/pwpolicy_present.yml file:

   ```
   ---
   - name: Tests
     hosts: ipaserver
     become: true

     tasks:
       - name: Ensure presence of pwpolicy for group ops
         ipapwpolicy:
           ipaadmin_password: MySecret123
           name: ops
   ```
minlife: 7
maxlife: 49
history: 5
priority: 1
lockouttime: 300
minlength: 8
minclasses: 4
maxfail: 3
failinterval: 5

For details on what the individual variables mean, see Password policy attributes.

3. Run the playbook:

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file path_to_playbooks_directory_/new_pwpolicy_present.yml
```

You have successfully used an Ansible playbook to ensure that a password policy for the ops group is present in IdM.

**IMPORTANT**

The priority of the ops password policy is set to 1, whereas the global_policy password policy has no priority set. For this reason, the ops policy automatically supersedes global_policy for the ops group and is enforced immediately.

global_policy serves as a fallback policy when no group policy is set for a user, and it can never take precedence over a group policy.

Additional resources

- For more details about using Ansible to define password policies in IdM and about playbook variables, see the README-pwpolicy.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory.

- For more details about how password policy priorities work in IdM, see Password policy priorities in RHEL 7 documentation.

### 43.4. ADDITIONAL PASSWORD POLICY OPTIONS IN IDM

As an Identity Management (IdM) administrator, you can strengthen the default password requirements by enabling additional password policy options based on the libpwquality feature set. The additional password policy options include the following:

---maxrepeat

Specifies the maximum acceptable number of same consecutive characters in the new password.

---maxsequence

Specifies the maximum length of monotonic character sequences in the new password. Examples of such a sequence are 12345 or fedcb. Most such passwords will not pass the simplicity check. The only exception is when the sequence is only a minor part of the password.

--dictcheck

If nonzero, checks whether the password, with possible modifications, matches a word in a dictionary. Currently libpwquality performs the dictionary check using the cracklib library.
--usercheck

If nonzero, checks whether the password, with possible modifications, contains the user name in some form. It is not performed for user names shorter than 3 characters.

You cannot apply the additional password policy options to existing passwords. If you apply any of the additional options, IdM automatically sets the --minlength option, the minimum number of characters in a password, to 6 characters.

NOTE

In a mixed environment with RHEL 7, RHEL 8, and RHEL 9 servers, you can enforce the additional password policy settings only on servers running on RHEL 8.4 and later. If a user is logged in to an IdM client and the IdM client is communicating with an IdM server running on RHEL 8.3 or earlier, then the new password policy requirements set by the system administrator will not be applied. To ensure consistent behavior, upgrade or update all servers to RHEL 8.4 and later.

Additional resources:

- Applying additional password policies to an IdM group
- pwquality(3) man page

43.5. APPLYING ADDITIONAL PASSWORD POLICY OPTIONS TO AN IDM GROUP

This section describes how to apply additional password policy options in Identity Management (IdM). The example describes how to strengthen the password policy for the managers group by making sure that the new passwords do not contain the users’ respective user names and that the passwords contain no more than two identical characters in succession.

Prerequisites

- You are logged in as an IdM administrator.
- The managers group exists in IdM.
- The managers password policy exists in IdM.

Procedure

1. Apply the user name check to all new passwords suggested by the users in the managers group:

   $ ipa pwpolicy-mod --usercheck=True managers

   NOTE

   If you do not specify the name of the password policy, the default global_policy is modified.

2. Set the maximum number of identical consecutive characters to 2 in the managers password policy:
$ ipa pwpolicy-mod --maxrepeat=2 managers

A password now will not be accepted if it contains more than 2 identical consecutive characters. For example, the **eR873mUi111YJQ** combination is unacceptable because it contains three **1s** in succession.

**Verification**

1. Add a test user named **test_user**:

   $ ipa user-add test_user
   First name: test
   Last name: user
   ---------------------------
   Added user “test_user”
   ---------------------------

2. Add the test user to the **managers** group:
   a. In the IdM Web UI, click **Identity** → **Groups** → **User Groups**.
   b. Click **managers**.
   c. Click **Add**.
   d. In the **Add users into user group 'managers'** page, check **test_user**.
   e. Click the **>** arrow to move the user to the **Prospective** column.
   f. Click **Add**.

3. Reset the password for the test user:
   a. Go to **Identity** → **Users**.
   b. Click **test_user**.
   c. In the **Actions** menu, click **Reset Password**.
   d. Enter a temporary password for the user.

4. On the command line, try to obtain a Kerberos ticket-granting ticket (TGT) for the **test_user**:

   $ kinit test_user
   a. Enter the temporary password.
   b. The system informs you that you must change your password. Enter a password that contains the user name of **test_user**:

   Password expired. You must change it now.
   Enter new password:
   Enter it again:
**NOTE**

Kerberos does not have fine-grained error password policy reporting and, in certain cases, does not provide a clear reason why a password was rejected.

---

**c.** The system informs you that the entered password was rejected. Enter a password that contains three or more identical characters in succession:

```
Password change rejected: Password not changed.
Unspecified password quality failure while trying to change password.
Please try again.

Enter new password:
Enter it again:
```

**d.** The system informs you that the entered password was rejected. Enter a password that meets the criteria of the **managers** password policy:

```
Password change rejected: Password not changed.
Unspecified password quality failure while trying to change password.
Please try again.

Enter new password:
Enter it again:
```

---

**5.** View the obtained TGT:

```
$ klist
Ticket cache: KCM:0:33945
Default principal: test_user@IDM.EXAMPLE.COM

Valid starting       Expires              Service principal
07/07/2021 12:44:44  07/08/2021 12:44:44
krbtgt@IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
```

The **managers** password policy now works correctly for users in the **managers** group.

**Additional resources**

- Additional password policies in IdM
CHAPTER 44. MANAGING EXPIRING PASSWORD NOTIFICATIONS

You can use the Expiring Password Notification (EPN) tool, provided by the ipa-client-epn package, to build a list of Identity Management (IdM) users whose passwords are expiring in a configured amount of time. To install, configure, and use the EPN tool, refer to the relevant sections.

- What is the Expiring Password Notification tool
- Installing the Expiring Password Notification tool
- Running the EPN tool to send emails to users whose passwords are expiring
- Enabling the ipa-epn.timer to send an email to all users whose passwords are expiring
- Modifying the Expiring Password Notification email template

44.1. WHAT IS THE EXPIRING PASSWORD NOTIFICATION TOOL

The Expiring Password Notification (EPN) tool is a standalone tool you can use to build a list of Identity Management (IdM) users whose passwords are expiring in a configured amount of time.

IdM administrators can use EPN to:

- Display a list of affected users in JSON format, which is created when run in dry-run mode.
- Calculate how many emails will be sent for a given day or date range.
- Send password expiration email notifications to users.
- Configure the ipa-epn.timer to run the EPN tool daily and send an email to users whose passwords are expiring within the defined future date ranges.
- Customize the email notification to send to users.

NOTE

If a user account is disabled, no email notifications are sent if the password is going to expire.

44.2. INSTALLING THE EXPIRING PASSWORD NOTIFICATION TOOL

This procedure describes how to install the Expiring Password Notification (EPN) tool.

Prerequisites

- Install the EPN tool on either an Identity Management (IdM) replica or an IdM client with a local Postfix SMTP server configured as a smart host.

Procedure

- Install the EPN tool:

  # dnf install ipa-client-epn
44.3. RUNNING THE EPN TOOL TO SEND EMAILS TO USERS WHOSE PASSWORDS ARE EXPIRING

This procedure describes how to run the Expiring Password Notification (EPN) tool to send emails to users whose passwords are expiring.

NOTE

The EPN tool is stateless. If the EPN tool fails to email any of the users whose passwords are expiring on a given day, the EPN tool does not save a list of those users.

Prerequisites

- The ipa-client-epn package is installed. See Installing the Expiring Password Notification tool.
- Customize the ipa-epn email template if required. See Modifying the Expiring Password Notification email template.

Procedure

1. Update the epn.conf configuration file to set the options for the EPN tool to notify users of upcoming password expiration.

   # vi /etc/ipa/epn.conf

2. Update the notify_ttls as required. The default is to notify users whose passwords are expiring in 28, 14, 7, 3, and 1 day(s).

   notify_ttls = 28, 14, 7, 3, 1

3. Configure your SMTP server and port:

   smtp_server = localhost
   smtp_port = 25

4. Specify the email address from which the email expiration notification is sent. Any unsuccessfully delivered emails are returned to this address.

   mail_from = admin-email@example.com

5. Save the /etc/ipa/epn.conf file.

6. Run the EPN tool in dry-run mode to generate a list of the users to whom the password expiration email notification would be sent if you run the tool without the --dry-run option.

   ipa-epn --dry-run
   [
   {  
      "uid": "user5",
      "cn": "user 5",
      "krbpasswordexpiration": "2020-04-17 15:51:53",
      "mail": ["user5@ipa.test"]
   }
   ]
The IPA-EPN command was successful

NOTE
If the list of users returned is very large and you run the tool without the **--dry-run** option, this might cause an issue with your email server.

7. Run the EPN tool without the **--dry-run** option to send expiration emails to the list of all the users returned when you ran the EPN tool in dry-run mode:

```bash
ipa-epn
```

```json
[
{
    "uid": "user5",
    "cn": "user 5",
    "krbpasswordexpiration": "2020-10-01 15:51:53",
    "mail": "[user5@ipa.test]"
}
]
```

The IPA-EPN command was successful

8. You can add EPN to any monitoring system and invoke it with the **--from-nbdays** and **--to-nbdays** options to determine how many users passwords are going to expire within a specific time frame:

```bash
# ipa-epn --from-nbdays 8 --to-nbdays 12
```

NOTE
If you invoke the EPN tool with the **--from-nbdays** and **--to-nbdays** options, it is automatically executed in dry-run mode.

Verification steps

- Run the EPN tool and verify an email notification is sent.

Additional resources
44.4. ENABLING THE IPA-EPN.TIMER TO SEND AN EMAIL TO ALL USERS WHOSE PASSWORDS ARE EXPIRING

This procedure describes how to use `ipa-epn.timer` to run the Expiring Password Notification (EPN) tool to send emails to users whose passwords are expiring. The `ipa-epn.timer` parses the `epn.conf` file and sends an email to users whose passwords are expiring within the defined future date ranges configured in that file.

Prerequisites

- The `ipa-client-epn` package is installed. See Installing the Expiring Password Notification tool
- Customize the `ipa-epn` email template if required. See Modifying the Expiring Password Notification email template

Procedure

- Start the `ipa-epn.timer`:
  ```bash
  systemctl start ipa-epn.timer
  ```

Once you start the timer, by default, the EPN tool is run every day at 1 a.m.

Additional resources

- See the `ipa-epn` man page.

44.5. MODIFYING THE EXPIRING PASSWORD NOTIFICATION EMAIL TEMPLATE

This procedure describes how to customize the Expiring Password Notification (EPN) email message template.

Prerequisites

- The `ipa-client-epn` package is installed.

Procedure

1. Open the EPN message template:
   ```bash
   # vi /etc/ipa/epn/expire_msg.template
   ```
2. Update the template text as required.
   ```text
   Hi {{ fullname }},
   ```
Your password will expire on {{ expiration }}.

Please change it as soon as possible.

You can use the following variables in the template.

- User ID: uid
- Full name: fullname
- First name: first
- Last name: last
- Password expiration date: expiration

3. Save the message template file.

Verification steps

- Run the EPN tool and verify the email notification contains the updated text.

Additional resources

- See the ipa-epn man page.
CHAPTER 45. GRANTING SUDO ACCESS TO AN IDM USER ON AN IDM CLIENT

45.1. 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

- For details on creating local `sudo` rules on a RHEL host, see Managing `sudo` access.

45.2. 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.
- 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.
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:
**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.

   ```bash
   [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
   KR5CCNAME",
   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:

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

### 45.3. 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.

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.

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`.

   ![Figure 45.1. Adding IdM sudo command](image)

   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 45.2. Adding IdM sudo rule

<table>
<thead>
<tr>
<th>Who</th>
<th>Access this host</th>
<th>Run Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>User category the rule applies to:</td>
<td>Host category the rule applies to:</td>
<td>Command category the rule applies to:</td>
</tr>
<tr>
<td>Users</td>
<td>Internal</td>
<td>Any Command</td>
</tr>
<tr>
<td>idm_user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Groups</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host Groups</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

**45.4. 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.
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.

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`.

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

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

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

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

   ```bash
   [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:

   ```bash
   [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:

   ```bash
   [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
   ---------------------------------
   
   [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
   ---------------------------------
   
   [root@idmclient ~]# ipa sudorule-add-user run_third-party-app_report --users idm_user
   Rule name: run_third-party-app_report
   Enabled: TRUE
   Users: idm_user
   Hosts: idmclient.idm.example.com
   Sudo Allow Commands: /opt/third-party-app/bin/report
   RunAs External User: thirdpartyapp
   ---------------------------------
   Number of members added 1
   
   NOTE
   Propagating the changes from the server to the client can take a few minutes.

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

   ```bash
   [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
   ---------------------------------
   
   [root@idmclient ~]# ipa sudorule-add-user run_third-party-app_report --users idm_user
   Rule name: run_third-party-app_report
   Enabled: TRUE
   Users: idm_user
   Hosts: idmclient.idm.example.com
   Sudo Allow Commands: /opt/third-party-app/bin/report
   RunAs External User: thirdpartyapp
   ---------------------------------
   Number of members added 1
   
   NOTE
   Propagating the changes from the server to the client can take a few minutes.

Verification steps

1. Log in to the `idmclient` host as the `idm_user` account.

2. Test the new sudo rule:
   a. Display which `sudo` rules the `idm_user` account is allowed to perform.

   ```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,
   ```
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.

[sudo] password for idm_user@idm.example.com:
Executing report...
Report successful.

45.5. CREATING A SUOD 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.
- 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.
- 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.
Figure 45.3. Details of the sudo rule

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,
```
User idm_user@idm.example.com may run the following commands on idmclient:
(第三partyapp) /opt/third-party-app/bin/report

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

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

45.6. 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.

3. Save and close the /etc/sssd/sssd.conf file.

4. Restart the SSSD service to load the configuration changes.

[thirdpartyapp]@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
```

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.
**45.7. 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

- 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

### 45.8. 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

45.9. 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 able to resolve the correct realm for the service ticket based on the host name:

  Server not found in Kerberos database

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

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

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

  No Kerberos credentials available

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

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

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

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

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

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

  [idm-user@idm-client ~]$ cat /etc/sssd/sssd.conf
  ...
  pam_gssapi_check_upn = false

- For additional troubleshooting, you can enable debugging output for the pam_sss_gss.so PAM module.

  Add the debug option at the end of all pam_sss_gss.so entries in PAM files, such as /etc/pam.d/sudo.
/etc/pam.d/sudo and /etc/pam.d/sudo-i:

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

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

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

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

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

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

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

**Prerequisites**

- You have installed the `ansible-freeipa` package on the Ansible controller.
- You know the IdM administrator password.
You have ensured the presence of a user account for `idm_user` in IdM and unlocked the account by creating a password for the user. For details on adding a new IdM user using the command-line interface, see Adding users using the command line.

No local `idm_user` account exists on `idmclient`. The `idm_user` user is not listed in the `/etc/passwd` file on `idmclient`.

Procedure

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

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

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

   ```
   ---
   - name: Playbook to manage sudo command
     hosts: ipaserver
     become: true
     tasks:
       # Ensure sudo command is present
       - ipasudocmd:
           ipaadmin_password: MySecret123
           name: /usr/sbin/reboot
           state: present
   ```
   b. Run the playbook:

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

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

   ```
   ---
   ```
- name: Tests
  hosts: ipaserver
  become: true

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

b. Run the playbook:

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

**Verification steps**

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

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

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

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

**Additional materials**

- For more details on how to apply `sudo` commands, command groups, and rules in IdM using an Ansible playbook including the descriptions of playbook variables, see the README-sudocmd.md, README-sudocmdgroup.md, and README-sudorule.md Markdown files available in the `/usr/share/doc/ansible-freeipa/` directory.
CHAPTER 46. 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).

46.1. 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.

46.2. ENSURING THE PRESENCE OF AN HBAC RULE IN IDM USING AN ANSIBLE PLAYBOOK

This section describes how to ensure the presence of a host-based access control (HBAC) rule in Identity Management (IdM) using an Ansible playbook.

Prerequisites

- The `ansible-freeipa` package is installed on the Ansible controller.
- You know the IdM administrator password.
- The users and user groups you want to use for your HBAC rule exist in IdM. See Managing user accounts using Ansible playbooks and Ensuring the presence of IdM groups and group members using Ansible playbooks for details.
- The hosts and host groups to which you want to apply your HBAC rule exist in IdM. See Managing hosts using Ansible playbooks and Managing host groups using Ansible playbooks for details.

Procedure

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

   ```
   [ipaserver]
   server.idm.example.com
   ```
2. Create your Ansible playbook file that defines the HBAC policy whose presence you want to ensure. To simplify this step, you can copy and modify the example in the `/usr/share/doc/ansible-freeipa/playbooks/hbacrule/ensure-hbacrule-allhosts-present.yml` file:

```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

- For more details about and examples of, configuring HBAC services, service groups, and rules using Ansible, see the README-hbacsvc.md, README-hbacsvcgroup.md, and README-hbacrule.md Markdown files. These files are available in the `/usr/share/doc/ansible-freeipa` directory. Also see the playbooks available in the relevant subdirectories of the `/usr/share/doc/ansible-freeipa/playbooks` directory.
CHAPTER 47. MANAGING REPLICATION TOPOLOGY

This chapter describes how to manage replication between servers in an Identity Management (IdM) domain.

47.1. EXPLAINING REPLICATION AGREEMENTS, TOPOLOGY SUFFIXES AND TOPOLOGY SEGMENTS

This section explains the following concepts:

- Replication agreements
- Topology suffixes
- Topology segments

Replication agreements
When an administrator creates a replica based on an existing server, Identity Management (IdM) creates a replication agreement between the initial server and the replica. The replication agreement ensures that the data and configuration is continuously replicated between the two servers.

IdM uses multiple read/write replica replication. In this configuration, all replicas joined in a replication agreement receive and provide updates, and are therefore considered suppliers and consumers. Replication agreements are always bilateral.

Figure 47.1. Server and replica agreements

IdM uses two types of replication agreements:

- Domain replication agreements
  These agreements replicate the identity information.

- Certificate replication agreements
  These agreements replicate the certificate information.

Both replication channels are independent. Two servers can have one or both types of replication agreements configured between them. For example, when server A and server B have only domain replication agreement configured, only identity information is replicated between them, not the certificate information.

Topology suffixes
Topology suffixes store the data that is replicated. IdM supports two types of topology suffixes: domain and ca. Each suffix represents a separate server, a separate replication topology.
When a replication agreement is configured, it joins two topology suffixes of the same type on two different servers.

**The domain suffix: dc=example,dc=com**

The domain suffix contains all domain-related data. When two replicas have a replication agreement between their domain suffixes, they share directory data, such as users, groups, and policies.

**The ca suffix: o=ipaca**

The ca suffix contains data for the Certificate System component. It is only present on servers with a certificate authority (CA) installed. When two replicas have a replication agreement between their ca suffixes, they share certificate data.

**Figure 47.2. Topology suffixes**

An initial topology replication agreement is set up between two servers by the `ipa-replica-install` script when installing a new replica.

**Example 47.1. Viewing topology suffixes**

The `ipa topologysuffix-find` command displays a list of topology suffixes:

```
$ ipa topologysuffix-find
---------------------------
2 topology suffixes matched
---------------------------
Suffix name: ca
Managed LDAP suffix DN: o=ipaca

Suffix name: domain
Managed LDAP suffix DN: dc=example,dc=com
---------------------------
Number of entries returned 2
---------------------------
```

**Topology segments**
When two replicas have a replication agreement between their suffixes, the suffixes form a **topology segment**. Each topology segment consists of a **left node** and a **right node**. The nodes represent the servers joined in the replication agreement.

Topology segments in IdM are always bidirectional. Each segment represents two replication agreements: from server A to server B, and from server B to server A. The data is therefore replicated in both directions.

**Figure 47.3. Topology segments**

Example 47.2. Viewing topology segments

The **ipa topologysegment-find** command shows the current topology segments configured for the domain or CA suffixes. For example, for the domain suffix:

```
$ ipa topologysegment-find
Suffix name: domain
1 segment matched
Segment name: server1.example.com-to-server2.example.com
Left node: server1.example.com
Right node: server2.example.com
Connectivity: both
Number of entries returned 1
```

In this example, domain-related data is only replicated between two servers: **server1.example.com** and **server2.example.com**.

To display details for a particular segment only, use the **ipa topologysegment-show** command:

```
$ ipa topologysegment-show
Suffix name: domain
Segment name: server1.example.com-to-server2.example.com
Segment name: server1.example.com-to-server2.example.com
Left node: server1.example.com
Right node: server2.example.com
Connectivity: both
```
47.2. USING THE TOPOLOGY GRAPH TO MANAGE REPLICATION TOPOLOGY

The topology graph in the web UI shows the relationships between the servers in the domain. Using the Web UI, you can manipulate and transform the representation of the topology.

Accessing the topology graph
To access the topology graph:

1. Select IPA Server → Topology → Topology Graph.
2. If you make any changes to the topology that are not immediately reflected in the graph, click Refresh.

Interpreting the topology graph
Servers joined in a domain replication agreement are connected by an orange arrow. Servers joined in a CA replication agreement are connected by a blue arrow.

Topology graph example: recommended topology

Figure 47.4, "Recommended topology example" shows one of the possible recommended topologies for four servers: each server is connected to at least two other servers, and more than one server is a CA master.

![Recommended topology example](image)

Topology graph example: discouraged topology

In Figure 47.5, "Discouraged topology example: Single Point of Failure", server1 is a single point of failure. All the other servers have replication agreements with this server, but not with any of the other servers. Therefore, if server1 fails, all the other servers will become isolated. Avoid creating topologies like this.
Customizing the topology view
You can move individual topology nodes by dragging the mouse:

You can zoom in and zoom out the topology graph using the mouse wheel:
47.3. SETTING UP REPLICATION BETWEEN TWO SERVERS USING THE WEB UI

Using the Web interface of Identity Management (IdM) you can choose two servers and create new replication agreement between them.

Prerequisites

- You have the IdM administrator credentials.

Procedure

1. In the topology graph, hover your mouse over one of the server nodes.
2. Click on the **domain** or the **ca** part of the circle depending on what type of topology segment you want to create.

3. A new arrow representing the new replication agreement appears under your mouse pointer. Move your mouse to the other server node, and click on it.

4. In the **Add topology segment** window, click **Add** to confirm the properties of the new segment.

The new topology segment between the two servers joins them in a replication agreement. The topology graph now shows the updated replication topology:
47.4. STOPPING REPLICATION BETWEEN TWO SERVERS USING THE WEB UI

Using the web interface of Identity Management (IdM) you can remove a replication agreement from servers.

Prerequisites

- You have the IdM administrator credentials.

Procedure

1. Click on an arrow representing the replication agreement you want to remove. This highlights the arrow.

2. Click Delete.

3. In the Confirmation window, click OK.
IdM removes the topology segment between the two servers, which deletes their replication agreement. The topology graph now shows the updated replication topology:

Figure 47.13. Topology segment deleted

47.5. SETTING UP REPLICATION BETWEEN TWO SERVERS USING THE CLI

You can configure replication agreements between two servers using the `ipa topologysegment-add` command.

**Prerequisites**

- You have the IdM administrator credentials.

**Procedure**

1. Use the `ipa topologysegment-add` command to create a topology segment for the two servers. When prompted, provide:
   - the required topology suffix: `domain` or `ca`
   - the left node and the right node, representing the two servers
   - optionally, a custom name for the segment
   
   For example:
   
   ```
   $ ipa topologysegment-add
   Suffix name: domain
   Left node: server1.example.com
   Right node: server2.example.com
   Segment name [server1.example.com-to-server2.example.com]: new_segment
   ---------------------------
   Added segment "new_segment"
   ---------------------------
   Segment name: new_segment
   Left node: server1.example.com
   Right node: server2.example.com
   Connectivity: both
   ```

   Adding the new segment joins the servers in a replication agreement.
2. Optional. Use the `ipa topologysegment-show` command to verify that the new segment is configured.

```
$ ipa topologysegment-show
Suffix name: domain
Segment name: new_segment
Segment name: new_segment
Left node: server1.example.com
Right node: server2.example.com
Connectivity: both
```

### 47.6. STOPPING REPLICATION BETWEEN TWO SERVERS USING THE CLI

You can terminate replication agreements from command line using the `ipa topology_segment-del` command.

**Prerequisites**

- You have the IdM administrator credentials.

**Procedure**

1. To stop replication, you must delete the corresponding replication segment between the servers. To do that, you need to know the segment name. If you do not know the name, use the `ipa topologysegment-find` command to display all segments, and locate the required segment in the output. When prompted, provide the required topology suffix: `domain` or `ca`. For example:

```
$ ipa topologysegment-find
Suffix name: domain
------------------
8 segments matched
------------------
  Segment name: new_segment
  Left node: server1.example.com
  Right node: server2.example.com
  Connectivity: both
...
Number of entries returned 8
------------------
```

2. Use the `ipa topologysegment-del` command to remove the topology segment joining the two servers.

```
$ ipa topologysegment-del
Suffix name: domain
Segment name: new_segment
```
Deleting the segment removes the replication agreement.

3. Optional. Use the `ipa topologysegment-find` command to verify that the segment is no longer listed.

```
$ ipa topologysegment-find
Suffix name: domain
-------------------
7 segments matched
-------------------
Segment name: server2.example.com-to-server3.example.com
Left node: server2.example.com
Right node: server3.example.com
Connectivity: both
...
-------------------
Number of entries returned 7
-------------------
```

47.7. REMOVING SERVER FROM TOPOLOGY USING THE WEB UI

You can use Identity Management (IdM) web interface to remove a server from the topology.

**Prerequisites**

- You have the IdM administrator credentials.
- The server you want to remove is **not** the only server connecting other servers with the rest of the topology; this would cause the other servers to become isolated, which is not allowed.
- The server you want to remove is **not** your last CA or DNS server.

**WARNING**

Removing a server is an irreversible action. If you remove a server, the only way to introduce it back into the topology is to install a new replica on the machine.

**Procedure**

To remove a server from the topology without uninstalling the server components from the machine:

1. Select **IPA Server → Topology → IPA Servers**.
2. Click on the name of the server you want to delete.
3. Click **Delete Server**.

### 47.8. REMOVING SERVER FROM TOPOLOGY USING THE CLI

You can use the command line interface to remove a server from the topology.

**Prerequisites**

- You have the IdM administrator credentials.
- The server you want to remove is **not** the only server connecting other servers with the rest of the topology; this would cause the other servers to become isolated, which is not allowed.
- The server you want to remove is **not** your last CA or DNS server.

**IMPORTANT**

Removing a server is an irreversible action. If you remove a server, the only way to introduce it back into the topology is to install a new replica on the machine.

**Procedure**

To remove **server1.example.com**:

1. On another server, run the `ipa server-del` command to remove `server1.example.com`. The command removes all topology segments pointing to the server:

   ```bash
   [user@server2 ~]$ ipa server-del
   Server name: server1.example.com
   Removing server1.example.com from replication topology, please wait...
   ----------------------------------------------------------
   Deleted IPA server "server1.example.com"
   ----------------------------------------------------------
   ```

2. **Optional**: on **server1.example.com**, run the `ipa server-install --uninstall` command to uninstall the server components from the machine.

   ```bash
   [root@server1 ~]# ipa server-install --uninstall
   ```
47.9. VIEWING SERVER ROLES ON AN IDM SERVER USING THE WEB UI

Based on the services installed on an IdM server, it can perform various server roles. For example:

- CA server
- DNS server
- Key recovery authority (KRA) server.

For a complete list of the supported server roles, see **IPA Server → Topology → Server Roles**.

**NOTE**

- Role status **absent** means that no server in the topology is performing the role.
- Role status **enabled** means that one or more servers in the topology are performing the role.

47.10. VIEWING SERVER ROLES ON AN IDM SERVER USING THE CLI

Based on the services installed on an IdM server, it can perform various server roles. For example:

- CA server
- DNS server
- Key recovery authority (KRA) server.

You can view which servers perform which roles in the topology using the following commands.

- The **ipa config-show** command displays all CA servers and the current CA renewal server:

  ```bash
  $ ipa config-show
  ...
  IPA masters: server1.example.com, server2.example.com, server3.example.com
  IPA CA servers: server1.example.com, server2.example.com
  IPA CA renewal master: server1.example.com
  ```

- The **ipa server-show** command displays a list of roles enabled on a particular server. For example, for a list of roles enabled on server.example.com:
$ ipa server-show
Server name: server.example.com
...  
*Enabled server roles: CA server, DNS server, KRA server*

- The **ipa server-find --servrole** searches for all servers with a particular server role enabled. For example, to search for all CA servers:

  $ ipa server-find --servrole "CA server"

  2 IPA servers matched

  Server name: server1.example.com
  ...

  Server name: server2.example.com
  ...

  Number of entries returned 2

### 47.11. PROMOTING A REPLICA TO A CA RENEWAL SERVER AND CRL PUBLISHER SERVER

If your IdM deployment uses an embedded certificate authority (CA), one of the IdM CA servers acts as the CA renewal server, a server that manages the renewal of CA subsystem certificates. One of the IdM CA servers also acts as the IdM CRL publisher server, a server that generates certificate revocation lists. By default, the CA renewal server and CRL publisher server roles are installed on the first server on which the system administrator installed the CA role using the **ipa-server-install** or **ipa-ca-install** command.

**Prerequisites**

- You have the IdM administrator credentials.

**Procedure**

- Change the current CA renewal server.
- Configure replica to generate CRLs.

### 47.12. DEMOTING OR PROMOTING HIDDEN REPLICAS

After a replica has been installed, you can configure whether the replica is hidden or visible.

For details about hidden replicas, see [The hidden replica mode](#).

If the replica is a CA renewal server, move the service to another replica before making this replica hidden. For details, see [Changing and resetting IdM CA renewal server](#).

**Procedure**

- To hide the replica, enter:
# ipa server-state replica.idm.example.com --state=hidden

Alternatively, you can make the replica visible with the following command:

# ipa server-state replica.idm.example.com --state=enabled
CHAPTER 48. PUBLIC KEY CERTIFICATES IN IDENTITY MANAGEMENT

This chapter describes X.509 public key certificates, which are used to authenticate users, hosts and services in Identity Management (IdM). In addition to authentication, X.509 certificates also enable digital signing and encryption to provide privacy, integrity and non-repudiation.

A certificate contains the following information:

- The subject that the certificate authenticates.
- The issuer, that is the CA that has signed the certificate.
- The start and end date of the validity of the certificate.
- The valid uses of the certificate.
- The public key of the subject.

A message encrypted by the public key can only be decrypted by a corresponding private key. While a certificate and the public key it includes can be made publicly available, the user, host or service must keep their private key secret.

48.1. CERTIFICATE AUTHORITIES IN IDM

Certificate authorities operate in a hierarchy of trust. In an IdM environment with an internal Certificate Authority (CA), all the IdM hosts, users and services trust certificates that have been signed by the CA. Apart from this root CA, IdM supports sub-CAs to which the root CA has granted the ability to sign certificates in their turn. Frequently, the certificates that such sub-CAs are able to sign are certificates of a specific kind, for example VPN certificates. Finally, IdM supports using external CAs. The table below presents the specifics of using the individual types of CA in IdM.

Table 48.1. Comparison of using integrated and external CAs in IdM

<table>
<thead>
<tr>
<th>Name of CA</th>
<th>Description</th>
<th>Use</th>
<th>Useful links</th>
</tr>
</thead>
<tbody>
<tr>
<td>The <strong>ipa</strong> CA</td>
<td>An integrated CA based on the Dogtag upstream project</td>
<td>Integrated CAs can create, revoke, and issue certificates for users, hosts, and services.</td>
<td>Using the ipa CA to request a new user certificate and exporting it to the client</td>
</tr>
<tr>
<td>IdM sub-CAs</td>
<td>An integrated CA that is subordinate to the <strong>ipa</strong> CA</td>
<td>IdM sub-CAs are CAs to which the <strong>ipa</strong> CA has granted the ability to sign certificates. Frequently, these certificates are of a specific kind, for example VPN certificates.</td>
<td>Restricting an application to trust only a subset of certificates</td>
</tr>
<tr>
<td>External CAs</td>
<td>An external CA is a CA other than the integrated IdM CA or its sub-CAs.</td>
<td>Using IdM tools, you add certificates issued by these CAs to users, services, or hosts as well as remove them.</td>
<td>Managing certificates issued by external CAs in RHEL 7 documentation</td>
</tr>
</tbody>
</table>
From the certificate point of view, there is no difference between being signed by a self-signed IdM CA and being signed externally.

The role of the CA includes the following purposes:

- It issues digital certificates.
- By signing a certificate, it certifies that the subject named in the certificate owns a public key. The subject can be a user, host or service.
- It can revoke certificates, and provides revocation status via Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP).

Additional resources

- For more details on the supported CA configurations of the IdM server, see Planning your CA services.

### 48.2. COMPARISON OF CERTIFICATES AND KERBEROS

Certificates perform a similar function to that performed by Kerberos tickets. Kerberos is a computer network authentication protocol that works on the basis of tickets to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. The following table shows a comparison of Kerberos and X.509 certificates:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Kerberos</th>
<th>X.509</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Privacy</td>
<td>Optional</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrity</td>
<td>Optional</td>
<td>Yes</td>
</tr>
<tr>
<td>Type of cryptography involved</td>
<td>Symmetrical</td>
<td>Asymmetrical</td>
</tr>
<tr>
<td>Default validity</td>
<td>Short (1 day)</td>
<td>Long (2 years)</td>
</tr>
</tbody>
</table>

By default, Kerberos in Identity Management only ensures the identity of the communicating parties.

### 48.3. THE PROS AND CONS OF USING CERTIFICATES TO AUTHENTICATE USERS IN IDM

The advantages of using certificates to authenticate users in IdM include the following points:

- A PIN that protects the private key on a smart card is typically less complex and easier to remember than a regular password.
- Depending on the device, a private key stored on a smart card cannot be exported. This provides additional security.
• Smart cards can make logout automatic: IdM can be configured to log out users when they remove the smart card from the reader.

• Stealing the private key requires actual physical access to a smart card, making smart cards secure against hacking attacks.

• Smart card authentication is an example of two-factor authentication: it requires both something you have (the card) and something you know (the PIN).

• Smart cards are more flexible than passwords because they provide the keys that can be used for other purposes, such as encrypting email.

• Using smart cards use on shared machines that are IdM clients does not typically pose additional configuration problems for system administrators. In fact, smart card authentication is an ideal choice for shared machines.

The disadvantages of using certificates to authenticate users in IdM include the following points:

• Users might lose or forget to bring their smart card or certificate and be effectively locked out.

• Mistyping a PIN multiple times might result in a card becoming locked.

• There is generally an intermediate step between request and authorization by some sort of security officer or approver. In IdM, the security officer or administrator must run the `ipa cert-request` command.

• Smart cards and readers tend to be vendor and driver specific: although a lot of readers can be used for different cards, a smart card of a specific vendor might not work in the reader of another vendor or in the type of a reader for which it was not designed.

• Certificates and smart cards have a steep learning curve for administrators.
CHAPTER 49. MANAGING CERTIFICATES FOR USERS, HOSTS, AND SERVICES USING THE INTEGRATED IDM CA

This chapter describes how to manage certificates in Identity Management (IdM) using the integrated CA, the ipa CA, and its sub-CAs.

This chapter contains the following sections:

- Requesting new certificates for a user, host, or service using the IdM Web UI.
- Requesting new certificates for a user, host, or service from the IdM CA using the IdM CLI:
  - Requesting new certificates for a user, host, or service from IdM CA using certutil
    - For a specific example of requesting a new user certificate from the IdM CA using the certutil utility and exporting it to an IdM client, see Requesting a new user certificate and exporting it to the client.
  - Requesting new certificates for a user, host, or service from IdM CA using openssl
- You can also request new certificates for a service from the IdM CA using the certmonger utility. For more information, see Requesting new certificates for a service from IdM CA using certmonger.

Prerequisites

- Your IdM deployment contains an integrated CA:
  - For information on how to plan your CA services in IdM, see Planning your CA services.
  - For information on how to install an IdM server with integrated DNS and integrated CA as the root CA, see Installing an IdM server: With integrated DNS, with an integrated CA as the root CA.
  - For information on how to install an IdM server with integrated DNS and an external CA as the root CA, see Installing an IdM server: With integrated DNS, with an external CA as the root CA.
  - For information on how to install an IdM server without integrated DNS and with an integrated CA as the root CA, see Installing an IdM server: Without integrated DNS, with an integrated CA as the root CA.
  - [Optional] Your IdM deployment supports users authenticating with a certificate:
    - For information on how to configure your IdM deployment to support user authentication with a certificate stored in the IdM client filesystem, see Configuring authentication with a certificate stored on the desktop of an IdM client.
    - For information on how to configure your IdM deployment to support user authentication with a certificate stored on a smart card inserted into an IdM client, see Configuring Identity Management for smart card authentication.
    - For information on how to configure your IdM deployment to support user authentication with smart cards issued by an Active Directory certificate system, see Configuring certificates issued by ADCS for smart card authentication in IdM.
49.1. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE USING IDM WEB UI

This section describes how to use the Identity Management (IdM) Web UI to request a new certificate for any IdM entity from the integrated IdM certificate authorities (CAs): the ipa CA or any of its sub-CAs.

IdM entities include:

- Users
- Hosts
- Services

**IMPORTANT**

Services typically run on dedicated service nodes on which the private keys are stored. Copying a service’s private key to the IdM server is considered insecure. Therefore, when requesting a certificate for a service, create the certificate signing request (CSR) on the service node.

**Prerequisites**

- Your IdM deployment contains an integrated CA.
- You are logged into the IdM Web UI as the IdM administrator.

**Procedure**

1. Under the **Identity** tab, select the **Users**, **Hosts**, or **Services** subtab.
2. Click the name of the user, host, or service to open its configuration page.
3. Click **Actions** → **New Certificate**.
4. Optional: Select the issuing CA and profile ID.
5. Follow the instructions for using the **certutil** command-line (CLI) utility on the screen.
6. Click **Issue**.

49.2. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE FROM IDM CA USING CERTUTIL

You can use the **certutil** utility to request a certificate for an Identity Management (IdM) user, host or service in standard IdM situations. To ensure that a host or service Kerberos alias can use a certificate, use the **openssl** utility to request a certificate instead.

This section describes how to request a certificate for an IdM user, host, or service from **ipa**, the IdM certificate authority (CA), using **certutil**.

**IMPORTANT**

Services typically run on dedicated service nodes on which the private keys are stored. Copying a service’s private key to the IdM server is considered insecure. Therefore, when requesting a certificate for a service, create the certificate signing request (CSR) on the service node.

Prerequisites

- Your IdM deployment contains an integrated CA.
- You are logged into the IdM command-line interface (CLI) as the IdM administrator.

Procedure

1. Create a temporary directory for the certificate database:

   ```
   # mkdir ~/certdb/
   ```

2. Create a new temporary certificate database, for example:

   ```
   # certutil -N -d ~/certdb/
   ```

3. Create the CSR and redirect the output to a file. For example, to create a CSR for a 4096 bit certificate and to set the subject to `CN=server.example.com,O=EXAMPLE.COM`:

   ```
   # certutil -R -d ~/certdb/ -a -g 4096 -s "CN=server.example.com,O=EXAMPLE.COM" -8 server.example.com > certificate_request.csr
   ```

4. Submit the certificate request file to the CA running on the IdM server. Specify the Kerberos principal to associate with the newly-issued certificate:

   ```
   # ipa cert-request certificate_request.csr --principal=host/server.example.com
   ```

The **ipa cert-request** command in IdM uses the following defaults:

- The **calPAserviceCert** certificate profile
  To select a custom profile, use the **--profile-id** option.

- The integrated IdM root CA, **ipa**
  To select a sub-CA, use the **--ca** option.

Additional resources
49.3. REQUESTING NEW CERTIFICATES FOR A USER, HOST, OR SERVICE FROM IDM CA USING OPENSSL

You can use the openssl utility to request a certificate for an Identity Management (IdM) host or service if you want to ensure that the Kerberos alias of the host or service can use the certificate. In standard situations, consider requesting a new certificate using the certutil utility instead.

This section describes how to request a certificate for an IdM host, or service from ipa, the IdM certificate authority, using openssl.

IMPORTANT

Services typically run on dedicated service nodes on which the private keys are stored. Copying a service's private key to the IdM server is considered insecure. Therefore, when requesting a certificate for a service, create the certificate signing request (CSR) on the service node.

Prerequisites

- Your IdM deployment contains an integrated CA.
- You are logged into the IdM command-line interface (CLI) as the IdM administrator.

Procedure

1. Create one or more aliases for your Kerberos principal test/server.example.com. For example, test1/server.example.com and test2/server.example.com.

2. In the CSR, add a subjectAltName for dnsName (server.example.com) and otherName (test2/server.example.com). To do this, configure the openssl.conf file to include the following line specifying the UPN otherName and subjectAltName:

   ```
   otherName=1.3.6.1.4.1.311.20.2.3;UTF8:test2/server.example.com@EXAMPLE.COM
   DNS.1 = server.example.com
   ```

3. Create a certificate request using openssl:

   ```
   openssl req -new -newkey rsa:2048 -keyout test2service.key -sha256 -nodes -out certificate_request.csr -config openssl.conf
   ```

4. Submit the certificate request file to the CA running on the IdM server. Specify the Kerberos principal to associate with the newly-issued certificate:

   ```
   # ipa cert-request certificate_request.csr --principal=host/server.example.com
   ```

The ipa cert-request command in IdM uses the following defaults:

- The calPAserviceCert certificate profile
To select a custom profile, use the `--profile-id` option.

- The integrated IdM root CA, `ipa`
  To select a sub-CA, use the `--ca` option.

**Additional resources**

- For more information about the `ipa cert-request` command, see the output of the `ipa cert-request --help` command.

- For more information about creating a custom certificate profile, see [Creating and managing certificate profiles in Identity Management](#).

### 49.4. ADDITIONAL RESOURCES

- For information on how to revoke certificates using the IdM CA, see [Revoking certificates with the integrated IdM CAs](#).

- For information on how to restore certificates using the IdM CA, see [Restoring certificates with the integrated IdM CAs](#).

- For information on how to restrict an application to trust only certificates that were issued by an IdM sub-CA, see [Restricting an application to trust only a subset of certificates](#).
CHAPTER 50. CONVERTING CERTIFICATE FORMATS TO WORK WITH IDM

This user story describes how to make sure that you as an IdM system administrator are using the correct format of a certificate with specific IdM commands. This is useful, for example, in the following situations:

- You are loading an external certificate into a user profile. For details, see Section 50.2, “Converting an external certificate to load into an IdM user account”.

- You are using an external CA certificate when configuring the IdM server for smart card authentication or configuring the IdM client for smart card authentication so that users can authenticate to IdM using smart cards with certificates on them that have been issued by the external certificate authority.

- You are exporting a certificate from an NSS database into a pkcs #12 format that includes both the certificate and the private key. For details, see Section 50.3.1, “Exporting a certificate and private key from an NSS database into a PKCS #12 file”.

50.1. CERTIFICATE FORMATS AND ENCODINGS IN IDM

Certificate authentication including smart card authentication in IdM proceeds by comparing the certificate that the user presents with the certificate, or certificate data, that are stored in the user’s IdM profile.

System configuration
What is stored in the IdM profile is only the certificate, not the corresponding private key. During authentication, the user must also show that he is in possession of the corresponding private key. The user does that by either presenting a PKCS #12 file that contains both the certificate and the private key or by presenting two files: one that contains the certificate and the other containing the private key.

Therefore, processes such as loading a certificate into a user profile only accept certificate files that do not contain the private key.

Similarly, when a system administrator provides you with an external CA certificate, he will provide only the public data: the certificate without the private key. The ipa-advise utility for configuring the IdM server or the IdM client for smart card authentication expects the input file to contain the certificate of the external CA but not the private key.

Certificate encodings
There are two common certificate encodings: Privacy-enhanced Electronic Mail (PEM) and Distinguished Encoding Rules (DER). The base64 format is almost identical to the PEM format but it does not contain the -----BEGIN CERTIFICATE-----/-----END CERTIFICATE----- header and footer.

A certificate that has been encoded using DER is a binary X509 digital certificate file. As a binary file, the certificate is not human-readable. DER files sometimes use the .der filename extension, but files with the .crt and .cer filename extensions also sometimes contain DER certificates. DER files containing keys can be named .key.

A certificate that has been encoded using PEM Base64 is a human-readable file. The file contains ASCII (Base64) armored data prefixed with a "-----BEGIN ..." line. PEM files sometimes use the .pem filename extension, but files with the .crt and .cer filename extensions also sometimes contain PEM certificates. PEM files containing keys can be named .key.
Different `ipa` commands have different limitations regarding the types of certificates that they accept. For example, the `ipa user-add-cert` command only accepts certificates encoded in the `base64` format but `ipa-server-certinstall` accepts PEM, DER, PKCS #7, PKCS #8 and PKCS #12 certificates.

<table>
<thead>
<tr>
<th>Encoding format</th>
<th>Human-readable</th>
<th>Common filename extensions</th>
<th>Sample IdM commands accepting the encoding format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEM/base64</td>
<td>Yes</td>
<td>.pem,.crt,.cer</td>
<td>ipa user-add-cert, ipa-server-certinstall, ...</td>
</tr>
<tr>
<td>DER</td>
<td>No</td>
<td>.der,.crt,.cer</td>
<td>ipa-server-certinstall, ...</td>
</tr>
</tbody>
</table>

Table 50.1. Certificate encodings

Section 50.4, “Certificate-related commands and formats in IdM” lists further `ipa` commands with the certificate formats that the commands accept.

**User authentication**

When using the web UI to access IdM, the user proves that he is in possession of the private key corresponding to the certificate by having both stored in the browser’s database.

When using the CLI to access IdM, the user proves that he is in possession of the private key corresponding to the certificate by one of the following methods:

- The user adds, as the value of the `X509_user_identity` parameter of the `kinit -X` command, the path to the smart card module that is connected to the smart card that contains both the certificate and the key:
  
  ```bash
  $ kinit -X X509_user_identity='PKCS11:opensc-pkcs11.so' idm_user
  ```

- The user adds two files as the values of the `X509_user_identity` parameter of the `kinit -X` command, one containing the certificate and the other the private key:
  
  ```bash
  $ kinit -X X509_user_identity='FILE:`/path/to/cert.pem,/path/to/cert.key`' idm_user
  ```

**Useful certificate commands**

To view the certificate data, such as the subject and the issuer:

```bash
$ openssl x509 -noout -text -in ca.pem
```

To compare in which lines two certificates differ:

```bash
$ diff cert1.crt cert2.crt
```

To compare in which lines two certificates differ with the output displayed in two columns:

```bash
$ diff cert1.crt cert2.crt -y
```

**50.2. CONVERTING AN EXTERNAL CERTIFICATE TO LOAD INTO AN IDM USER ACCOUNT**
This section describes how to make sure that an external certificate is correctly encoded and formatted before adding it to a user entry.

### Prerequisites

- If your certificate was issued by an Active Directory certificate authority and uses the **PEM** encoding, make sure that the **PEM** file has been converted into the **UNIX** format. To convert a file, use the `dos2unix` utility provided by the eponymous package.

### 50.2.1. Converting an external certificate in the IdM CLI and loading it into an IdM user account

The **IdM CLI** only accepts a **PEM** certificate from which the first and last lines (**-----BEGIN CERTIFICATE-----** and **-----END CERTIFICATE-----**) have been removed.

#### Procedure

1. Convert the certificate to the **PEM** format:
   - If your certificate is in the **DER** format:
     ```
     $ openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
     ```
   - If your file is in the **PKCS #12** format, whose common filename extensions are **.pfx** and **.p12**, and contains a certificate, a private key, and possibly other data, extract the certificate using the `openssl pkcs12` utility. When prompted, enter the password protecting the private key stored in the file:
     ```
     $ openssl pkcs12 -in cert_and_key.p12 -clcerts -nokeys -out cert.pem
     Enter Import Password:
     ```

2. Obtain the administrator’s credentials:
   ```
   $ kinit admin
   ```

3. Add the certificate to the user account using the **IdM CLI** following one of the following methods:
   - Remove the first and last lines (**-----BEGIN CERTIFICATE-----** and **-----END CERTIFICATE-----**) of the **PEM** file using the `sed` utility before adding the string to the **ipa user-add-cert** command:
     ```
     $ ipa user-add-cert some_user --certificate="$(sed -e '/BEGIN CERTIFICATE/d;/END CERTIFICATE/d' cert.pem)"
     ```
   - Copy and paste the contents of the certificate file without the first and last lines (**-----BEGIN CERTIFICATE-----** and **-----END CERTIFICATE-----**) into the **ipa user-add-cert** command:
     ```
     $ ipa user-add-cert some_user --
     certificate=MIIDlzCCAn+gAwIBAgIBATANBgkqhki...
     ```
NOTE

You cannot pass a PEM file containing the certificate as input to the `ipa user-add-cert` command directly, without first removing the first and last lines (-----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----):

```
$ ipa user-add-cert some_user --cert=some_user_cert.pem
```

This command results in the "ipa: ERROR: Base64 decoding failed: Incorrect padding" error message.

4. Optionally, to check if the certificate was accepted by the system:

```
[idm_user@r8server]$ ipa user-show some_user
```

50.2.2. Converting an external certificate in the IdM web UI for loading into an IdM user account:

**Procedure**

1. Using the CLI, convert the certificate to the PEM format:
   - If your certificate is in the DER format:
     ```
     $ openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
     ```
   - If your file is in the PKCS #12 format, whose common filename extensions are .pfx and .p12, and contains a certificate, a private key, and possibly other data, extract the certificate using the `openssl pkcs12` utility. When prompted, enter the password protecting the private key stored in the file:
     ```
     $ openssl pkcs12 -in cert_and_key.p12 -clcerts -nokeys -out cert.pem
     ```
     Enter Import Password:

2. Open the certificate in an editor and copy the contents. You can include the "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE-----" header and footer lines but you do not have to, as both the PEM and base64 formats are accepted by the IdM web UI.

3. In the IdM web UI, log in as security officer.

4. Go to **Identity → Users → some_user**.

5. Click **Add** next to **Certificates**.

6. Paste the PEM-formatted contents of the certificate into the window that opens.

7. Click **Add**.

If the certificate was accepted by the system, you can see it listed among the **Certificates** in the user profile.

**50.3. PREPARING TO LOAD A CERTIFICATE INTO THE BROWSER**
Before importing a user certificate into the browser, make sure that the certificate and the corresponding private key are in a **PKCS #12** format. There are two common situations requiring extra preparatory work:

- The certificate is located in an NSS database. For details how to proceed in this situation, see Section 50.3.1, “Exporting a certificate and private key from an NSS database into a PKCS #12 file”.

- The certificate and the private key are in two separate **PEM** files. For details how to proceed in this situation, see Section 50.3.2, “Combining certificate and private key PEM files into a PKCS #12 file”.

Afterwards, to import both the CA certificate in the **PEM** format and the user certificate in the **PKCS #12** format into the browser, follow the procedures in Configuring a browser to enable certificate authentication and Authenticating to the Identity Management Web UI with a Certificate as an Identity Management User.

---

### 50.3.1. Exporting a certificate and private key from an NSS database into a PKCS #12 file

**Procedure**

1. Use the `pk12util` command to export the certificate from the NSS database to the **PKCS12** format. For example, to export the certificate with the `some_user` nickname from the NSS database stored in the `~/certdb` directory into the `~/some_user.p12` file:

   ```
   $ pk12util -d ~/certdb -o ~/some_user.p12 -n some_user
   Enter Password or Pin for “NSS Certificate DB”:
   Enter password for PKCS12 file:
   Re-enter password:
   pk12util: PKCS12 EXPORT SUCCESSFUL
   ```

2. Set appropriate permissions for the `.p12` file:

   ```
   # chmod 600 ~/some_user.p12
   ```

   Because the **PKCS #12** file also contains the private key, it must be protected to prevent other users from using the file. Otherwise, they would be able to impersonate the user.

---

### 50.3.2. Combining certificate and private key PEM files into a PKCS #12 file

This section describes how to combine a certificate and the corresponding key stored in separate **PEM** files into a **PKCS #12** file.

**Procedure**

- To combine a certificate stored in `certfile.cer` and a key stored in `certfile.key` into a `certfile.p12` file that contains both the certificate and the key:

  ```
  $ openssl pkcs12 -export -in certfile.cer -inkey certfile.key -out certfile.p12
  ```

---

### 50.4. CERTIFICATE-RELATED COMMANDS AND FORMATS IN IDM
Table **IdM certificate commands and formats** displays certificate-related commands in IdM with acceptable formats.

**Table 50.2. IdM certificate commands and formats**

<table>
<thead>
<tr>
<th>Command</th>
<th>Acceptable formats</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipa user-add-cert some_user --certificate</td>
<td>base64 PEM certificate</td>
<td></td>
</tr>
<tr>
<td>ipa-server-certinstall</td>
<td>PEM and DER certificate; PKCS#7 certificate chain; PKCS#8 and raw private key; PKCS#12 certificate and private key</td>
<td></td>
</tr>
<tr>
<td>ipa-cacert-manage install</td>
<td>DER; PEM; PKCS#7</td>
<td></td>
</tr>
<tr>
<td>ipa-cacert-manage renew --external-cert-file</td>
<td>PEM and DER certificate; PKCS#7 certificate chain</td>
<td></td>
</tr>
<tr>
<td>ipa-ca-install --external-cert-file</td>
<td>PEM and DER certificate; PKCS#7 certificate chain</td>
<td></td>
</tr>
<tr>
<td>ipa cert-show &lt;cert serial&gt; --certificate-out /path/to/file.pem</td>
<td>N/A</td>
<td>Creates the PEM-encoded <em>file.pem</em> file with the certificate having the <code>&lt;cert_serial&gt;</code> serial number.</td>
</tr>
<tr>
<td>ipa cert-show &lt;cert serial&gt; --certificate-out /path/to/file.pem</td>
<td>N/A</td>
<td>Creates the PEM-encoded <em>file.pem</em> file with the certificate having the <code>&lt;cert_serial&gt;</code> serial number. If the <code>--chain</code> option is used, the PEM file contains the certificate including the certificate chain.</td>
</tr>
<tr>
<td>ipa cert-request --certificate-out=FILE /path/to/req.csr</td>
<td>N/A</td>
<td>Creates the <em>req.csr</em> file in the PEM format with the new certificate.</td>
</tr>
<tr>
<td>ipa cert-request --certificate-out=FILE /path/to/req.csr</td>
<td>N/A</td>
<td>Creates the <em>req.csr</em> file in the PEM format with the new certificate. If the <code>--chain</code> option is used, the PEM file contains the certificate including the certificate chain.</td>
</tr>
</tbody>
</table>
CHAPTER 51. CREATING AND MANAGING CERTIFICATE PROFILES IN IDENTITY MANAGEMENT

Certificate profiles are used by the Certificate Authority (CA) when signing certificates to determine if a certificate signing request (CSR) is acceptable, and if so what features and extensions are present on the certificate. A certificate profile is associated with issuing a particular type of certificate. By combining certificate profiles and CA access control lists (ACLs), you can define and control access to custom certificate profiles.

In describing how to create certificate profiles, the procedures use S/MIME certificates as an example. Some email programs support digitally signed and encrypted email using the Secure Multipurpose Internet Mail Extension (S/MIME) protocol. Using S/MIME to sign or encrypt email messages requires the sender of the message to have an S/MIME certificate.

- What is a certificate profile
- Creating a certificate profile
- What is a CA access control list
- Defining a CA ACL to control access to certificate profiles
- Using certificate profiles and CA ACLs to issue certificates
- Modifying a certificate profile
- Certificate profile configuration parameters

51.1. WHAT IS A CERTIFICATE PROFILE?

You can use certificate profiles to determine the content of certificates, as well as constraints for issuing the certificates, such as the following:

- The signing algorithm to use to encipher the certificate signing request.
- The default validity of the certificate.
- The revocation reasons that can be used to revoke a certificate.
- If the common name of the principal is copied to the subject alternative name field.
- The features and extensions that should be present on the certificate.

A single certificate profile is associated with issuing a particular type of certificate. You can define different certificate profiles for users, services, and hosts in IdM. IdM includes the following certificate profiles by default:

- calIPAserviceCert
- IECUserRoles
- KDCs_PKINIT_Certs (used internally)

In addition, you can create and import custom profiles, which allow you to issue certificates for specific purposes. For example, you can restrict the use of a particular profile to only one user or one group, preventing other users and groups from using that profile to issue a certificate for authentication.
create custom certificate profiles, use the ipa certprofile command.

Additional resources

- For information on the ipa certprofile command, run the ipa help certprofile command.

### 51.2. CREATING A CERTIFICATE PROFILE

This procedure describes how to create a certificate profile through the command line by creating a profile configuration file for requesting S/MIME certificates.

**Procedure**

1. Create a custom profile by copying an existing default profile:

   ```
   $ ipa certprofile-show --out smime.cfg caIPAserviceCert
   Profile configuration stored in file 'smime.cfg'
   Profile ID: caIPAserviceCert
   Profile description: Standard profile for network services
   Store issued certificates: TRUE
   ```

2. Open the newly created profile configuration file in a text editor.

   ```
   $ vi smime.cfg
   ```

3. Change the **Profile ID** to a name that reflects the usage of the profile, for example **smime**.

   **NOTE**

   When you are importing a newly created profile, the **profileId** field, if present, must match the ID specified on the command line.

4. Update the Extended Key Usage configuration. The default Extended Key Usage extension configuration is for TLS server and client authentication. For example for S/MIME, the Extended Key Usage must be configured for email protection:

   ```
   policyset.serverCertSet.7.default.params.exKeyUsageOIDs=1.3.6.1.5.5.7.3.4
   ```

5. Import the new profile:

   ```
   $ ipa certprofile-import smime --file smime.cfg --desc "S/MIME certificates" --store TRUE
   ------------------------
   Imported profile "smime"
   Profile ID: smime
   Profile description: S/MIME certificates
   Store issued certificates: TRUE
   ```
Verification steps

- Verify the new certificate profile has been imported:

```bash
$ ipa certprofile-find
```

```
------------------
4 profiles matched
------------------
Profile ID: caIPAserviceCert
Profile description: Standard profile for network services
Store issued certificates: TRUE

Profile ID: IECUserRoles
Profile description: User profile that includes IECUserRoles extension from request
Store issued certificates: TRUE

Profile ID: KDCs_PKINIT_Certs
Profile description: Profile for PKINIT support by KDCs
Store issued certificates: TRUE

Profile ID: smime
Profile description: S/MIME certificates
Store issued certificates: TRUE
```

```
Number of entries returned 4
```

Additional resources

- For details on the `certprofile` plug-in, run the `ipa help certprofile` command.

```
For more information on the Extended Key Usage extension, see RFC 5280, section 4.2.1.12.
```

51.3. WHAT IS A CA ACCESS CONTROL LIST?

Certificate Authority access control list (CA ACL) rules define which profiles can be used to issue certificates to which principals. You can use CA ACLs to do this, for example:

- Determine which user, host, or service can be issued a certificate with a particular profile

- Determine which IdM certificate authority or sub-CA is permitted to issue the certificate

For example, using CA ACLs, you can restrict use of a profile intended for employees working from an office located in London only to users that are members of the London office-related IdM user group.

The `ipa caacl` utility for management of CA ACL rules allows privileged users to add, display, modify, or delete a specified CA ACL.

Additional resources

- For information on the `ipa caacl` command, run the `ipa help caacl` command.
51.4. DEFINING A CA ACL TO CONTROL ACCESS TO CERTIFICATE PROFILES

This procedure describes how to use the caacl utility to define a CA Access Control List (ACL) rule to allow users in a group access to a custom certificate profile. In this case, the procedure describes how to create an S/MIME user's group and a CA ACL to allow users in that group access to the smime certificate profile.

Prerequisites

- Make sure that you have obtained IdM administrator’s credentials.

Procedure

1. Create a new group for the users of the certificate profile:

   ```
   $ ipa group-add smime_users_group
   -----------------------------
   Added group "smime users group"
   -----------------------------
   Group name: smime_users_group
   GID: 75400001
   ```

2. Create a new user to add to the smime_users_group group:

   ```
   $ ipa user-add smime_user
   First name: smime
   Last name: user
   ----------------------
   Added user "smime_user"
   ----------------------
   User login: smime_user
   First name: smime
   Last name: user
   Full name: smime user
   Display name: smime user
   Initials: TU
   Home directory: /home/smime_user
   GECOS: smime user
   Login shell: /bin/sh
   Principal name: smime_user@IDM.EXAMPLE.COM
   Principal alias: smime_user@IDM.EXAMPLE.COM
   Email address: smime_user@idm.example.com
   UID: 1505000004
   GID: 1505000004
   Password: False
   Member of groups: ipausers
   Kerberos keys available: False
   ```

3. Add the smime_user to the smime_users_group group:

   ```
   $ ipa group-add-member smime_users_group --users=smime_user
   Group name: smime_users_group
   GID: 1505000003
   ```
Member users: smime_user

Number of members added 1

4. Create the CA ACL to allow users in the group to access the certificate profile:

$ ipa caacl-add smime_acl

Added CA ACL “smime_acl”

ACL name: smime_acl
Enabled: TRUE

5. Add the user group to the CA ACL:

$ ipa caacl-add-user smime_acl --group smime_users_group

ACL name: smime_acl
Enabled: TRUE
User Groups: smime_users_group

Number of members added 1

6. Add the certificate profile to the CA ACL:

$ ipa caacl-add-profile smime_acl --certprofile smime

ACL name: smime_acl
Enabled: TRUE
Profiles: smime
User Groups: smime_users_group

Number of members added 1

Verification steps

- View the details of the CA ACL you created:

$ ipa caacl-show smime_acl

ACL name: smime_acl
Enabled: TRUE
Profiles: smime
User Groups: smime_users_group

Additional resources

- See ipa man page.

- For further details about the ipa caacl command, refer to the ipa help caacl command.
51.5. USING CERTIFICATE PROFILES AND CA ACLS TO ISSUE CERTIFICATES

You can request certificates using a certificate profile when permitted by the Certificate Authority access control lists (CA ACLs). This procedure describes how to request an S/MIME certificate for a user using a custom certificate profile which has been granted access through a CA ACL.

Prerequisites

- Your certificate profile has been created.
- An CA ACL has been created which permits the user to use the required certificate profile to request a certificate.

**NOTE**

You can bypass the CA ACL check if the user performing the `cert-request` command:
- Is the `admin` user.
- Has the **Request Certificate ignoring CA ACLs** permission.

Procedure

1. Generate a certificate request for the user. For example, using OpenSSL:

   ```
   $ openssl req -new -newkey rsa:2048 -days 365 -nodes -keyout private.key -out cert.csr -subj '/CN=smime_user'
   ```

2. Request a new certificate for the user from the IdM CA:

   ```
   $ ipa cert-request cert.csr --principal=smime_user --profile-id=smime
   ```

   Optionally pass the `--ca sub-CA_name` option to the command to request the certificate from a sub-CA instead of the root CA.

Verification steps

- Verify the newly-issued certificate is assigned to the user:

   ```
   $ ipa user-show user
   User login: user
   ...
   Certificate: MIICfzCCAWcCAQA...
   ```

Additional resources

- See `ipa(a)` man page.
- For further details about the `ipa user-show` command, refer to the `ipa help user-show` command.
51.6. MODIFYING A CERTIFICATE PROFILE

This procedure describes how to modify certificate profiles directly through the command line using the `ipa certprofile-mod` command.

Procedure

1. Determine the certificate profile ID for the certificate profile you are modifying. To display all certificate profiles currently stored in IdM:

   ```
   # ipa certprofile-find
   ------------
   4 profiles matched
   ------------
   Profile ID: caIPAserviceCert
   Profile description: Standard profile for network services
   Store issued certificates: TRUE
   
   Profile ID: IECUserRoles
   ...
   
   Profile ID: smime
   Profile description: S/MIME certificates
   Store issued certificates: TRUE
   ------------
   Number of entries returned
   ------------
   ```

2. Modify the certificate profile description. For example, if you created a custom certificate profile for S/MIME certificates using an existing profile, change the description in line with the new usage:

   ```
   # ipa certprofile-mod smime --desc "New certificate profile description"
   ------------
   Modified Certificate Profile "smime"
   ------------
   Profile ID: smime
   Profile description: New certificate profile description
   Store issued certificates: TRUE
   ```

3. Open your customer certificate profile file in a text editor and modify to suit your requirements:

   ```
   # vi smime.cfg
   ```

   For details on the options which can be configured in the certificate profile configuration file, see Certificate profile configuration parameters.

4. Update the existing certificate profile configuration file:
# ipa certprofile-mod _profile_ID_ --file=smime.cfg

**Verification steps**

- Verify the certificate profile has been updated:

```
$ ipa certprofile-show smime
Profile ID: smime
Profile description: New certificate profile description
Store issued certificates: TRUE
```

**Additional resources**

- See `ipa(a)` man page.
- For further details about the `ipa certprofile-mod` command, refer to the `ipa help certprofile-mod` command.

## 51.7. CERTIFICATE PROFILE CONFIGURATION PARAMETERS

Certificate profile configuration parameters are stored in a `profile_name.cfg` file in the CA profile directory, `/var/lib/pki/pki-tomcat/ca/profiles/ca`. All of the parameters for a profile – defaults, inputs, outputs, and constraints – are configured within a single policy set. A policy set for a certificate profile has the name `policyset.policyName.policyNumber`. For example, for policy set `serverCertSet`:

```
policyset.list=serverCertSet
policyset.serverCertSet.list=1,2,3,4,5,6,7,8
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=CN=^[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=CN=$request.req_subject_name.cn$, OU=pki-ipa, O=IPA
policyset.serverCertSet.2.constraint.class_id=validityConstraintImpl
policyset.serverCertSet.2.constraint.name=Validity Constraint
policyset.serverCertSet.2.constraint.params.range=740
policyset.serverCertSet.2.constraint.params.notBeforeCheck=false
policyset.serverCertSet.2.constraint.params.notAfterCheck=false
policyset.serverCertSet.2.default.class_id=validityDefaultImpl
policyset.serverCertSet.2.default.name=Validity Default
policyset.serverCertSet.2.default.params.range=731
policyset.serverCertSet.2.default.params.startTime=0
```

Each policy set contains a list of policies configured for the certificate profile by policy ID number in the order in which they should be evaluated. The server evaluates each policy set for each request it receives. When a single certificate request is received, one set is evaluated, and any other sets in the profile are ignored. When dual key pairs are issued, the first policy set is evaluated for the first certificate request, and the second set is evaluated for the second certificate request. You do not need more than one policy set when issuing single certificates or more than two sets when issuing dual key pairs.

**Table 51.1. Certificate profile configuration file parameters**

432
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>desc</td>
<td>A free text description of the certificate profile, which is shown on the end-entities page. For example, desc=<em>This certificate profile is for enrolling server certificates with agent authentication.</em></td>
</tr>
<tr>
<td>enable</td>
<td>Enables the profile so it is accessible through the end-entities page. For example, enable=<em>true.</em></td>
</tr>
<tr>
<td>auth.instance_id</td>
<td>Sets the authentication manager plug-in to use to authenticate the certificate request. For automatic enrollment, the CA issues a certificate immediately if the authentication is successful. If authentication fails or there is no authentication plug-in specified, the request is queued to be manually approved by an agent. For example, auth.instance_id=<em>AgentCertAuth.</em></td>
</tr>
<tr>
<td>authz.acl</td>
<td>Specifies the authorization constraint. This is predominantly used to set the group evaluation Access Control List (ACL). For example, the caCMCUserCert parameter requires that the signer of the CMC request belongs to the Certificate Manager Agents group: authz.acl=<em>group=&quot;Certificate Manager Agents</em></td>
</tr>
<tr>
<td>name</td>
<td>The name of the certificate profile. For example, name=<em>Agent-Authenticated Server Certificate Enrollment.</em> This name is displayed on the end users enrollment or renewal page.</td>
</tr>
<tr>
<td>input.list</td>
<td>Lists the allowed inputs for the certificate profile by name. For example, input.list=<em>i1,i2.</em></td>
</tr>
<tr>
<td>input.input_id.class_id</td>
<td>Indicates the java class name for the input by input ID (the name of the input listed in input.list). For example, input.i1.class_id=certReqInputImpl.</td>
</tr>
<tr>
<td>output.list</td>
<td>Lists the possible output formats for the certificate profile by name. For example, output.list=<em>o1.</em></td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>output.output_id.class_id</code></td>
<td>Specifies the java class name for the output format named in <code>output.list</code>. For example, <code>&lt;output.o1.class_id=certOutputImpl&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.list</code></td>
<td>Lists the configured certificate profile rules. For dual certificates, one set of rules applies to the signing key and the other to the encryption key. Single certificates use only one set of certificate profile rules. For example, <code>&lt;policyset.list=serverCertSet&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.list</code></td>
<td>Lists the policies within the policy set configured for the certificate profile by policy ID number in the order in which they should be evaluated. For example, <code>&lt;policyset.serverCertSet.list=1,2,3,4,5,6,7,8&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.constraint.class_id</code></td>
<td>Indicates the java class name of the constraint plug-in set for the default configured in the profile rule. For example, <code>&lt;policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.constraint.name</code></td>
<td>Gives the user-defined name of the constraint. For example, <code>&lt;policyset.serverCertSet.1.constraint.name=Subject Name Constraint&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.constraint.params.attribute</code></td>
<td>Specifies a value for an allowed attribute for the constraint. The possible attributes vary depending on the type of constraint. For example, <code>&lt;policyset.serverCertSet.1.constraint.params.attribute=CN=.*&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.default.class_id</code></td>
<td>Gives the java class name for the default set in the profile rule. For example, <code>&lt;policyset.serverCertSet.1.default.class_id=userSubjectNameDefaultImpl&gt;</code>.</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.default.name</code></td>
<td>Gives the user-defined name of the default. For example, <code>&lt;policyset.serverCertSet.1.default.name=Subject Name Default&gt;</code>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>policyset.policyset_id.policy_number.default.params.attribute</code></td>
<td>Specifies a value for an allowed attribute for the default. The possible attributes vary depending on the type of default. For example, <code>policyset.serverCertSet.1.default.params.name=CN=(Name)$request.requestor_name$</code>.</td>
</tr>
</tbody>
</table>
CHAPTER 52. MANAGING THE VALIDITY OF CERTIFICATES IN IDM

In Identity Management (IdM), you can manage the validity of both already existing certificates and certificates you want to issue in the future, but the methods are different.

Managing the validity of an existing certificate that was issued by IdM CA

In IdM, the following methods of viewing the expiry date of a certificate are available:

- Viewing the expiry date in IdM WebUI;
- Viewing the expiry date in the CLI.

You can manage the validity of an already existing certificate that was issued by IdM CA in the following ways:

- Renew a certificate by requesting a new certificate using either the original certificate signing request (CSR) or a new CSR generated from the private key. You can request a new certificate using the following utilities:
  - certmonger
    You can use certmonger to request a service certificate. Before the certificate is due to expire, certmonger will automatically renew the certificate, thereby ensuring a continuing validity of the service certificate. For details, see Obtaining an IdM certificate for a service using certmonger;
  - certutil
    You can use certutil to renew user, host, and service certificates. For details on requesting a user certificate, see Requesting a new user certificate and exporting it to the client;
  - openssl
    You can use openssl to renew user, host, and service certificates.
- Revoke a certificate. For details, see:
  - Revoking certificates with the integrated IdM CAs using IdM WebUI;
  - Revoking certificates with the integrated IdM CAs using IdM CLI;
- Restore a certificate if it has been temporarily revoked. For details, see:
  - Restoring certificates with the integrated IdM CAs using IdM WebUI;
  - Restoring certificates with the integrated IdM CAs using IdM CLI.

Managing the validity of future certificates issued by IdM CA

To manage the validity of future certificates issued by IdM CA, modify, import, or create a certificate profile. For details, see Creating and managing certificate profiles in Identity Management.

52.1. VIEWING THE EXPIRY DATE OF A CERTIFICATE

52.1.1. Viewing the expiry date of a certificate in IdM WebUI

You can use IdM WebUI to view the expiry date of all the certificates that have been issued by IdM CA.
Prerequisites

- Ensure that you have obtained the administrator’s credentials.

Procedure

1. In the Authentication menu, click Certificates > Certificates.

2. Click the serial number of the certificate to open the certificate information page.

Figure 52.1. List of Certificates

3. In the certificate information page, locate the Expires On information.

52.1.2. Viewing the expiry date of a certificate in the CLI

You can use the command-line interface (CLI) to view the expiry date of a certificate.

Procedure

- Use the openssl utility to open the file in a human-readable format:

```bash
$ openssl x509 -noout -text -in ca.pem
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: O = IDM.EXAMPLE.COM, CN = Certificate Authority
    Validity
      Not After : Oct 30 19:39:14 2037 GMT
```

52.2. REVOKING CERTIFICATES WITH THE INTEGRATED IDM CAS

52.2.1. Certificate revocation reasons

A revoked certificate is invalid and cannot be used for authentication. All revocations are permanent, except for reason 6: Certificate Hold.

The default revocation reason is 0: unspecified.
### Table 52.1. Revocation Reasons

<table>
<thead>
<tr>
<th>ID</th>
<th>Reason</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unspecified</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Key Compromised</td>
<td>The key that issued the certificate is no longer trusted. Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lost token, improperly accessed file.</td>
</tr>
<tr>
<td>2</td>
<td>CA Compromised</td>
<td>The CA that issued the certificate is no longer trusted.</td>
</tr>
<tr>
<td>3</td>
<td>Affiliation Changed</td>
<td>Possible causes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* A person has left the company or moved to another department.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* A host or service is being retired.</td>
</tr>
<tr>
<td>4</td>
<td>Superseded</td>
<td>A newer certificate has replaced the current certificate.</td>
</tr>
<tr>
<td>5</td>
<td>Cessation of Operation</td>
<td>The host or service is being decommissioned.</td>
</tr>
<tr>
<td>6</td>
<td>Certificate Hold</td>
<td>The certificate is temporarily revoked. You can restore the certificate later.</td>
</tr>
<tr>
<td>8</td>
<td>Remove from CRL</td>
<td>The certificate is not included in the certificate revocation list (CRL).</td>
</tr>
<tr>
<td>9</td>
<td>Privilege Withdraw</td>
<td>The user, host, or service is no longer permitted to use the certificate.</td>
</tr>
<tr>
<td>10</td>
<td>Attribute Authority (AA)</td>
<td>The AA certificate is no longer trusted.</td>
</tr>
<tr>
<td></td>
<td>Compromise</td>
<td></td>
</tr>
</tbody>
</table>

### 52.2.2. Revoking certificates with the integrated IdM CAs using IdM WebUI

If you know you have lost the private key for your certificate, you must revoke the certificate to prevent its abuse. Complete this procedure to use the IdM WebUI to revoke a certificate issued by the IdM CA.

**Procedure**

1. Click **Authentication > Certificates > Certificates**.

2. Click the serial number of the certificate to open the certificate information page.
3. In the certificate information page, click Actions → Revoke Certificate.

4. Select the reason for revoking and click Revoke. See Section 52.2.1, “Certificate revocation reasons” for details.

52.2.3. Revoking certificates with the integrated IdM CAs using IdM CLI

If you know you have lost the private key for your certificate, you must revoke the certificate to prevent its abuse. Complete this procedure to use the IdM CLI to revoke a certificate issued by the IdM CA.

Procedure

- Use the `ipa cert-revoke` command, and specify:
  - the certificate serial number
  - the ID number for the revocation reason; see Section 52.2.1, “Certificate revocation reasons” for details

For example, to revoke the certificate with serial number 1032 because of reason 1: Key Compromised, enter:

```
$ ipa cert-revoke 1032 --revocation-reason=1
```

For details on requesting a new certificate, see the following documentation:

- Requesting a new user certificate and exporting it to the client;
- Obtaining an IdM certificate for a service using certmonger.

52.3. RESTORING CERTIFICATES WITH THE INTEGRATED IDM CAS

If you have revoked a certificate because of reason 6: Certificate Hold, you can restore it again if the private key for the certificate has not been compromised. To restore a certificate, use one of the following procedures:

- Restore certificates with the integrated IdM CAs using IdM WebUI;
- Restore certificates with the integrated IdM CAs using IdM CLI.
52.3.1. Restoring certificates with the integrated IdM CAs using IdM WebUI

Complete this procedure to use the IdM WebUI to restore an IdM certificate that has been revoked because of Reason 6: **Certificate Hold**.

**Procedure**

1. In the **Authentication** menu, click **Certificates > Certificates**.

2. Click the serial number of the certificate to open the certificate information page.

   ![Figure 52.3. List of Certificates](image)

3. In the certificate information page, click **Actions → Restore Certificate**.

52.3.2. Restoring certificates with the integrated IdM CAs using IdM CLI

Complete this procedure to use the IdM CLI to restore an IdM certificate that has been revoked because of Reason 6: **Certificate Hold**.

**Procedure**

- Use the `ipa cert-remove-hold` command and specify the certificate serial number. For example:

  ```bash
  $ ipa cert-remove-hold 1032
  ```
CHAPTER 53. CONFIGURING IDENTITY MANAGEMENT FOR SMART CARD AUTHENTICATION

Authentication based on smart cards is an alternative to passwords. You can store user credentials on a smart card in the form of a private key and a certificate, and special software and hardware is used to access them. Place the smart card into a reader or a USB port and supply the PIN code for the smart card instead of providing your password.

Identity Management (IdM) supports smart card authentication with:

- User certificates issued by the IdM certificate authority
- User certificates issued by an external certificate authority

This user story shows how to set up smart card authentication in IdM for both types of certificates. In the user story, the `smartcard_ca.pem` CA certificate is the file containing the certificate of a trusted external certificate authority.

The user story contains the following modules:

- Configuring the IdM server for smart card authentication
- Configuring the IdM client for smart card authentication
- Adding a certificate to a user entry in the IdM Web UI
- Adding a certificate to a user entry in the IdM CLI
- Installing tools for managing and using smart cards
- Storing a certificate on a smart card
- Logging in to IdM with smart cards
- Configuring GDM access using smart card authentication
- Configuring su access using smart card authentication

53.1. CONFIGURING THE IDM SERVER FOR SMART CARD AUTHENTICATION

If you want to enable smart card authentication for users whose certificates have been issued by the certificate authority of the `EXAMPLE.ORG` domain, whose LDAP distinguished name (DN) is `CN=Certificate Authority,DC=EXAMPLE,DC=ORG`, then you need to obtain the certificate of the authority so that you can run it with the script configuring the IdM server. You can, for example, download the certificate from a web page whose certificate has been issued by the authority. For details, see Steps 1 - 4a in Configuring a browser to enable certificate authentication.

To enable smart card authentication for IdM users who have been issued a certificate by the IdM Certificate Authority, obtain the CA certificate from the `/etc/ipa/ca.crt` file on the IdM server on which the IdM CA is running.

This section describes how to configure an IdM server for smart card authentication. First, obtain files with the CA certificates in the PEM format, then run the in-built `ipa-advise` script. Finally, reload the system configuration.
Prerequisites

- You have root access to the IdM server.
- You have the root CA certificate and any sub CA certificates.

Procedure

1. Create a directory in which you will do the configuration:

   [root@server#] `mkdir ~/SmartCard/`

2. Navigate to the directory:

   [root@server#] `cd ~/SmartCard/`

3. Obtain the relevant CA certificates stored in files in PEM format. If your CA certificate is stored in a file of a different format, such as DER, convert it to PEM format. The IdM Certificate Authority certificate is located in the `/etc/ipa/ca.crt` file.

   Convert a DER file to a PEM file:

   ```
   # openssl x509 -in <filename>.der -inform DER -out <filename>.pem -outform PEM
   ```

4. For convenience, copy the certificates to the directory in which you want to do the configuration:

   [root@server SmartCard#] `cp /etc/ipa/ca.crt ~/SmartCard/`
   [root@server SmartCard#] `cp /tmp/smartcard_ca.pem ~/SmartCard/`

5. Optionally, if you use certificates of external certificate authorities, use the `openssl x509` utility to view the contents of the files in the PEM format to check that the Issuer and Subject values are correct:

   [root@server SmartCard#] `openssl x509 -noout -text -in smartcard_ca.pem | more`

6. Generate a configuration script with the in-built `ipa-advise` utility, using the administrator’s privileges:

   [root@server SmartCard#] `kinit admin`
   [root@server SmartCard#] `sudo ipa-advise config-server-for-smart-card-auth > config-server-for-smart-card-auth.sh`

   The `config-server-for-smart-card-auth.sh` script performs the following actions:

   - It configures the IdM Apache HTTP Server.
   - It enables Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) on the Key Distribution Center (KDC).
   - It configures the IdM Web UI to accept smart card authorization requests.

7. Execute the script, adding the PEM files containing the root CA and sub CA certificates as arguments:
NOTE

Ensure that you add the root CA's certificate as an argument before any sub CA certificates and that the CA or sub CA certificates have not expired.

8. Optionally, if the certificate authority that issued the user certificate does not provide any Online Certificate Status Protocol (OCSP) responder, you may need to disable OCSP check for authentication to the IdM Web UI:

   a. Set the `SSLOCSPEnable` parameter to `off` in the `/etc/httpd/conf.d/ssl.conf` file:
      
      ```
      SSLOCSPEnable off
      ```

   b. Restart the Apache daemon (httpd) for the changes to take effect immediately:
      
      ```
      [root@server SmartCard]# sudo systemctl restart httpd
      ```

   WARNING

   Do not disable the OCSP check if you only use user certificates issued by the IdM CA. OCSP responders are part of IdM.

For instructions on how to keep the OCSP check enabled, and yet prevent a user certificate from being rejected by the IdM server if it does not contain the information about the location at which the CA that issued the user certificate listens for OCSP service requests, see the `SSLOCSPDefaultResponder` directive in Apache mod_ssl configuration options.

The server is now configured for smart card authentication.

NOTE

To enable smart card authentication in the whole topology, run the procedure on each IdM server.

53.2. CONFIGURING THE IDM CLIENT FOR SMART CARD AUTHENTICATION

This section describes how to configure IdM clients for smart card authentication. The procedure needs
to be run on each IdM system, a client or a server, to which you want to connect while using a smart card for authentication. For example, to enable an ssh connection from host A to host B, the script needs to be run on host B.

As an administrator, run this procedure to enable smart card authentication using

- the ssh protocol
  For details see Configuring SSH access using smart card authentication.
- the console login
- the Gnome Display Manager (GDM)
- the su command

This procedure is not required for authenticating to the IdM Web UI. Authenticating to the IdM Web UI involves two hosts, neither of which needs to be an IdM client:

- the machine – possibly outside of the IdM domain – on which the browser is running
- the IdM server on which httpd is running

The following procedure assumes that you are configuring smart card authentication on an IdM client, not an IdM server. For this reason you need two computers: an IdM server to generate the configuration script, and the IdM client on which to run the script.

Prerequisites

- Your IdM server has been configured for smart card authentication, as described in Configuring the IdM server for smart card authentication.
- You have root access to the IdM server and the IdM client.
- You have access to the root CA certificate and any sub CA certificates.
- You installed the IdM client with the --mkhomedir option to ensure remote users can log in successfully. If you do not create a home directory, the default login location is root.

Procedure

1. On an IdM server, generate a configuration script with ipa-advise using the administrator’s privileges:

   [root@server SmartCard]# kinit admin
   [root@server SmartCard]# ipa-advise config-client-for-smart-card-auth > config-client-for-smart-card-auth.sh

   The config-client-for-smart-card-auth.sh script performs the following actions:

   - It configures the smart card daemon.
   - It sets the system-wide trust store.
   - It configures the System Security Services Daemon (SSSD) to allow smart card logins to the desktop.
2. From the IdM server, copy the script to a directory of your choice on the IdM client machine:

```
[root@server SmartCard]# scp config-client-for-smart-card-auth.sh root@client.idm.example.com:/root/SmartCard/
Password: 
config-client-for-smart-card-auth.sh        100%   2419       3.5MB/s   00:00
```

3. From the IdM server, copy the CA certificate files in the PEM format for convenience to the same directory on the IdM client machine as used in the previous step:

```
[root@server SmartCard]# scp {smartcard_ca.pem,ca.crt} root@client.idm.example.com:/root/SmartCard/
Password: 
smartcard_ca.pem                    100%   1237     9.6KB/s   00:00
c.crt                              100%   2514    19.6KB/s   00:00
```

4. On the client machine, execute the script, adding the PEM files containing the CA certificates as arguments:

```
[root@client SmartCard]# kinit admin
[root@client SmartCard]# chmod +x config-client-for-smart-card-auth.sh
[root@client SmartCard]# ./config-client-for-smart-card-auth.sh smartcard_ca.pem ca.crt
Ticket cache:KEYRING:persistent:0:0
Default principal: admin@IDM.EXAMPLE.COM
[...] 
Systemwide CA database updated.
The ipa-certupdate command was successful
```

**NOTE**

Ensure that you add the root CA's certificate as an argument before any sub CA certificates and that the CA or sub CA certificates have not expired.

The client is now configured for smart card authentication.

### 53.3. ADDING A CERTIFICATE TO A USER ENTRY IN THE IDM WEB UI

This procedure describes how to add an external certificate to a user entry in IdM Web UI.

Instead of uploading the whole certificate, it is also possible to upload certificate mapping data to a user entry in IdM. User entries containing either full certificates or certificate mapping data can be used in conjunction with corresponding certificate mapping rules to facilitate the configuration of smart card authentication for system administrators. For details, see Certificate mapping rules for configuring authentication on smart cards

**NOTE**

If the user's certificate has been issued by the IdM Certificate Authority, the certificate is already stored in the user entry, and you can skip this section.

**Prerequisites**

- You have the certificate that you want to add to the user entry at your disposal.
Procedure

1. Log into the IdM Web UI as an administrator if you want to add a certificate to another user. For adding a certificate to your own profile, you do not need the administrator’s credentials.

2. Navigate to **Users → Active users → sc_user**.

3. Find the **Certificate** option and click **Add**.

4. In the **Command-Line Interface**, display the certificate in the **PEM** format using the **cat** utility or a text editor:

   ```
   [user@client SmartCard]$ cat testuser.crt
   ```

5. Copy and paste the certificate from the CLI into the window that has opened in the Web UI.

6. Click **Add**.

**Figure 53.1. Adding a new certificate in the IdM Web UI**

The **sc_user** entry now contains an external certificate.

### 53.4. ADDING A CERTIFICATE TO A USER ENTRY IN THE IDM CLI

This procedure describes how to add an external certificate to a user entry in IdM CLI.

Instead of uploading the whole certificate, it is also possible to upload certificate mapping data to a user entry in IdM. User entries containing either full certificates or certificate mapping data can be used in conjunction with corresponding certificate mapping rules to facilitate the configuration of smart card...
authentication for system administrators. For details, see Certificate mapping rules for configuring authentication on smart cards.

**NOTE**

If the user’s certificate has been issued by the IdM Certificate Authority, the certificate is already stored in the user entry, and you can skip this section.

### Prerequisites

- You have the certificate that you want to add to the user entry at your disposal.

### Procedure

1. Log into the IdM CLI as an administrator if you want to add a certificate to another user:

   ```bash
   [user@client SmartCard]$ kinit admin
   ```

   For adding a certificate to your own profile, you do not need the administrator’s credentials:

   ```bash
   [user@client SmartCard]$ kinit sc_user
   ```

2. Create an environment variable containing the certificate with the header and footer removed and concatenated into a single line, which is the format expected by the `ipa user-add-cert` command:

   ```bash
   [user@client SmartCard]$ export CERT=`openssl x509 -outform der -in testuser.crt | base64 -w0`
   ```

   Note that certificate in the `testuser.crt` file must be in the **PEM** format.

3. Add the certificate to the profile of sc_user using the `ipa user-add-cert` command:

   ```bash
   [user@client SmartCard]$ ipa user-add-cert sc_user --certificate=$CERT
   ```

   The **sc_user** entry now contains an external certificate.

### 53.5. Installing Tools for Managing and Using Smart Cards

To configure your smart card, you need tools which can generate certificates and store them on a smart card.

You must:

- Install the **gnutls-utils** package which helps you to manage certificates.
- Install the **opensc** package which provides a set of libraries and utilities to work with smart cards.
- Start the **pcscd** service which communicates with the smart card reader.

### Procedure
1. Install the `opensc` and `gnutls-utils` packages:
   
   ```bash
   # dnf -y install opensc gnutls-utils
   ```

2. Start the `pcscd` service.
   
   ```bash
   # systemctl start pcscd
   ```

Verify that the `pcscd` service is up and running.

### 53.6. STORING A CERTIFICATE ON A SMART CARD

This section describes smart card configuration with the `pkcs15-init` tool, which helps you to configure:

- Erasing your smart card
- Setting new PINs and optional PIN Unblocking Keys (PUKs)
- Creating a new slot on the smart card
- Storing the certificate, private key, and public key in the slot
- Locking the smart card settings (some smart cards require this type of finalization)

#### Prerequisites

- The `opensc` package, which includes the `pkcs15-init` tool is installed. For details, see [Installing tools for managing and using smart cards](#).
- The card is inserted in the reader and connected to the computer.
- You have the private key, public key, and certificate to store on the smart card. In this procedure, `testuser.key`, `testuserpublic.key`, and `testuser.crt` are the names used for the private key, public key, and the certificate.
- Your current smart card user PIN and Security Officer PIN (SO-PIN)

#### Procedure

1. Erase your smart card and authenticate yourself with your PIN:

   ```bash
   $ pkcs15-init --erase-card --use-default-transport-keys
   Using reader with a card: Reader name
   PIN [Security Officer PIN] required.
   Please enter PIN [Security Officer PIN]:
   
   The card has been erased.
   ```

2. Initialize your smart card, set your user PIN and PUK, and your Security Officer PIN and PUK:

   ```bash
   $ pkcs15-init --create-pkcs15 --use-default-transport-keys \ 
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
   Using reader with a card: Reader name
   ```
The `pkcs15-init` tool creates a new slot on the smart card.

3. Set the label and the authentication ID for the slot:

```bash
$ pkcs15-init --store-pin --label testuser \
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
Using reader with a card: Reader name
```

The label is set to a human-readable value, in this case, `testuser`. The `auth-id` must be two hexadecimal values, in this case it is set to `01`.

4. Store and label the private key in the new slot on the smart card:

```bash
$ pkcs15-init --store-private-key testuser.key --label testuser_key \
   --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

**NOTE**

The value you specify for `--id` must be the same when storing your private key, and certificate. If you do not specify a value for `--id`, a more complicated value is calculated by the tool and it is therefore easier to define your own value.

5. Store and label the certificate in the new slot on the smart card:

```bash
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \
   --auth-id 01 --id 01 --format pem --pin 963214
Using reader with a card: Reader name
```

6. (Optional) Store and label the public key in the new slot on the smart card:

```bash
$ pkcs15-init --store-public-key testuserpublic.key \
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
Using reader with a card: Reader name
```

**NOTE**

If the public key corresponds to a private key and/or certificate, you should specify the same ID as that private key and/or certificate.

7. (Optional) Some smart cards require you to finalize the card by locking the settings:

```bash
$ pkcs15-init -F
```

At this stage, your smart card includes the certificate, private key, and public key in the newly created slot. You have also created your user PIN and PUK and the Security Officer PIN and PUK.

### 53.7. LOGGING IN TO IDM WITH SMART CARDS

This section provides information about using smart cards for logging in to IdM Web UI.
Prerequisites

- The web browser is configured for using smart card authentication.
- The IdM server has been configured for smart card authentication.
- The certificate installed on your smart card is known to the IdM server.
- You need the PIN to unlock the smart card.
- The smart card has been plugged to the reader.

Procedure

1. Open the IdM Web UI in the browser.

2. Click on Log In Using Certificate

   ![Login Using Certificate](image)

3. If the Password Required dialog box opens, add the PIN to unlock the smart card and click the OK button.
   The User Identification Request dialog box opens.

   If the smart card contains more than one certificate, select the certificate you want to use for authentication in the drop down list below Choose a certificate to present as identification

4. Click the OK button.

Now you are successfully logged in to the IdM Web UI.
53.8. CONFIGURING GDM ACCESS USING SMART CARD AUTHENTICATION

The Gnome Desktop Manager (GDM) requires authentication. You can use your password, however, you can also use a smart card for authentication.

This section describes smart card authentication to access GDM.

The advantage of using smart card authentication is that if the user account is part of the Identity Management domain, you also get a ticket-granting ticket (TGT).

Prerequisites

- The smart card contains your certificate and private key.
- The user account is a member of the IdM domain.
- The certificate on the smart card maps to the user entry through:
  - Assigning the certificate to a particular user entry. For details, see, Adding a certificate to a user entry in the IdM Web UI or Adding a certificate to a user entry in the IdM CLI.
  - The certificate mapping data being applied to the account. For details, see Certificate mapping rules for configuring authentication on smart cards.

Procedure

1. Insert the smart card in the reader.
2. Enter the smart card PIN.
3. Click Sign In.

You are successfully logged in to the RHEL system and you have a TGT provided by the IdM server.

Verification steps

- In the Terminal window, enter klist and check the result:
  
  $ klist
  Ticket cache: KEYRING: persistent: 1358900015: krb_cache_TObtNMd
  Default principal: example.user@REDHAT.COM
  Valid starting Expires Service principal
  04/20/2020 13:58:24 04/20/2020 23:58:24 krbtgt/EXAMPLE.COM@EXAMPLE.COM
  renew until 04/27/2020 08:58:15

53.9. CONFIGURING SU ACCESS USING SMART CARD AUTHENTICATION

Changing to a different user requires authentication. You can use a password or a certificate. This section describes using your smart card with the su command. It means that after entering the su command, you are prompted for the smart card PIN.

Prerequisites
**Prerequisites**

- The smart card contains your certificate and private key.
- The card is inserted in the reader and connected to the computer.

**Procedure**

- In a terminal window, change to a different user with the **su** command:

```bash
$ su - example.user
PIN for smart_card
```

If the configuration is successful, you are prompted to enter the smart card PIN.
CHAPTER 54. CONFIGURING CERTIFICATES ISSUED BY ADCS FOR SMART CARD AUTHENTICATION IN IDM

This scenario describes the following situation:

- Your deployment is based on cross-forest trust between Identity Management (IdM) and Active Directory (AD).
- You want to allow smart card authentication for users whose accounts are stored in AD.
- Certificates are created and stored in Active Directory Certificate Services (ADCS).

Configuration will be accomplished in the following steps:

- Copying CA and user certificates from Active Directory to the IdM server and client
- Configuring the IdM server and clients for smart card authentication using ADCS certificates
- Converting a PFX (PKCS#12) file to be able to store the certificate and private key into the smart card
- Configuring timeouts in the sssd.conf file
- Creating certificate mapping rules for smart card authentication

Prerequisites

- Identity Management (IdM) and Active Directory (AD) trust is installed
  For details, see Installing trust between IdM and AD.
- Active Directory Certificate Services (ADCS) is installed and certificates for users are generated

54.1. SMART CARD AUTHENTICATION

A smart card is a physical device which can provide personal authentication using certificates stored on the card. Personal authentication means that you can use smart cards in the same way as user passwords.

You can store user credentials on the smart card in the form of a private key and a certificate, and special software and hardware is used to access them. You place the smart card into a reader or a USB socket and supply the PIN code for the smart card instead of providing your password.

You can configure how you want smart card authentication to work in a particular IdM client:

- Users can authenticate with the user name and password or with their smart cards
- Users can authenticate with their smart cards, and passwords are not allowed
- Users can use the smart card for logout with a function lock on removal, and passwords are not allowed

Identity Management (IdM) supports smart card authentication with:

- User certificates issued by the IdM certificate authority. For details, see Configuring Identity Management for smart card authentication.
• User certificates issued by the ADCS certificate authority. For details, see Configuring certificates issued by ADCS for smart card authentication in IdM.

• User certificates issued by local certification authority generated on a RHEL system. For details, see Configuring and importing local certificates to a smart card.

• User certificates issued by an external certificate authority.

**NOTE**

If you want to start to use smart card authentication, see the hardware requirements: Smart Card support in RHEL9.

### 54.2. WINDOWS SERVER SETTINGS REQUIRED FOR TRUST CONFIGURATION AND CERTIFICATE USAGE

This section summarizes what must be configured on Windows Server:

• Active Directory Certificate Services (ADCS) is installed

• Certificate Authority is created

• [Optional] If you are using Certificate Authority Web Enrollment, the Internet Information Services (IIS) must be configured

Export the certificate:

• Key must have **2048** bits or more

• Include a private key

• You will need a certificate in the following format: Personal Information Exchange – PKCS #12 (.PFX)
  
  ○ Enable certificate privacy

### 54.3. COPYING CERTIFICATES FROM ACTIVE DIRECTORY USING SFTP

To be able to use smart card authetication, you need to copy the following certificate files:

• A root CA certificate in the **CER** format: `adcs-winserver-ca.cer` on your IdM server.

• A user certificate with a private key in the **PFX** format: `aduser1.pfx` on an IdM client.

**NOTE**

This procedure expects SSH access is allowed. If SSH is unavailable the user must copy the file from the AD Server to the IdM server and client.

**Procedure**

1. Connect from the IdM server and copy the `adcs-winserver-ca.cer` root certificate to the IdM server:
2. Connect from the IdM client and copy the aduser1.pfx user certificate to the client:

```
[root@client1 ~]# sftp Administrator@winserver.ad.example.com
Administrator@winserver.ad.example.com's password:
Connected to Administrator@winserver.ad.example.com.
sftp> cd /<Path to certificates>
sftp> ls
adcs-winserver-ca.cer  aduser1.pfx
sftp>
sftp> get adcs-winserver-ca.cer
Fetching /<Path to certificates>/adcs-winserver-ca.cer to adcs-winserver-ca.cer
/<Path to certificates>/adcs-winserver-ca.cer                 100%  1254    15KB/s 00:00
sftp quit
```

Now the CA certificate is stored in the IdM server and the user certificates is stored on the client machine.

54.4. CONFIGURING THE IDM SERVER AND CLIENTS FOR SMART CARD AUTHENTICATION USING ADCS CERTIFICATES

You must configure the IdM (Identity Management) server and clients to be able to use smart card authentication in the IdM environment. IdM includes the ipa-advises scripts which makes all necessary changes:

- install necessary packages
- it configures IdM server and clients
- copy the CA certificates into expected locations

You can run ipa-advises on your IdM server.

This procedure describes:

- On an IdM server: Preparing the ipa-advises script to configure your IdM server for smart card authentication.
- On an IdM server: Preparing the ipa-advises script to configure your IdM client for smart card authentication.
- On an IdM server: Applying the the ipa-advises server script on the IdM server using the AD certificate.
- Moving the client script to the IdM client machine.
- On an IdM client: Applying the the `ipa-advise` client script on the IdM client using the AD certificate.

**Prerequisites**

- The certificate has been copied to the IdM server.
- Obtain the Kerberos ticket.
- Log in as a user with administration rights.

**Procedure**

1. On the IdM server, use the `ipa-advise` script for configuring a client:
   ```
   [root@idmserver ~]# ipa-advise config-client-for-smart-card-auth > sc_client.sh
   ```

2. On the IdM server, use the `ipa-advise` script for configuring a server:
   ```
   [root@idmserver ~]# ipa-advise config-server-for-smart-card-auth > sc_server.sh
   ```

3. On the IdM server, execute the script:
   ```
   [root@idmserver ~]# sh -x sc_server.sh adcs-winserver-ca.cer
   ```
   - It configures the IdM Apache HTTP Server.
   - It enables Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) on the Key Distribution Center (KDC).
   - It configures the IdM Web UI to accept smart card authorization requests.

4. Copy the `sc_client.sh` script to the client system:
   ```
   [root@idmserver ~]# scp sc_client.sh root@client1.idm.example.com:/root
   ```

5. Copy the Windows certificate to the client system:
   ```
   [root@idmserver ~]# scp adcs-winserver-ca.cer root@client1.idm.example.com:/root
   ```

6. On the client system, run the client script:
   ```
   [root@idmclient1 ~]# sh -x sc_client.sh adcs-winserver-ca.cer
   ```

The CA certificate is installed in the correct format on the IdM server and client systems and next step is to copy the user certificates onto the smart card itself.

**54.5. CONVERTING THE PFX FILE**
Before you store the PFX (PKCS#12) file into the smart card, you must:

- convert the file to the PEM format
- extract the private key and the certificate to two different files

**Prerequisites**

- The PFX file is copied into the IdM client machine.

**Procedure**

1. On the IdM client, into the PEM format:

   ```bash
   [root@idmclient1 ~]# openssl pkcs12 -in aduser1.pfx -out aduser1_cert_only.pem -clcerts -nodes
   Enter Import Password:
   ```

2. Extract the key into the separate file:

   ```bash
   [root@idmclient1 ~]# openssl pkcs12 -in adduser1.pfx -nocerts -out adduser1.pem >
   aduser1.key
   ```

3. Extract the public certificate into the separate file:

   ```bash
   [root@idmclient1 ~]# openssl pkcs12 -in adduser1.pfx -clcerts -nokeys -out
   aduser1_cert_only.pem > aduser1.crt
   ```

At this point, you can store the `aduser1.key` and `aduser1.crt` into the smart card.

## 54.6. INSTALLING TOOLS FOR MANAGING AND USING SMART CARDS

To configure your smart card, you need tools which can generate certificates and store them on a smart card.

You must:

- Install the [gnutls-utils](https://www.gnutls.org/) package which helps you to manage certificates.
- Install the [opensc](https://www.opensc.org/) package which provides a set of libraries and utilities to work with smart cards.
- Start the [pcscd](https://www.pcscd.org/) service which communicates with the smart card reader.

**Procedure**

1. Install the opensc and gnutls-utils packages:

   ```bash
   # dnf -y install opensc gnutls-utils
   ```

2. Start the pcscd service:

   ```bash
   # systemctl start pcscd
   ```
Verify that the `pcscd` service is up and running.

### 54.7. STORING A CERTIFICATE ON A SMART CARD

This section describes smart card configuration with the `pkcs15-init` tool, which helps you to configure:

- Erasing your smart card
- Setting new PINs and optional PIN Unblocking Keys (PUKs)
- Creating a new slot on the smart card
- Storing the certificate, private key, and public key in the slot
- Locking the smart card settings (some smart cards require this type of finalization)

#### Prerequisites

- The `opensc` package, which includes the `pkcs15-init` tool is installed. For details, see [Installing tools for managing and using smart cards](#).
- The card is inserted in the reader and connected to the computer.
- You have the private key, public key, and certificate to store on the smart card. In this procedure, `testuser.key`, `testuserpublic.key`, and `testuser.crt` are the names used for the private key, public key, and the certificate.
- Your current smart card user PIN and Security Officer PIN (SO-PIN)

#### Procedure

1. Erase your smart card and authenticate yourself with your PIN:

   ```
   $ pkcs15-init --erase-card --use-default-transport-keys
   Using reader with a card: Reader name
   PIN [Security Officer PIN] required.
   Please enter PIN [Security Officer PIN]:
   
   The card has been erased.
   ```

2. Initialize your smart card, set your user PIN and PUK, and your Security Officer PIN and PUK:

   ```
   $ pkcs15-init --create-pkcs15 --use-default-transport-keys \
   --pin 963214 --puk 321478 --so-pin 65498714 --so-puk 784123
   Using reader with a card: Reader name
   
   The `pcsk15-init` tool creates a new slot on the smart card.
   ```

3. Set the label and the authentication ID for the slot:

   ```
   $ pkcs15-init --store-pin --label testuser \ 
   --auth-id 01 --so-pin 65498714 --pin 963214 --puk 321478
   Using reader with a card: Reader name
   ```
The label is set to a human-readable value, in this case, `testuser`. The `auth-id` must be two hexadecimal values, in this case it is set to `01`.

4. Store and label the private key in the new slot on the smart card:

```
$ pkcs15-init --store-private-key testuser.key --label testuser_key \ 
   --auth-id 01 --id 01 --pin 963214
```

Using reader with a card: `Reader name`

**NOTE**

The value you specify for `--id` must be the same when storing your private key, and certificate. If you do not specify a value for `--id`, a more complicated value is calculated by the tool and it is therefore easier to define your own value.

5. Store and label the certificate in the new slot on the smart card:

```
$ pkcs15-init --store-certificate testuser.crt --label testuser_crt \ 
   --auth-id 01 --id 01 --format pem --pin 963214
```

Using reader with a card: `Reader name`

6. (Optional) Store and label the public key in the new slot on the smart card:

```
$ pkcs15-init --store-public-key testuserpublic.key \ 
   --label testuserpublic_key --auth-id 01 --id 01 --pin 963214
```

Using reader with a card: `Reader name`

**NOTE**

If the public key corresponds to a private key and/or certificate, you should specify the same ID as that private key and/or certificate.

7. (Optional) Some smart cards require you to finalize the card by locking the settings:

```
$ pkcs15-init -F
```

At this stage, your smart card includes the certificate, private key, and public key in the newly created slot. You have also created your user PIN and PUK and the Security Officer PIN and PUK.

### 54.8. Configuring Timeouts in sssd.conf

Authentication with a smart card certificate might take longer than the default timeouts used by SSSD. Time out expiration can be caused by:

- slow reader
- a forwarding form a physical device into a virtual environment
- too many certificates stored on the smart card
- slow response from the OCSP (Online Certificate Status Protocol) responder if OCSP is used to verify the certificates
In this case you can prolong the following timeouts in the `sssd.conf` file, for example, to 60 seconds:

- `p11_child_timeout`
- `krb5_auth_timeout`

**Prerequisites**

- You must be logged in as root.

**Procedure**

1. Open the `sssd.conf` file:
   ```
   [root@idmclient1 ~]# vim /etc/sssd/sssd.conf
   ```
2. Change the value of `p11_child_timeout`:
   ```
   [pam]
   p11_child_timeout = 60
   ```
3. Change the value of `krb5_auth_timeout`:
   ```
   [domain/IDM.EXAMPLE.COM]
   krb5_auth_timeout = 60
   ```
4. Save the settings.

Now, the interaction with the smart card is allowed to run for 1 minute (60 seconds) before authentication will fail with a timeout.

**54.9. CREATING CERTIFICATE MAPPING RULES FOR SMART CARD AUTHENTICATION**

If you want to use one certificate for a user who has accounts in AD (Active Directory) and in IdM (Identity Management), you can create a certificate mapping rule on the IdM server.

After creating such a rule, the user is able to authenticate with their smart card in both domains.

For details about certificate mapping rules, see Certificate mapping rules for configuring authentication on smart cards.
CHAPTER 55. CONFIGURING CERTIFICATE MAPPING RULES IN IDENTITY MANAGEMENT

55.1. CERTIFICATE MAPPING RULES FOR CONFIGURING AUTHENTICATION ON SMART CARDS

Certificate mapping rules are a convenient way of allowing users to authenticate using certificates in scenarios when the Identity Management (IdM) administrator does not have access to certain users’ certificates. This lack of access is typically caused by the fact that the certificates have been issued by an external certificate authority. A special use case is represented by certificates issued by the Certificate System of an Active Directory (AD) with which the IdM domain is in a trust relationship.

Certificate mapping rules are also convenient if the IdM environment is large with a lot of users using smart cards. In this situation, adding full certificates can be complicated. The subject and issuer are predictable in most scenarios and thus easier to add ahead of time than the full certificate. As a system administrator, you can create a certificate mapping rule and add certificate mapping data to a user entry even before a certificate is issued to a particular user. Once the certificate is issued, the user can log in using the certificate even though the full certificate has not yet been uploaded to the user entry.

In addition, as certificates have to be renewed at regular intervals, certificate mapping rules reduce administrative overhead. When a user’s certificate gets renewed, the administrator does not have to update the user entry. For example, if the mapping is based on the Subject and Issuer values, and if the new certificate has the same subject and issuer as the old one, the mapping still applies. If, in contrast, the full certificate was used, then the administrator would have to upload the new certificate to the user entry to replace the old one.

To set up certificate mapping:

1. An administrator has to load the certificate mapping data (typically the issuer and subject) or the full certificate into a user account.
2. An administrator has to create a certificate mapping rule to allow successful logging into IdM for a user
   a. whose account contains a certificate mapping data entry
   b. whose certificate mapping data entry matches the information on the certificate

For details on the individual components that make up a mapping rule and how to obtain and use them, see Components of an identity mapping rule in IdM and Obtaining the issuer from a certificate for use in a matching rule.

Afterwards, when the end-user presents the certificate, stored either in the filesystem or on a smart card, authentication is successful.

55.1.1. Certificate mapping rules for trusts with Active Directory domains

This section outlines the different certificate mapping use cases that are possible if an IdM deployment is in a trust relationship with an Active Directory (AD) domain.

Certificate mapping rules are a convenient way to enable access to IdM resources for users who have smart card certificates that were issued by the trusted AD Certificate System. Depending on the AD configuration, the following scenarios are possible:
If the certificate is issued by AD but the user and the certificate are stored in IdM, the mapping and the whole processing of the authentication request takes place on the IdM side. For details of configuring this scenario, see Configuring certificate mapping for users stored in IdM.

If the user is stored in AD, the processing of the authentication request takes place in AD. There are three different subcases:

- The AD user entry contains the whole certificate. For details how to configure IdM in this scenario, see Configuring certificate mapping for users whose AD user entry contains the whole certificate.

- AD is configured to map user certificates to user accounts. In this case, the AD user entry does not contain the whole certificate but instead contains an attribute called altSecurityIdentities. For details how to configure IdM in this scenario, see Configuring certificate mapping if AD is configured to map user certificates to user accounts.

- The AD user entry contains neither the whole certificate nor the mapping data. In this case, the only solution is to use the `ipa idoverrideuser-add` command to add the whole certificate to the AD user's ID override in IdM. For details, see Configuring certificate mapping if AD user entry contains no certificate or mapping data.

### 55.1.2. Components of an identity mapping rule in IdM

This section describes the components of an *identity mapping rule* in IdM and how to configure them. Each component has a default value that you can override. You can define the components in either the web UI or the CLI. In the CLI, the identity mapping rule is created using the `ipa certmaprule-add` command.

#### Mapping rule

The mapping rule component associates (or maps) a certificate with one or more user accounts. The rule defines an LDAP search filter that associates a certificate with the intended user account. Certificates issued by different certificate authorities (CAs) might have different properties and might be used in different domains. Therefore, IdM does not apply mapping rules unconditionally, but only to the appropriate certificates. The appropriate certificates are defined using matching rules.

Note that if you leave the mapping rule option empty, the certificates are searched in the `userCertificate` attribute as a DER encoded binary file.

Define the mapping rule in the CLI using the `--maprule` option.

#### Matching rule

The matching rule component selects a certificate to which you want to apply the mapping rule. The default matching rule matches certificates with the `digitalSignature` key usage and `clientAuth` extended key usage.

Define the matching rule in the CLI using the `--matchrule` option.

#### Domain list

The domain list specifies the identity domains in which you want IdM to search the users when processing identity mapping rules. If you leave the option unspecified, IdM searches the users only in the local domain to which the IdM client belongs.

Define the domain in the CLI using the `--domain` option.

#### Priority

When multiple rules are applicable to a certificate, the rule with the highest priority takes precedence.
When multiple rules are applicable to a certificate, the rule with the highest priority takes precedence. All other rules are ignored.

- The lower the numerical value, the higher the priority of the identity mapping rule. For example, a rule with a priority 1 has higher priority than a rule with a priority 2.
- If a rule has no priority value defined, it has the lowest priority.

Define the mapping rule priority in the CLI using the \texttt{--priority} option.

**Certificate mapping rule example**

To define, using the CLI, a certificate mapping rule called \texttt{simple_rule} that allows authentication for a certificate issued by the Smart Card CA of the EXAMPLE.ORG organisation as long as the Subject on that certificate matches a \texttt{certmapdata} entry in a user account in IdM:

```
# ipa certmaprule-add simple_rule --matchrule '<ISSUER>CN=Smart Card CA,O=EXAMPLE.ORG' --maprule '(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})'
```

55.1.3. Obtaining the issuer from a certificate for use in a matching rule

This procedure describes how to obtain the issuer information from a certificate so that you can copy and paste it into the matching rule of a certificate mapping rule. To get the issuer format required by a matching rule, use the \texttt{openssl x509} utility.

**Prerequisites**

- You have the user certificate in a .pem or .crt format

**Procedure**

1. Obtain the user information from the certificate. Use the \texttt{openssl x509} certificate display and signing utility with:
   - the \texttt{-noout} option to prevent the output of an encoded version of the request
   - the \texttt{-issuer} option to output the issuer name
   - the \texttt{-in} option to specify the input file name to read the certificate from
   - the \texttt{-nameopt} option with the RFC2253 value to display the output with the most specific relative distinguished name (RDN) first
     If the input file contains an Identity Management certificate, the output of the command shows that the Issuer is defined using the Organisation information:

     ```
     # openssl x509 -noout -issuer -in idm_user.crt -nameopt RFC2253
     issuer=CN=Certificate Authority,O=REALM.EXAMPLE.COM
     ```
     
     If the input file contains an Active Directory certificate, the output of the command shows that the Issuer is defined using the Domain Component information:

     ```
     # openssl x509 -noout -issuer -in ad_user.crt -nameopt RFC2253
     issuer=CN=AD-WIN2012R2-CA,DC=AD,DC=EXAMPLE,DC=COM
     ```
2. Optionally, to create a new mapping rule in the CLI based on a matching rule which specifies that the certificate issuer must be the extracted `AD-WIN2012R2-CA` of the `ad.example.com` domain and the subject on the certificate must match the `certmapdata` entry in a user account in IdM:

```
# ipa certmaprule-add simple_rule --matchrule '<ISSUER>CN=AD-WIN2012R2-CA,DC=AD,DC=EXAMPLE,DC=COM' --maprule '(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})'
```

55.1.4. Additional resources

- See the `sss-certmap(5)` man page.

## 55.2. CONFIGURING CERTIFICATE MAPPING FOR USERS STORED IN IDM

This user story describes the steps a system administrator must take to enable certificate mapping in IdM if the user for whom certificate authentication is being configured is stored in IdM.

**Prerequisites**

- The user has an account in IdM.
- The administrator has either the whole certificate or the certificate mapping data to add to the user entry.

### 55.2.1. Adding a certificate mapping rule in IdM

This section describes how to set up a certificate mapping rule so that IdM users with certificates that match the conditions specified in the mapping rule and in their certificate mapping data entries can authenticate to IdM.

### 55.2.1.1. Adding a certificate mapping rule in the IdM web UI

1. Log in to the IdM web UI as an administrator.
3. Click **Add**.

   ![Adding a new certificate mapping rule in the IdM web UI](redhat-enterprise-linux-9.0-beta-configuring-and-managing-identity-management.png)

4. Enter the rule name.
5. Enter the mapping rule. For example, to make IdM search for the **Issuer** and **Subject** entries in any certificate presented to them, and base its decision to authenticate or not on the information found in these two entries of the presented certificate:

```
(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})
```

6. Enter the matching rule. For example, to only allow certificates issued by the **Smart Card CA** of the **EXAMPLE.ORG** organization to authenticate users to IdM:

```
<ISSUER>CN=Smart Card CA,O=EXAMPLE.ORG
```

Figure 55.2. Entering the details for a certificate mapping rule in the IdM web UI

7. Click **Add** at the bottom of the dialog box to add the rule and close the box.

8. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD:

```
# systemctl restart sssd
```

Now you have a certificate mapping rule set up that compares the type of data specified in the mapping rule that it finds on a smart card certificate with the certificate mapping data in your IdM user entries. Once it finds a match, it authenticates the matching user.

### 55.2.1.2. Adding a certificate mapping rule in the IdM CLI

1. Obtain the administrator’s credentials:

   ```
   # kinit admin
   ```

2. Enter the mapping rule and the matching rule the mapping rule is based on. For example, to make IdM search for the **Issuer** and **Subject** entries in any certificate presented, and base its decision to authenticate or not on the information found in these two entries of the presented certificate, recognizing only certificates issued by the **Smart Card CA** of the **EXAMPLE.ORG** organization:

   ```
   # ipa certmaprule-add rule_name --matchrule '<ISSUER>CN=Smart Card CA,O=EXAMPLE.ORG' --maprule '(ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})
   ```
Added Certificate Identity Mapping Rule "rule_name"

Rule name: rule_name
Mapping rule: (ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500})
Matching rule: <ISSUER>CN=Smart Card CA,O=EXAMPLE.ORG
Enabled: TRUE

3. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD:

```
# systemctl restart sssd
```

Now you have a certificate mapping rule set up that compares the type of data specified in the mapping rule that it finds on a smart card certificate with the certificate mapping data in your IdM user entries. Once it finds a match, it authenticates the matching user.

### 55.2.2. Adding certificate mapping data to a user entry in IdM

This section describes how to enter certificate mapping data to an IdM user entry so that the user can authenticate using multiple certificates as long as they all contain the values specified in the certificate mapping data entry.

#### 55.2.2.1. Adding certificate mapping data to a user entry in the IdM web UI

1. Log into the IdM web UI as an administrator.
2. Navigate to Users → Active users → idm_user.
3. Find the Certificate mapping data option and click Add.
4. If you have the certificate of idm_user at your disposal:
   a. In the Command-Line Interface, display the certificate using the `cat` utility or a text editor:

   ```
   [root@server ~]# cat idm_user_certificate.pem
   -----BEGIN CERTIFICATE-----
   MIIF...[output truncated...]
   -----END CERTIFICATE-----
   ```

   b. Copy the certificate.

   c. In the IdM web UI, click Add next to Certificate and paste the certificate into the window that opens up.
Alternatively, if you do not have the certificate of `idm_user` at your disposal but know the Issuer and the Subject of the certificate, check the radio button of Issuer and subject and enter the values in the two respective boxes.

5. Click Add.

6. Optionally, if you have access to the whole certificate in the .pem format, verify that the user and certificate are linked:
   a. Use the `sss_cache` utility to invalidate the record of `idm_user` in the SSSD cache and force a reload of the `idm_user` information:

   ```bash
   # sss_cache -u idm_user
   ```
b. Run the `ipa certmap-match` command with the name of the file containing the certificate of the IdM user:

```
# ipa certmap-match idm_user_cert.pem
-------------
1 user matched
-------------
Domain: IDM.EXAMPLE.COM
User logins: idm_user
-------------
Number of entries returned 1
-------------
```

The output confirms that now you have certificate mapping data added to `idm_user` and that a corresponding mapping rule exists. This means that you can use any certificate that matches the defined certificate mapping data to authenticate as `idm_user`.

55.2.2.2. Adding certificate mapping data to a user entry in the IdM CLI

1. Obtain the administrator's credentials:

   ```
   # kinit admin
   ```

2. If you have the certificate of `idm_user` at your disposal, add the certificate to the user account using the `ipa user-add-cert` command:

   ```
   # CERT=`cat idm_user_cert.pem | tail -n +2| head -n -1 | tr -d '\n'`
   # ipa user-add-certmapdata idm_user --certificate $CERT
   ```

   Alternatively, if you do not have the certificate of `idm_user` at your disposal but know the Issuer and the Subject of `idm_user`'s certificate:

   ```
   # ipa user-add-certmapdata idm_user --subject "O=EXAMPLE.ORG,CN=test" --issuer "CN=Smart Card CA,O=EXAMPLE.ORG"
   -------------
   Added certificate mappings to user "idm_user"
   -------------
   User login: idm_user
   Certificate mapping data: X509:<I>O=EXAMPLE.ORG,CN=Smart Card CA<S>CN=test,O=EXAMPLE.ORG
   ```

3. Optionally, if you have access to the whole certificate in the `.pem` format, verify that the user and certificate are linked:

   a. Use the `sss_cache` utility to invalidate the record of `idm_user` in the SSSD cache and force a reload of the `idm_user` information:

      ```
      # sss_cache -u idm_user
      ```

   b. Run the `ipa certmap-match` command with the name of the file containing the certificate of the IdM user:

      ```
      # ipa certmap-match idm_user_cert.pem
      ```
1 user matched
--------------------
Domain: IDM.EXAMPLE.COM
User logins: idm_user
--------------------
Number of entries returned 1
--------------------

The output confirms that now you have certificate mapping data added to $idm_user$ and that a corresponding mapping rule exists. This means that you can use any certificate that matches the defined certificate mapping data to authenticate as $idm_user$.

55.3. CONFIGURING CERTIFICATE MAPPING FOR USERS WHOSE AD USER ENTRY CONTAINS THE WHOLE CERTIFICATE

This user story describes the steps necessary for enabling certificate mapping in IdM if the IdM deployment is in trust with Active Directory (AD), the user is stored in AD and the user entry in AD contains the whole certificate.

Prerequisites

- The user does not have an account in IdM.
- The user has an account in AD which contains a certificate.
- The IdM administrator has access to data on which the IdM certificate mapping rule can be based.

55.3.1. Adding a certificate mapping rule for users whose AD entry contains whole certificates

55.3.1.1. Adding a certificate mapping rule in the IdM web UI

1. Log into the IdM web UI as an administrator.


3. Click Add.

Figure 55.5. Adding a new certificate mapping rule in the IdM web UI

4. Enter the rule name.

5. Enter the mapping rule. To have the whole certificate that is presented to IdM for authentication compared to what is available in AD:
6. Enter the matching rule. For example, to only allow certificates issued by the **AD-ROOT-CA** of the **AD.EXAMPLE.COM** domain to authenticate:

```
<ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com
```

Figure 55.6. Certificate mapping rule for a user with a certificate stored in AD

7. Click **Add**.

8. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD in the CLI:

```
# systemctl restart sssd
```

55.3.1.2. Adding a certificate mapping rule in the IdM CLI

1. Obtain the administrator’s credentials:

```
# kinit admin
```

2. Enter the mapping rule and the matching rule the mapping rule is based on. To have the whole certificate that is presented for authentication compared to what is available in AD, only allowing certificates issued by the **AD-ROOT-CA** of the **AD.EXAMPLE.COM** domain to authenticate:

```
# ipa certmaprule-add simpleADrule --matchrule '<<ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com' --maprule '(userCertificate;binary={cert!bin})' --domain ad.example.com
```

```
Added Certificate Identity Mapping Rule "simpleADrule"
```

```
Rule name: simpleADrule
Mapping rule: (userCertificate;binary={cert!bin})
Matching rule: <ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com
Domain name: ad.example.com
Enabled: TRUE
```
3. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD:

```
# systemctl restart sssd
```

### 55.4. CONFIGURING CERTIFICATE MAPPING IF AD IS CONFIGURED TO MAP USER CERTIFICATES TO USER ACCOUNTS

This user story describes the steps necessary for enabling certificate mapping in IdM if the IdM deployment is in trust with Active Directory (AD), the user is stored in AD and the user entry in AD contains certificate mapping data.

**Prerequisites**

- The user does not have an account in IdM.
- The user has an account in AD which contains the `altSecurityIdentities` attribute, the AD equivalent of the IdM `certmapdata` attribute.
- The IdM administrator has access to data on which the IdM certificate mapping rule can be based.

#### 55.4.1. Adding a certificate mapping rule if the trusted AD domain is configured to map user certificates

##### 55.4.1.1. Adding a certificate mapping rule in the IdM web UI

1. Log into the IdM web UI as an administrator.
3. Click **Add**.

![Figure 55.7. Adding a new certificate mapping rule in the IdM web UI](image)

4. Enter the rule name.
5. Enter the mapping rule. For example, to make AD DC search for the `Issuer` and `Subject` entries in any certificate presented, and base its decision to authenticate or not on the information found in these two entries of the presented certificate:

   ```
   (altSecurityIdentities=X509:<I>{issuer_dn!ad_x500}<S>{subject_dn!ad_x500})
   ```

6. Enter the matching rule. For example, to only allow certificates issued by the AD-ROOT-CA of the AD.EXAMPLE.COM domain to authenticate users to IdM:
7. Enter the domain:

ad.example.com

Figure 55.8. Certificate mapping rule if AD is configured for mapping

8. Click Add.

9. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD in the CLI:

```
# systemctl restart sssd
```

55.4.1.2. Adding a certificate mapping rule in the IdM CLI

1. Obtain the administrator’s credentials:

```
# kinit admin
```

2. Enter the mapping rule and the matching rule the mapping rule is based on. For example, to make AD search for the Issuer and Subject entries in any certificate presented, and only allow certificates issued by the AD-ROOT-CA of the AD.EXAMPLE.COM domain:

```
# ipa certmaprule-add ad_configured_for_mapping_rule --matchrule '<ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com' --maprule '(altSecurityIdentities=X509:<I>{issuer_dn!ad_x500}<S>{subject_dn!ad_x500})' --domain=ad.example.com
```

```
Added Certificate Identity Mapping Rule "ad_configured_for_mapping_rule"
```

```
Rule name: ad_configured_for_mapping_rule
Mapping rule: (altSecurityIdentities=X509:<I>{issuer_dn!ad_x500}<S>{subject_dn!ad_x500})
Matching rule: <ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com
Domain name: ad.example.com
Enabled: TRUE
```
3. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD:

```
# systemctl restart sssd
```

55.4.2. Checking certificate mapping data on the AD side

The `altSecurityIdentities` attribute is the Active Directory (AD) equivalent of `certmapdata` user attribute in IdM. When configuring certificate mapping in IdM in the scenario when a trusted AD domain is configured to map user certificates to user accounts, the IdM system administrator needs to check that the `altSecurityIdentities` attribute is set correctly in the user entries in AD.

To check that AD contains the right information for the user stored in AD, use the `ldapsearch` command.

- For example, enter the command below to check with the `adserver.ad.example.com` server that the following conditions apply:
  - The `altSecurityIdentities` attribute is set in the user entry of `ad_user`.
  - The matchrule stipulates that the following conditions apply:
    - The certificate that `ad_user` uses to authenticate to AD was issued by `AD-ROOT-CA` of the `ad.example.com` domain.
    - The subject is `<S>DC=com,DC=example,DC=ad,CN=Users,CN=ad_user`.

```
$ ldapsearch -o ldif-wrap=no -LLL -h adserver.ad.example.com -p 389 -D cn=Administrator,cn=users,dc=ad,dc=example,dc=com -W -b cn=users,dc=ad,dc=example,dc=com "(cn=ad_user)" altSecurityIdentities
Enter LDAP Password:
dn: CN=ad_user,CN=Users,DC=ad,DC=example,DC=com
altSecurityIdentities: X509:<I>DC=com,DC=example,DC=ad,CN=AD-ROOT-CA</I>DC=com,DC=example,DC=ad,CN=Users,CN=ad_user
```

55.5. CONFIGURING CERTIFICATE MAPPING IF AD USER ENTRY CONTAINS NO CERTIFICATE OR MAPPING DATA

This user story describes the steps necessary for enabling certificate mapping in IdM if the IdM deployment is in trust with Active Directory (AD), the user is stored in AD and the user entry in AD contains neither the whole certificate nor certificate mapping data.

**Prerequisites**

- The user does not have an account in IdM.
- The user has an account in AD which contains neither the whole certificate nor the `altSecurityIdentities` attribute, the AD equivalent of the IdM `certmapdata` attribute.
- The IdM administrator has the whole AD user certificate to add to the AD user’s user ID override in IdM.
55.5.1. Adding a certificate mapping rule if the AD user entry contains no certificate or mapping data

55.5.1.1. Adding a certificate mapping rule in the IdM web UI

1. Log into the IdM web UI as an administrator.


3. Click Add.

Figure 55.9. Adding a new certificate mapping rule in the IdM web UI

4. Enter the rule name.

5. Enter the mapping rule. To have the whole certificate that is presented to IdM for authentication compared to the certificate stored in the user ID override entry of the AD user entry in IdM:

\[(\text{userCertificate};\text{binary}={\text{cert!bin}})\]

6. Enter the matching rule. For example, to only allow certificates issued by the AD-ROOT-CA of the AD.EXAMPLE.COM domain to authenticate:

\[<\text{ISSUER}>\text{CN=AD-ROOT-CA,DC=ad,DC=example,DC=com}\]

7. Enter the domain name. For example, to search for users in the ad.example.com domain:

Figure 55.10. Certificate mapping rule for a user with no certificate or mapping data stored in AD

8. Click Add.
9. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD in the CLI:

```
# systemctl restart sssd
```

55.5.1.2. Adding a certificate mapping rule in the IdM CLI

1. Obtain the administrator’s credentials:

```
# kinit admin
```

2. Enter the mapping rule and the matching rule the mapping rule is based on. To have the whole certificate that is presented for authentication compared to the certificate stored in the user ID override entry of the AD user entry in IdM, only allowing certificates issued by the AD-ROOT-CA of the AD.EXAMPLE.COM domain to authenticate:

```
# ipa certmaprule-add simpleADrule --matchrule '<ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com' --maprule '(userCertificate;binary={cert!bin})' --domain ad.example.com
```

```
Added Certificate Identity Mapping Rule "simpleADrule"
```

```
Rule name: simpleADrule
Mapping rule: (userCertificate;binary={cert!bin})
Matching rule: <ISSUER-CN=AD-ROOT-CA,DC=ad,DC=example,DC=com
Domain name: ad.example.com
Enabled: TRUE
```

3. The System Security Services Daemon (SSSD) periodically re-reads the certificate mapping rules. To force the newly-created rule to be loaded immediately, restart SSSD:

```
# systemctl restart sssd
```

55.5.2. Adding a certificate to an AD user’s ID override if the user entry in AD contains no certificate or mapping data

55.5.2.1. Adding a certificate to an AD user’s ID override in the IdM web UI

1. Navigate to Identity → ID Views → Default Trust View.

2. Click Add.

Figure 55.11. Adding a new user ID override in the IdM web UI
3. In the **User to override** field, enter **ad_user@ad.example.com**.

4. Copy and paste the certificate of **ad_user** into the **Certificate** field.

Figure 55.12. Configuring the User ID override for an AD user

5. Click **Add**.

6. Optionally, verify that the user and certificate are linked:
   a. Use the `sss_cache` utility to invalidate the record of **ad_user@ad.example.com** in the SSSD cache and force a reload of the **ad_user@ad.example.com** information:

      ```
      # sss_cache -u ad_user@ad.example.com
      ```

   b. Run the `ipa certmap-match` command with the name of the file containing the certificate of the AD user:

      ```
      # ipa certmap-match ad_user_cert.pem
      ------------
      1 user matched
      ------------
      Domain: AD.EXAMPLE.COM
      User logins: ad_user@ad.example.com
      ------------
      Number of entries returned 1
      ------------
      ```

      The output confirms that you have certificate mapping data added to **ad_user@ad.example.com** and that a corresponding mapping rule defined in _Adding a certificate mapping rule if the AD user entry contains no certificate or mapping data_ exists.
This means that you can use any certificate that matches the defined certificate mapping data to authenticate as `ad_user@ad.example.com`.

55.5.2.2. Adding a certificate to an AD user’s ID override in the IdM CLI

1. Obtain the administrator’s credentials:

   ```
   # kinit admin
   ```

2. Add the certificate of `ad_user@ad.example.com` to the user account using the `ipa idoverrideuser-add-cert` command:

   ```
   # CERT=`cat ad_user_cert.pem | tail -n +2| head -n -1 | tr -d '
'
   # ipa idoverrideuser-add-cert ad_user@ad.example.com --certificate $CERT
   ```

3. Optionally, verify that the user and certificate are linked:

   a. Use the `sss_cache` utility to invalidate the record of `ad_user@ad.example.com` in the SSSD cache and force a reload of the `ad_user@ad.example.com` information:

      ```
      # sss_cache -u ad_user@ad.example.com
      ```

   b. Run the `ipa certmap-match` command with the name of the file containing the certificate of the AD user:

      ```
      # ipa certmap-match ad_user_cert.pem
      ------------
      1 user matched
      ------------
      Domain: AD.EXAMPLE.COM
      User logins: ad_user@ad.example.com
      ------------
      Number of entries returned 1
      ------------
      ```

      The output confirms that you have certificate mapping data added to `ad_user@ad.example.com` and that a corresponding mapping rule defined in Adding a certificate mapping rule if the AD user entry contains no certificate or mapping data exists. This means that you can use any certificate that matches the defined certificate mapping data to authenticate as `ad_user@ad.example.com`.

55.6. COMBINING SEVERAL IDENTITY MAPPING RULES INTO ONE

To combine several identity mapping rules into one combined rule, use the `|` (or) character to precede the individual mapping rules, and separate them using `()` brackets, for example:

Certificate mapping filter example 1

```bash
$ ipa certmaprule-add ad_cert_for_ipa_and_ad_users \ --maprule='(|(ipacertmapdata=X509:<I> {issuer_dn!nss_x500}<S>{subject_dn!nss_x500})(altSecurityIdentities=X509:<I> {issuer_dn!ad_x500}<S>{subject_dn!ad_x500}))' \ --matchrule='<ISSUER>CN=AD-ROOT-CA,DC=ad,DC=example,DC=com' \ --domain=ad.example.com
```
In the above example, the filter definition in the `--maprule` option includes these criteria:

- `ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500}` is a filter that links the subject and issuer from a smart card certificate to the value of the `ipacertmapdata` attribute in an IdM user account, as described in Adding a certificate mapping rule in IdM.

- `altSecurityIdentities=X509:<I>{issuer_dn!ad_x500}<S>{subject_dn!ad_x500}` is a filter that links the subject and issuer from a smart card certificate to the value of the `altSecurityIdentities` attribute in an AD user account, as described in Adding a certificate mapping rule if the trusted AD domain is configured to map user certificates.

- The addition of the `--domain=ad.example.com` option means that users mapped to a given certificate are not only searched in the local `idm.example.com` domain but also in the `ad.example.com` domain.

The filter definition in the `--maprule` option accepts the logical operator `|` (or), so that you can specify multiple criteria. In this case, the rule maps all user accounts that meet at least one of the criteria.

Certificate mapping filter example 2

```
$ ipa certmaprule-add ipa_cert_for_ad_users \
   --maprule='(|(userCertificate;binary={cert!bin})(ipacertmapdata=X509:<I>\n   {issuer_dn!nss_x500}<S>{subject_dn!nss_x500})(altSecurityIdentities=X509:<I>\n   {issuer_dn!ad_x500}<S>{subject_dn!ad_x500}))' \n   --matchrule='<ISSUER>CN=Certificate Authority,O=REALM.EXAMPLE.COM' \n   --domain=idm.example.com --domain=ad.example.com
```

In the above example, the filter definition in the `--maprule` option includes these criteria:

- `userCertificate;binary={cert!bin}` is a filter that returns user entries that include the whole certificate. For AD users, creating this type of filter is described in detail in Adding a certificate mapping rule if the AD user entry contains no certificate or mapping data.

- `ipacertmapdata=X509:<I>{issuer_dn!nss_x500}<S>{subject_dn!nss_x500}` is a filter that links the subject and issuer from a smart card certificate to the value of the `ipacertmapdata` attribute in an IdM user account, as described in Adding a certificate mapping rule in IdM.

- `altSecurityIdentities=X509:<I>{issuer_dn!ad_x500}<S>{subject_dn!ad_x500}` is a filter that links the subject and issuer from a smart card certificate to the value of the `altSecurityIdentities` attribute in an AD user account, as described in Adding a certificate mapping rule if the trusted AD domain is configured to map user certificates.

The filter definition in the `--maprule` option accepts the logical operator `|` (or), so that you can specify multiple criteria. In this case, the rule maps all user accounts that meet at least one of the criteria.
CHAPTER 56. CONFIGURING AUTHENTICATION WITH A CERTIFICATE STORED ON THE DESKTOP OF AN IDM CLIENT

By configuring Identity Management (IdM), IdM system administrators can enable users to authenticate to the IdM web UI and command-line interface (CLI) using a certificate that a Certificate Authority (CA) has issued to the users.

The web browser can run on a system that is not part of the IdM domain.

This user story provides instructions on how to effectively configure and test logging into Identity Management web UI and CLI with a certificate stored on the desktop of an IdM client. In following this user story,

- you can skip Section 56.2, “Requesting a new user certificate and exporting it to the client” if the user you want to authenticate using a certificate already has a certificate;
- you can skip Section 56.3, “Making sure the certificate and user are linked together” if the user’s certificate has been issued by the IdM CA.

NOTE

Only Identity Management users can log into the web UI using a certificate. Active Directory users can log in with their user name and password.

56.1. CONFIGURING THE.IDENTITY MANAGEMENT SERVER FOR CERTIFICATE AUTHENTICATION IN THE WEB UI

As an Identity Management (IdM) administrator, you can allow users to use certificates to authenticate to your IdM environment.

Procedure

As the Identity Management administrator:

1. On an Identity Management server, obtain administrator privileges and create a shell script to configure the server.
   a. Run the `ipa-advise config-server-for-smart-card-auth` command, and save its output to a file, for example `server_certificate_script.sh`:

   ```bash
   # kinit admin
   # ipa-advise config-server-for-smart-card-auth > server_certificate_script.sh
   ```
   b. Add execute permissions to the file using the `chmod` utility:

   ```bash
   # chmod +x server_certificate_script.sh
   ```

2. On all the servers in the Identity Management domain, run the `server_certificate_script.sh` script
   a. with the path of the IdM Certificate Authority certificate, `/etc/ipa/ca.crt`, as input if the IdM CA is the only certificate authority that has issued the certificates of the users you want to enable certificate authentication for:
b. with the paths leading to the relevant CA certificates as input if different external CAs signed the certificates of the users who you want to enable certificate authentication for:

```
# ./server_certificate_script.sh /etc/ipa/ca.crt
```

NOTE
Do not forget to run the script on each new replica that you add to the system in the future if you want to have certificate authentication for users enabled in the whole topology.

56.2. REQUESTING A NEW USER CERTIFICATE AND EXPORTING IT TO THE CLIENT

As an Identity Management (IdM) administrator, you can create certificates for users in your IdM environment and export them to the IdM clients on which you want to enable certificate authentication for users.

NOTE
You can skip this section if the user you want to authenticate using a certificate already has a certificate.

Procedure

1. Optionally, create a new directory, for example `~/certdb/`, and make it a temporary certificate database. When asked, create an NSS Certificate DB password to encrypt the keys to the certificate to be generated in a subsequent step:

```
# mkdir ~/certdb/
# certutil -N -d ~/certdb/
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.
```

```
Enter new password:
Re-enter password:
```

2. Create the certificate signing request (CSR) and redirect the output to a file. For example, to create a CSR with the name `certificate_request.csr` for a `4096` bit certificate for the `idm_user` user in the `IDM.EXAMPLE.COM` realm, setting the nickname of the certificate private keys to `idm_user` for easy findability, and setting the subject to `CN=idm_user,O=IDM.EXAMPLE.COM`:

```
# certutil -R -d ~/certdb/ -a -g 4096 -n idm_user -s "CN=idm_user,O=IDM.EXAMPLE.COM"
```

3. When prompted, enter the same password that you entered when using `certutil` to create the temporary database. Then continue typing randomly until told to stop:
Enter Password or Pin for "NSS Certificate DB":

A random seed must be generated that will be used in the creation of your key. One of the easiest ways to create a random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter is full. DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!

Continue typing until the progress meter is full:

4. Submit the certificate request file to the server. Specify the Kerberos principal to associate with the newly-issued certificate, the output file to store the certificate, and optionally the certificate profile. For example, to obtain a certificate of the IECUserRoles profile, a profile with added user roles extension, for the idm_user@IDM.EXAMPLE.COM principal, and save it in the ~/idm_user.pem file:

```bash
# ipa cert-request certificate_request.csr --principal=idm_user@IDM.EXAMPLE.COM --profile-id=IECUserRoles --certificate-out=~/.idm_user.pem
```

5. Add the certificate to the NSS database. Use the -n option to set the same nickname that you used when creating the CSR previously so that the certificate matches the private key in the NSS database. The -t option sets the trust level. For details, see the certutil(1) man page. The -i option specifies the input certificate file. For example, to add to the NSS database a certificate with the idm_user nickname that is stored in the ~/idm_user.pem file in the ~/certdb/ database:

```bash
# certutil -A -d ~/certdb/ -n idm_user -t "P,," -i ~/idm_user.pem
```

6. Verify that the key in the NSS database does not show (orphan) as its nickname. For example, to verify that the certificate stored in the ~/certdb/ database is not orphaned:

```bash
# certutil -K -d ~/certdb/
< 0> rsa 5ad14d41463b87a095b1896cf0068ccc467df395  NSS Certificate DB:idm_user
```

7. Use the pk12util command to export the certificate from the NSS database to the PKCS12 format. For example, to export the certificate with the idm_user nickname from the /root/certdb NSS database into the ~/idm_user.p12 file:

```bash
# pk12util -d ~/certdb -o ~/idm_user.p12 -n idm_user
```

8. Transfer the certificate to the host on which you want the certificate authentication for idm_user to be enabled:

```bash
# scp ~/idm_user.p12 idm_user@client.idm.example.com:/home/idm_user/
```
9. On the host to which the certificate has been transferred, make the directory in which the .pkcs12 file is stored inaccessible to the 'other' group for security reasons:

```
# chmod o-rwx /home/idm_user/
```

10. For security reasons, remove the temporary NSS database and the .pkcs12 file from the server:

```
# rm ~/certdb/
# rm ~/idm_user.p12
```

### 56.3. MAKING SURE THE CERTIFICATE AND USER ARE LINKED TOGETHER

**NOTE**

You can skip this section if the user’s certificate has been issued by the IdM CA.

For certificate authentication to work, you need to make sure that the certificate is linked to the user that will use it to authenticate to Identity Management (IdM).

- If the certificate is provided by a Certificate Authority that is not part of your Identity Management environment, link the user and the certificate following the procedure described in Linking User Accounts to Certificates.

- If the certificate is provided by Identity Management CA, the certificate is already automatically added in the user entry and you do not have to link the certificate to the user account. For details on creating a new certificate in IdM, see Section 56.2, “Requesting a new user certificate and exporting it to the client”.

### 56.4. CONFIGURING A BROWSER TO ENABLE CERTIFICATE AUTHENTICATION

To be able to authenticate with a certificate when using the WebUI to log into Identity Management (IdM), you need to import the user and the relevant certificate authority (CA) certificates into the Mozilla Firefox or Google Chrome browser. The host itself on which the browser is running does not have to be part of the IdM domain.

IdM supports the following browsers for connecting to the WebUI:

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

The following procedure shows how to configure the Mozilla Firefox 57.0.1 browser.

**Prerequisites**

- You have the user certificate that you want to import to the browser at your disposal in the PKCS#12 format.

**Procedure**
1. Open Firefox, then navigate to **Preferences → Privacy & Security**.

   **Figure 56.1. Privacy and Security section in Preferences**

2. Click **View Certificates**.

   **Figure 56.2. View Certificates in Privacy and Security**

3. In the **Your Certificates** tab, click **Import**. Locate and open the certificate of the user in the PKCS12 format, then click **OK** and **OK**.

4. Make sure that the Identity Management Certificate Authority is recognized by Firefox as a trusted authority:

   a. Save the IdM CA certificate locally:

      • Navigate to the IdM web UI by writing the name of your IdM server in the Firefox address bar. Click **Advanced** on the Insecure Connection warning page.

   **Figure 56.3. Insecure Connection**

      • **Add Exception. Click View.**
Figure 56.4. View the Details of a Certificate

- In the Details tab, highlight the Certificate Authority fields.

Figure 56.5. Exporting the CA Certificate

- Click Export. Save the CA certificate, for example as the CertificateAuthority.crt file, then click Close, and Cancel.
b. Import the IdM CA certificate to Firefox as a trusted certificate authority certificate:

- Open Firefox, navigate to Preferences and click Privacy & Security.

**Figure 56.6. Privacy and Security section in Preferences**

![Privacy and Security section in Preferences](image)

- Click View Certificates.

**Figure 56.7. View Certificates in Privacy and Security**

![View Certificates in Privacy and Security](image)

- In the Authorities tab, click Import. Locate and open the CA certificate that you saved in the previous step in the CertificateAuthority.crt file. Trust the certificate to identify websites, then click OK and OK.


56.5. AUTHENTICATING TO THE IDENTITY MANAGEMENT WEB UI WITH A CERTIFICATE AS AN IDENTITY MANAGEMENT USER

This procedure describes authenticating as a user to the Identity Management (IdM) web UI using a certificate stored on the desktop of an Identity Management client.

**Procedure**

1. In the browser, navigate to the Identity Management web UI at, for example, https://server.idm.example.com/ipa/ui.

2. Click Login Using Certificate. Login Using Certificate in the Identity Management web UI.
3. The user’s certificate should already be selected. Uncheck **Remember this decision**, then click **OK**.

You are now authenticated as the user who corresponds to the certificate.

**Additional resources**

- For information about authenticating to the IdM web UI using a certificate stored on a smart card, see [Configuring Identity Management for smart card authentication](#).

### 56.6. Configuring an IDM Client to Enable Authenticating to the CLI Using a Certificate

To make certificate authentication work for an IdM user in the Command Line Interface (CLI) of your IdM client, import the IdM user’s certificate and the private key to the IdM client. For details on creating and transferring the user certificate, see [Section 56.2, “Requesting a new user certificate and exporting it to the client”](#).

**Procedure**

- Log into the IdM client and have the .p12 file containing the user’s certificate and the private key ready. To obtain and cache the Kerberos ticket granting ticket (TGT), run the `kinit` command with the user’s principal, using the `-X` option with the `X509_username:/path/to/file.p12` attribute to specify where to find the user’s X509 identity information. For example, to obtain the TGT for `idm_user` using the user’s identity information stored in the `~/idm_user.p12` file:

  ```bash
  $ kinit -X X509_idm_user='PKCS12:/~idm_user.p12' idm_user
  ```

**NOTE**

The command also supports the .pem file format: `kinit -X X509_username='FILE:/path/to/cert.pem,/path/to/key' user_principal`
CHAPTER 57. USING IDM CA RENEWAL SERVER

57.1. EXPLANATION OF IDM CA RENEWAL SERVER

In an Identity Management (IdM) deployment that uses an embedded certificate authority (CA), the CA renewal server maintains and renews IdM system certificates. It ensures robust IdM deployments.

IdM system certificates include:

- **IdM CA** certificate
- **OCSP** signing certificate
- **IdM CA subsystem** certificates
- **IdM CA audit signing** certificate
- **IdM renewal agent** (RA) certificate
- **KRA** transport and storage certificates

What characterizes system certificates is that their keys are shared by all CA replicas. In contrast, the IdM service certificates (for example, **LDAP**, **HTTP**, and **PKINIT** certificates), have different keypairs and subject names on different IdM CA servers.

In IdM topology, by default, the first IdM CA server is the CA renewal server.

**NOTE**

In upstream documentation, the IdM CA is called **Dogtag**.

The role of the CA renewal server

The **IdM CA**, **IdM CA subsystem**, and **IdM RA** certificates are crucial for IdM deployment. Each certificate is stored in an NSS database in the `/etc/pki/pki-tomcat/` directory and also as an LDAP database entry. The certificate stored in LDAP must match the certificate stored in the NSS database. If they do not match, authentication failures occur between the IdM framework and IdM CA, and between IdM CA and LDAP.

All IdM CA replicas have tracking requests for every system certificate. If an IdM deployment with integrated CA does not contain a CA renewal server, each IdM CA server requests the renewal of system certificates independently. This results in different CA replicas having various system certificates and authentication failures occurring.

Appointing one CA replica as the renewal server allows the system certificates to be renewed exactly once, when required, and thus prevents authentication failures.

The role of the certmonger service on CA replicas

The **certmonger** service running on all IdM CA replicas uses the **dogtag-ipa-ca-renew-agent** renewal helper to keep track of IdM system certificates. The renewal helper program reads the CA renewal server configuration. On each CA replica that is not the CA renewal server, the renewal helper retrieves the latest system certificates from the `ca_renewal` LDAP entries. Due to non-determinism in when exactly certmonger renewal attempts occur, the **dogtag-ipa-ca-renew-agent** helper sometimes attempts to update a system certificate before the CA renewal server has actually renewed the certificate. If this happens, the old, soon-to-expire certificate is returned to the certmonger service on
the CA replica. The certmonger service, realizing it is the same certificate that is already stored in its database, keeps attempting to renew the certificate with some delay between individual attempts until it can retrieve the updated certificate from the CA renewal server.

The correct functioning of IdM CA renewal server
An IdM deployment with an embedded CA is an IdM deployment that was installed with an IdM CA - or whose IdM CA server was installed later. An IdM deployment with an embedded CA must at all times have exactly one CA replica configured as the renewal server. The renewal server must be online and fully functional, and must replicate properly with the other servers.

If the current CA renewal server is being deleted using the `ipa server-del`, `ipa-replica-manage del`, `ipa-csreplica-manage del` or `ipa-server-install --uninstall` commands, another CA replica is automatically assigned as the CA renewal server. This policy ensures that the renewal server configuration remains valid.

This policy does not cover the following situations:

- **Offline renewal server**
  If the renewal server is offline for an extended duration, it may miss a renewal window. In this situation, all nonrenewal CA servers keep reinstalling the current system certificates until the certificates expire. When this occurs, the IdM deployment is disrupted because even one expired certificate can cause renewal failures for other certificates.

  To prevent this situation: if your current renewal server is offline and unavailable for an extended period of time, consider assigning a new CA renewal server manually.

- **Replication problems**
  If replication problems exist between the renewal server and other CA replicas, renewal might succeed, but the other CA replicas might not be able to retrieve the updated certificates before they expire.

  To prevent this situation, make sure that your replication agreements are working correctly. For details, see general or specific replication troubleshooting guidelines in the RHEL 7 Linux Domain Identity, Authentication, and Policy Guide.

57.2. CHANGING AND RESETTNG IDM CA RENEWAL SERVER

When a certificate authority (CA) renewal server is being decommissioned, Identity Management (IdM) automatically selects a new CA renewal server from the list of IdM CA servers. The system administrator cannot influence the selection.

To be able to select the new IdM CA renewal server, the system administrator must perform the replacement manually. Choose the new CA renewal server before starting the process of decommissioning the current renewal server.

If the current CA renewal server configuration is invalid, reset the IdM CA renewal server.

Complete this procedure to change or reset the CA renewal server.

**Prerequisites**

- You have the IdM administrator credentials.

**Procedure**

1. Obtain the IdM administrator credentials:
2. Optionally, to find out which IdM servers in the deployment have the CA role necessary to be eligible to become the new CA renewal server:

```
~$ ipa server-role-find --role 'CA server'
----------------------
2 server roles matched
----------------------
Server name: server.idm.example.com
Role name: CA server
Role status: enabled

Server name: replica.idm.example.com
Role name: CA server
Role status: enabled
----------------------
Number of entries returned 2
----------------------
```

There are two CA servers in the deployment.

3. Optionally, to find out which CA server is the current CA renewal server, enter:

```
~$ ipa config-show | grep 'CA renewal'
IPA CA renewal master: server.idm.example.com
```

The current renewal server is `server.idm.example.com`.

4. To change the renewal server configuration, use the `ipa config-mod` utility with the `--ca-renewal-master-server` option:

```
~$ ipa config-mod --ca-renewal-master-server replica.idm.example.com | grep 'CA renewal'
IPA CA renewal master: replica.idm.example.com
```

**IMPORTANT**

You can also switch to a new CA renewal server using:

- the `ipa-cacert-manage --renew` command. This command both renews the CA certificate and makes the CA server on which you execute the command the new CA renewal server.

- the `ipa-cert-fix` command. This command recovers the deployment when expired certificates are causing failures. It also makes the CA server on which you execute the command the new CA renewal server. For details, see [Renewing expired system certificates when IdM is offline](#).

## 57.3. SWITCHING FROM AN EXTERNALLY TO SELF-SIGNED CA IN IDM

Complete this procedure to switch from an externally-signed to a self-signed certificate of the Identity Management (IdM) certificate authority (CA). With a self-signed CA, the renewal of the CA certificate is
managed automatically: a system administrator does not need to submit a certificate signing request (CSR) to an external authority.

Switching from an externally-signed to a self-signed CA replaces only the CA certificate. The certificates signed by the previous CA are still valid and still in use. For example, the certificate chain for the LDAP certificate remains unchanged even after you have moved to a self-signed CA:

```
external_CA certificate > IdM CA certificate > LDAP certificate
```

**Prerequisites**

- You have `root` access to the IdM CA renewal server and all IdM clients and servers.

**Procedure**

1. On the IdM CA renewal server, renew the CA certificate as self-signed:

   ```sh
   # ipa-cacert-manage renew --self-signed
   Renewing CA certificate, please wait
   CA certificate successfully renewed
   The ipa-cacert-manage command was successful
   ```

2. **SSH** to all the remaining IdM servers and clients as `root`. For example:

   ```sh
   # ssh root@idmclient01.idm.example.com
   ```

3. On the IdM client, update the local IdM certificate databases with the certificates from the server:

   ```sh
   [idmclient01 ~]# ipa-certupdate
   Systemwide CA database updated.
   Systemwide CA database updated.
   The ipa-certupdate command was successful
   ```

4. Optionally, to check if your update has been successful and the new CA certificate has been added to the `/etc/ipa/ca.crt` file:

   ```sh
   [idmclient01 ~]$ openssl crl2pkcs7 -nocrl -certfile /etc/ipa/ca.crt | openssl pkcs7 -print_certs -text -noout
   [...] Certificate: Data:
   Version: 3 (0x2) Serial Number: 39 (0x27) Signature Algorithm: sha256WithRSAEncryption Issuer: O=IDM.EXAMPLE.COM, CN=Certificate Authority Validity Not Before: Jul 1 16:32:45 2019 GMT Not After : Jul 1 16:32:45 2039 GMT Subject: O=IDM.EXAMPLE.COM, CN=Certificate Authority
   [...]```

   The output shows that the update has been successful as the new CA certificate is listed with the older CA certificates.
57.4. RENEWING THE IDM CA RENEWAL SERVER WITH AN EXTERNALLY-SIGNED CERTIFICATE

This section describes how to renew the Identity Management (IdM) certificate authority (CA) certificate using an external CA to sign the certificate signing request (CSR). In this configuration, your IdM CA server is a subCA of the external CA. The external CA can, but does not have to, be an Active Directory Certificate Server (AD CS).

If the external certificate authority is AD CS, you can specify the template you want for the IdM CA certificate in the CSR. A certificate template defines the policies and rules that a CA uses when a certificate request is received. Certificate templates in AD correspond to certificate profiles in IdM.

You can define a specific AD CS template by its Object Identifier (OID). OIDs are unique numeric values issued by various issuing authorities to uniquely identify data elements, syntaxes, and other parts of distributed applications.

Alternatively, you can define a specific AD CS template by its name. For example, the name of the default profile used in a CSR submitted by an IdM CA to an AD CS is subCA.

To define a profile by specifying its OID or name in the CSR, use the external-ca-profile option. For details, see the ipa-cacert-manage man page.

Apart from using a ready-made certificate template, you can also create a custom certificate template in the AD CS, and use it in the CSR.

Prerequisites

- You have root access to the IdM CA renewal server.

Procedure

Complete this procedure to renew the certificate of the IdM CA with external signing, regardless of whether current CA certificate is self-signed or externally-signed.

1. Create a CSR to be submitted to the external CA:
   
   - If the external CA is an AD CS, use the --external-ca-type=ms-cs option. If you want a different template than the default subCA template, specify it using the --external-ca-profile option:
     ```
     ~]# ipa-cacert-manage renew --external-ca --external-ca-type=ms-cs [--external-ca-profile=PROFILE]
     Exporting CA certificate signing request, please wait
     The next step is to get /var/lib/ipa/ca.csr signed by your CA and re-run ipa-cacert-manage as:
     ipa-cacert-manage renew --external-cert-file=/path/to/signed_certificate --external-cert-file=/path/to/external_ca_certificate
     The ipa-cacert-manage command was successful
     ```
   
   - If the external CA is not an AD CS:
     ```
     ~]# ipa-cacert-manage renew --external-ca
     Exporting CA certificate signing request, please wait
     The next step is to get /var/lib/ipa/ca.csr signed by your CA and re-run ipa-cacert-manage as:
     ```
ipa-cacert-manage renew --external-cert-file=/path/to/signed_certificate --external-cert-file=/path/to/external_ca_certificate

The ipa-cacert-manage command was successful.

The output shows that a CSR has been created and is stored in the /var/lib/ipa/ca.csr file.

2. Submit the CSR located in /var/lib/ipa/ca.csr to the external CA. The process differs depending on the service to be used as the external CA.

3. Retrieve the issued certificate and the CA certificate chain for the issuing CA in a base 64-encoded blob, which is:
   - a PEM file if the external CA is not an AD CS.
   - a Base_64 certificate if the external CA is an AD CS.
   
   The process differs for every certificate service. Usually, a download link on a web page or in the notification email allows the administrator to download all the required certificates.

   If the external CA is an AD CS and you have submitted the CSR with a known template through the Microsoft Windows Certification Authority management window, the AD CS issues the certificate immediately and the Save Certificate dialog appears in the AD CS web interface, asking where to save the issued certificate.

4. Run the ipa-cacert-manage renew command again, adding all the CA certificate files required to supply a full certificate chain. Specify as many files as you need, using the --external-cert-file option multiple times:

   ~]$ ipa-cacert-manage renew --external-cert-file=/path/to/signed_certificate --external-cert-file=/path/to/external_ca_certificate_1 --external-cert-file=/path/to/external_ca_certificate_2

5. On all the IdM servers and clients, update the local IdM certificate databases with the certificates from the server:

   [client ~]$ ipa-certupdate
   Systemwide CA database updated.
   Systemwide CA database updated.
   The ipa-certupdate command was successful.

6. Optionally, to check if your update has been successful and the new CA certificate has been added to the /etc/ipa/ca.crt file:

   [client ~]$ openssl crl2pkcs7 -nocrl -certfile /etc/ipa/ca.crt | openssl pkcs7 -print_certs -text -noout
   [...]  
   Certificate:
   Data:
   Version: 3 (0x2)
   Serial Number: 39 (0x27)
   Signature Algorithm: sha256WithRSAEncryption
   Issuer: O=IDM.EXAMPLE.COM, CN=Certificate Authority
   Validity
   Not Before: Jul 1 16:32:45 2019 GMT
   Not After : Jul 1 16:32:45 2039 GMT
   Subject: O=IDM.EXAMPLE.COM, CN=Certificate Authority
   [...]
The output shows that the update has been successful as the new CA certificate is listed with the older CA certificates.
CHAPTER 58. RENEWING EXPIRED SYSTEM CERTIFICATES WHEN IDM IS OFFLINE

When a system certificate has expired, Identity Management (IdM) fails to start. IdM supports renewing system certificates when IdM is offline using the ipa-cert-fix tool.

58.1. RENEWING EXPIRED SYSTEM CERTIFICATES ON A CA RENEWAL SERVER

This section describes how to apply the ipa-cert-fix tool on expired IdM certificates.

IMPORTANT

If you run the ipa-cert-fix tool on a CA (Certificate Authority) host that is not the CA renewal server, and the utility renews shared certificates, that host automatically becomes the new CA renewal server in the domain. There must always be only one CA renewal server in the domain to avoid inconsistencies.

Prerequisites

- Log in to the server with administration rights

Procedure

1. Start the ipa-cert-fix tool to analyze the system and list expired certificates that require renewal:

```
# ipa-cert-fix
...
The following certificates will be renewed:

Dogtag sslserver certificate:
  Subject: CN=ca1.example.com,O=EXAMPLE.COM 201905222205
  Serial: 13
  ...
  Enter "yes" to proceed:
```

2. Enter yes to start the renewal process:

```
Enter "yes" to proceed: yes
Proceeding.
Renewed Dogtag sslserver certificate:
  Subject: CN=ca1.example.com,O=EXAMPLE.COM 201905222205
  Serial: 268369925
  Expires: 2021-08-14 02:19:33
  ...
  Becoming renewal master.
The ipa-cert-fix command was successful
```

It can take up to one minute before ipa-cert-fix renews all expired certificates.
3. Optionally, verify that all services are now running:

```
# ipactl status
Directory Service: RUNNING
krb5kdc Service: RUNNING
kadmind Service: RUNNING
httpd Service: RUNNING
ipa-custodia Service: RUNNING
pki-tomcatd Service: RUNNING
ipa-otpd Service: RUNNING
ipa: INFO: The ipactl command was successful
```

At this point, certificates have been renewed and services are running. The next step is to check other servers in the IdM domain.

NOTE

If you need to repair certificates across multiple CA servers:

1. After ensuring that LDAP replication is working across the topology, first run `ipa-cert-fix` on one CA server, according to the above procedure.

2. Before you run `ipa-cert-fix` on another CA server, trigger Certmonger renewals for shared certificates via `getcert-resubmit` (on the other CA server), to avoid unnecessary renewal of shared certificates.

58.2. VERIFYING OTHER IDM SERVERS IN THE IDM DOMAIN AFTER RENEWAL

After the renewing the CA renewal server’s certificates with the `ipa-cert-fix` tool, you must:

- Restart all other Identity Management (IdM) servers in the domain.
- Check if certmonger renewed certificates.
- If there are other Certificate Authority (CA) replicas with expired system certificates, renew those certificates with the `ipa-cert-fix` tool as well.

Prerequisites

- Log in to the server with administration rights.

Procedure

1. Restart IdM with the `--force` parameter:

```
# ipactl restart --force
```

With the `--force` parameter, the `ipactl` utility ignores individual service startup failures. For example, if the server is also a CA with expired certificates, the `pki-tomcat` service fails to start. This is expected and ignored because of using the `--force` parameter.

2. After the restart, verify that the `certmonger` service renewed the certificates (certificate status says MONITORING):
# getcert list | egrep '^Request|status:|subject:'
Request ID '20190522120745':
  status: MONITORING
  subject: CN=IPA RA,O=EXAMPLE.COM 201905222205
Request ID '20190522120834':
  status: MONITORING
  subject: CN=Certificate Authority,O=EXAMPLE.COM 201905222205
...

It can take some time before certmonger renews the shared certificates on the replica.

3. If the server is also a CA, the previous command reports CA_UNREACHABLE for the certificate the pki-tomcat service uses:

Request ID '20190522120835':
  status: CA_UNREACHABLE
  subject: CN=ca2.example.com,O=EXAMPLE.COM 201905222205
...

4. To renew this certificate, use the ipa-cert-fx utility:

```
# ipa-cert-fix
Dogtag sslserver certificate:
  Subject: CN=ca2.example.com,O=EXAMPLE.COM
  Serial: 3

Enter "yes" to proceed: yes
Proceeding.
Renewed Dogtag sslserver certificate:
  Subject: CN=ca2.example.com,O=EXAMPLE.COM 201905222205
  Serial: 15
  Expires: 2019-08-14 04:25:05
```

The ipa-cert-fix command was successful.

Now, all IdM certificates have been renewed and work correctly.

58.3. REPLACING THE WEB SERVER AND LDAP SERVER CERTIFICATES

As an Identity Management (IdM) system administrator, you can manually replace the certificates for the web (or httpd) and LDAP (or Directory) services running on an IdM server. For example, this might be necessary if the certmonger utility is not configured to renew the certificates automatically, or if the certificates are signed by an external certificate authority (CA).

The example installs the certificates for the services running on the server.idm.example.com IdM server. You obtain the certificates from an external CA.

**NOTE**

The HTTP and LDAP service certificates have different keypairs and subject names on different IdM servers and so you must renew the certificates on each IdM server individually.
Prerequisites

- You have `root` access to the IdM server.
- You know the Directory Manager password.
- You have access to a file storing the CA certificate chain of the external CA, `ca_certificate_chain_file.crt`.

Procedure

1. Install the certificates contained in `ca_certificate_chain_file.crt` as additional CA certificates to IdM:
   ```
   # ipa-cacert-manage install
   ```

2. Update the local IdM certificate databases with certificates from `ca_certificate_chain_file.crt`:
   ```
   # ipa-certupdate
   ```

3. Generate a private key and a certificate signing request (CSR) using the OpenSSL utility:
   ```
   $ openssl req -new -newkey rsa:2048 -days 365 -nodes -keyout new.key -out new.csr -addext "subjectAltName = DNS:ipa-ca.idm.example.test" -subj /CN=server.idm.example.com,O=IDM.EXAMPLE.COM
   ```
   Submit the CSR to the external CA. The process differs depending on the service to be used as the external CA. After the CA signs the certificate, import the certificate to the IdM server.

4. On the IdM server, replace the Apache web server’s old private key and certificate with the new key and the newly-signed certificate:
   ```
   # ipa-server-certinstall -w --pin= password new.key new.crt
   ```
   In the command above:
   - The `-w` option specifies that you are installing a certificate into the web server.
   - The `--pin` option specifies the password protecting the private key.

5. When prompted, enter the Directory Manager password.

6. Replace the LDAP server’s old private key and certificate with the new key and the newly-signed certificate:
   ```
   # ipa-server-certinstall -d --pin= password new.key new.cert
   ```
   In the command above:
   - The `-d` option specifies that you are installing a certificate into the LDAP server.
   - The `--pin` option specifies the password protecting the private key.

7. When prompted, enter the Directory Manager password.
8. Restart the **httpd** service:

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

9. Restart the **Directory** service:

```
# systemctl restart dirsrv@IDM.EXAMPLE.COM.service
```

### Additional resources

- Converting certificate formats to work with IdM
- The **ipa-server-certinstall(1)** man page
CHAPTER 59. GENERATING CRL ON THE IDM CA SERVER

If your IdM deployment uses an embedded certificate authority (CA), you may need to move generating the Certificate Revocation List (CRL) from one Identity Management (IdM) server to another. It can be necessary, for example, when you want to migrate the server to another system.

Only configure one server to generate the CRL. The IdM server that performs the CRL publisher role is usually the same server that performs the CA renewal server role, but this is not mandatory. Before you decommission the CRL publisher server, select and configure another server to perform the CRL publisher role.

This chapter describes:

- Stopping CRL generation on the IdM server.
- Starting to generate CRL on the IdM replica.

59.1. STOPPING CRL GENERATION ON AN IDM SERVER

To stop generating the Certificate Revocation List (CRL) on the IdM CRL publisher server, use the `ipa-crlgen-manage` command. Before you disable the generation, verify that the server really generates CRL. You can then disable it.

**Prerequisites**

- You must be logged in as root.

**Procedure**

1. Check if your server is generating the CRL:

   ```
   [root@server ~]# ipa-crlgen-manage status
   CRL generation: enabled
   Last CRL update: 2019-10-31 12:00:00
   Last CRL Number: 6
   The ipa-crlgen-manage command was successful
   ```

2. Stop generating the CRL on the server:

   ```
   [root@server ~]# ipa-crlgen-manage disable
   Stopping pki-tomcatd
   Editing /var/lib/pki/pki-tomcat/conf/ca/CS.cfg
   Starting pki-tomcatd
   Editing /etc/httpd/conf.d/ipa-pki-proxy.conf
   Restarting httpd
   CRL generation disabled on the local host. Please make sure to configure CRL generation on another master with ipa-crlgen-manage enable.
   The ipa-crlgen-manage command was successful
   ```

3. Check if the server stopped generating CRL:

   ```
   [root@server ~]# ipa-crlgen-manage status
   ```

   The server stopped generating the CRL. The next step is to enable CRL generation on the IdM replica.
59.2. STARTING CRL GENERATION ON AN IDM REPLICA SERVER

You can start generating the Certificate Revocation List (CRL) on an IdM CA server with the `ipa-crlgen-manage` command.

Prerequisites

- The RHEL system must be an IdM Certificate Authority server.
- You must be logged in as root.

Procedure

1. Start generating the CRL:

   ```
   [root@replica1 ~]# ipa-crlgen-manage enable
   Stopping pki-tomcatd
   Editing /var/lib/pki/pki-tomcat/conf/ca/CS.cfg
   Starting pki-tomcatd
   Editing /etc/httpd/conf.d/ipa-pki-proxy.conf
   Restarting httpd
   Forcing CRL update
   CRL generation enabled on the local host. Please make sure to have only a single CRL generation master.
   The ipa-crlgen-manage command was successful
   ```

2. Check if the CRL is generated:

   ```
   [root@replica1 ~]# ipa-crlgen-manage status
   CRL generation: enabled
   Last CRL update: 2019-10-31 12:10:00
   Last CRL Number: 7
   The ipa-crlgen-manage command was successful
   ```
CHAPTER 60. OBTAINING AN IDM CERTIFICATE FOR A SERVICE USING CERTMONGER

60.1. CERTMONGER OVERVIEW

What certmonger does
When Identity Management (IdM) is installed with an integrated IdM Certificate Authority (CA), it uses the certmonger service to track and renew system and service certificates. When the certificate is reaching its expiration date, certmonger manages the renewal process by:

- regenerating a certificate-signing request (CSR) using the options provided in the original request.
- submitting the CSR to the IdM CA using the IdM API cert-request command.
- receiving the certificate from the IdM CA.
- executing a pre-save command if specified by the original request.
- installing the new certificate in the location specified in the renewal request: either in an NSS database or in a file.
- executing a post-save command if specified by the original request. For example, the post-save command can instruct certmonger to restart a relevant service, so that the service picks up the new certificate.

Types of certificates certmonger tracks
Certificates can be divided into system and service certificates.

Unlike service certificates (for example, for HTTP, LDAP and PKINIT), which have different keypairs and subject names on different servers, IdM system certificates and their keys are shared by all CA replicas. The IdM system certificates include:

- IdM CA certificate
- OCSP signing certificate
- IdM CA subsystem certificates
- IdM CA audit signing certificate
- IdM renewal agent (RA) certificate
- KRA transport and storage certificates

The certmonger service tracks the IdM system and service certificates that were requested during the installation of IdM environment with an integrated CA. Certmonger also tracks certificates that have been requested manually by the system administrator for other services running on the IdM host. Certmonger does not track external CA certificates or user certificates.

Certmonger components
The certmonger service consists of two main components:

- The certmonger daemon, which is the engine tracking the list of certificates and launching renewal commands
The `getcert` utility for the command-line interface (CLI), which allows the system administrator to actively send commands to the `certmonger` daemon.

More specifically, the system administrator can use the `getcert` utility to:

- Request a new certificate
- View the list of certificates that `certmonger` tracks
- Start or stop tracking a certificate
- Renew a certificate

### 60.2. Obtaining an IDM Certificate for a Service Using `certmonger`

To ensure that communication between browsers and the web service running on your Identity Management (IdM) client is secure and encrypted, use a TLS certificate. Obtain the TLS certificate for your web service from the IdM Certificate Authority (CA).

This section describes how to use `certmonger` to obtain an IdM certificate for a service (HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM) running on an IdM client.

Using `certmonger` to request the certificate automatically means that `certmonger` manages and renews the certificate when it is due for a renewal.

For a visual representation of what happens when `certmonger` requests a service certificate, see Section 60.3, “Communication flow for `certmonger` requesting a service certificate”.

#### Prerequisites

- The web server is enrolled as an IdM client.
- You have root access to the IdM client on which you are running the procedure.
- The service for which you are requesting a certificate does not have to pre-exist in IdM.

#### Procedure

1. On the `my_company.idm.example.com` IdM client on which the HTTP service is running, request a certificate for the service corresponding to the `HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM` principal, and specify that

   - The certificate is to be stored in the local `/etc/pki/tls/certs/httpd.pem` file
   - The private key is to be stored in the local `/etc/pki/tls/private/httpd.key` file
   - That an extensionRequest for a SubjectAltName be added to the signing request with the DNS name of `my_company.idm.example.com`:

   ```
   # ipa-getcert request -K HTTP/my_company.idm.example.com -k /etc/pki/tls/private/httpd.key -f /etc/pki/tls/certs/httpd.pem -g 2048 -D my_company.idm.example.com -C "systemctl restart httpd"
   New signing request “20190604065735” added.
   ```

In the command above:
The `ipa-getcert request` command specifies that the certificate is to be obtained from the IdM CA. The `ipa-getcert request` command is a shortcut for `getcert request -c IPA`.

- The `-g` option specifies the size of key to be generated if one is not already in place.
- The `-D` option specifies the SubjectAltName DNS value to be added to the request.
- The `-C` option instructs certmonger to restart the `httpd` service after obtaining the certificate.
- To specify that the certificate be issued with a particular profile, use the `-T` option.
- To request a certificate using the named issuer from the specified CA, use the `-X ISSUER` option.

2. Optionally, to check the status of your request:

```shell
# ipa-getcert list -f /etc/pki/tls/certs/httpd.pem
Number of certificates and requests being tracked: 3.
Request ID '20190604065735':
  status: MONITORING
  stuck: no
  key pair storage: type=FILE,location="/etc/pki/tls/private/httpd.key"
  certificate: type=FILE,location="/etc/pki/tls/certs/httpd.crt"
  CA: IPA

[...]
```

The output shows that the request is in the MONITORING status, which means that a certificate has been obtained. The locations of the key pair and the certificate are those requested.

### 60.3. COMMUNICATION FLOW FOR CERTMONGER REQUESTING A SERVICE CERTIFICATE

The diagrams in this section show the stages of what happens when certmonger requests a service certificate from Identity Management (IdM) certificate authority (CA) server. The sequence consists of these diagrams:

- Figure 60.1, "Unencrypted communication"
- Figure 60.2, "Certmonger requesting a service certificate"
- Figure 60.3, "IdM CA issuing the service certificate"
- Figure 60.4, "Certmonger applying the service certificate"
- Figure 60.5, "Certmonger requesting a new certificate when the old one is nearing expiration"

Figure 60.1, "Unencrypted communication" shows the initial situation: without an HTTPS certificate, the communication between the web server and the browser is unencrypted.
Figure 60.1. Unencrypted communication

Figure 60.2, “Certmonger requesting a service certificate” shows the system administrator using certmonger to manually request an HTTPS certificate for the Apache web server. Note that when requesting a web server certificate, certmonger does not communicate directly with the CA. It proxies through IdM.

Figure 60.2. Certmonger requesting a service certificate

Figure 60.3, “IdM CA issuing the service certificate” shows an IdM CA issuing an HTTPS certificate for the web server.
Figure 60.3. IdM CA issuing the service certificate

Figure 60.4, “Certmonger applying the service certificate” shows certmonger placing the HTTPS certificate in appropriate locations on the IdM client and, if instructed to do so, restarting the httpd service. The Apache server subsequently uses the HTTPS certificate to encrypt the traffic between itself and the browser.

Figure 60.5, “Certmonger requesting a new certificate when the old one is nearing expiration”
Figure 60.5, “Certmonger requesting a new certificate when the old one is nearing expiration” shows certmonger automatically requesting a renewal of the service certificate from the IdM CA before the expiration of the certificate. The IdM CA issues a new certificate.

Figure 60.5. Certmonger requesting a new certificate when the old one is nearing expiration

---

60.4. VIEWING THE DETAILS OF A CERTIFICATE REQUEST TRACKED BY CERTMONGER

The certmonger service monitors certificate requests. When a request for a certificate is successfully signed, it results in a certificate. Certmonger manages certificate requests including the resulting certificates. This section describes how to view the details of a particular certificate request managed by certmonger.

Procedure

- If you know how to specify the certificate request, list the details of only that particular certificate request. You can, for example, specify:
  - The request ID
  - The location of the certificate
  - The certificate nickname

For example, to view the details of the certificate whose request ID is 20190408143846, using the -v option to view all the details of errors in case your request for a certificate was unsuccessful:

```
# getcert list -i 20190408143846 -v
```

Number of certificates and requests being tracked: 16.
The output displays several pieces of information about the certificate, for example:

- the certificate location; in the example above, it is the NSS database in the `/etc/dirsrv/slapd-IDM-EXAMPLE-COM` directory
- the certificate nickname; in the example above, it is `Server-Cert`
- the file storing the pin; in the example above, it is `/etc/dirsrv/slapd-IDM-EXAMPLE-COM/pwdfile.txt`
- the Certificate Authority (CA) that will be used to renew the certificate; in the example above, it is the IPA CA
- the expiration date; in the example above, it is `2021-04-08 16:38:47 CEST`
- the status of the certificate; in the example above, the `MONITORING` status means that the certificate is valid and it is being tracked
- the post-save command; in the example above, it is the restart of the LDAP service

If you do not know how to specify the certificate request, list the details of all the certificates that `certmonger` is monitoring or attempting to obtain:

```
# getcert list
```

Additional information

- To view the different options how to specify the certificate request displayed, see the `getcert list` man page.

## 60.5. STARTING AND STOPPING CERTIFICATE TRACKING

This section describes how you can use the `getcert stop-tracking` and `getcert start-tracking` commands to monitor certificates. The two commands are provided by the `certmonger` service.
Enabling certificate tracking is especially useful if you have imported a certificate issued by the Identity Management (IdM) certificate authority (CA) onto the machine from a different IdM client. Enabling certificate tracking can also be the final step of the following provisioning scenario:

1. On the IdM server, you create a certificate for a system that does not exist yet.
2. You create the new system.
3. You enroll the new system as an IdM client.
4. You import the certificate and the key from the IdM server on to the IdM client.
5. You start tracking the certificate using `certmonger` to ensure that it gets renewed when it is due to expire.

Procedure

- To disable the monitoring of a certificate with the Request ID of 20190408143846:

  ```
  # getcert stop-tracking -i 20190408143846
  ```

  For more options, see the `getcert stop-tracking` man page.

- To enable the monitoring of a certificate stored in the `/tmp/some_cert.crt` file, whose private key is stored in the `/tmp/some_key.key` file:

  ```
  # getcert start-tracking -c IPA -f /tmp/some_cert.crt -k /tmp/some_key.key
  ```

  Certmonger cannot automatically identify the CA type that issued the certificate. For this reason, add the `-c` option with the `IPA` value to the `getcert start-tracking` command if the certificate was issued by the IdM CA. Omitting to add the `-c` option results in `certmonger` entering the NEED_CA state.

  For more options, see the `getcert start-tracking` man page.

**NOTE**

The two commands do not manipulate the certificate. For example, `getcert stop-tracking` does not delete the certificate or remove it from the NSS database or from the filesystem but simply removes the certificate from the list of monitored certificates. Similarly, `getcert start-tracking` only adds a certificate to the list of monitored certificates.

### 60.6. RENEWING A CERTIFICATE MANUALLY

When a certificate is near its expiration date, the `certmonger` daemon automatically issues a renewal command using the certificate authority (CA) helper, obtains a renewed certificate and replaces the previous certificate with the new one.

It is also possible to manually renew a certificate in advance by using the `getcert resubmit` command. This way, you can update the information the certificate contains, e.g. by adding a Subject Alternative Name (SAN).

This section describes how to renew a certificate manually.
To renew a certificate with the Request ID of 20190408143846:

```
# getcert resubmit -i 20190408143846
```

To obtain the Request ID for a specific certificate, use the `getcert list` command. For details, see the `getcert list` man page.

### 60.7. MAKING CERTMONGER RESUME TRACKING OF IDM CERTIFICATES ON A CA REPLICA

This procedure shows how to make `certmonger` resume the tracking of Identity Management (IdM) system certificates that are crucial for an IdM deployment with an integrated certificate authority after the tracking of certificates was interrupted. The interruption may have been caused by the IdM host being unenrolled from IdM during the renewal of the system certificates or by replication topology not working properly. The procedure also shows how to make `certmonger` resume the tracking of the IdM service certificates, namely the HTTP, LDAP and PKINIT certificates.

**Prerequisites**

- The host on which you want to resume tracking system certificates is an IdM server that is also an IdM certificate authority (CA) but not the IdM CA renewal server.

**Procedure**

1. Get the PIN for the subsystem CA certificates:

   ```
   # grep 'internal=' /var/lib/pki/pki-tomcat/conf/password.conf
   ```

2. Add tracking to the subsystem CA certificates, replacing `[internal PIN]` in the commands below with the PIN obtained in the previous step:

   ```
   # getcert start-tracking -d /etc/pki/pki-tomcat/alias -n "caSigningCert cert-pki-ca" -c 'dogtag-ica-ca-renew-agent' -P [internal PIN] -B /usr/libexec/ipa/certmonger/stop_pkicad -C '/usr/libexec/ipa/certmonger/renew_ca_cert "caSigningCert cert-pki-ca"'
   
   # getcert start-tracking -d /etc/pki/pki-tomcat/alias -n "auditSigningCert cert-pki-ca" -c 'dogtag-ica-ca-renew-agent' -P [internal PIN] -B /usr/libexec/ipa/certmonger/stop_pkicad -C '/usr/libexec/ipa/certmonger/renew_ca_cert "auditSigningCert cert-pki-ca"'
   
   # getcert start-tracking -d /etc/pki/pki-tomcat/alias -n "ocspSigningCert cert-pki-ca" -c 'dogtag-ica-ca-renew-agent' -P [internal PIN] -B /usr/libexec/ipa/certmonger/stop_pkicad -C '/usr/libexec/ipa/certmonger/renew_ca_cert "ocspSigningCert cert-pki-ca"'
   
   # getcert start-tracking -d /etc/pki/pki-tomcat/alias -n "subsystemCert cert-pki-ca" -c 'dogtag-ica-ca-renew-agent' -P [internal PIN] -B /usr/libexec/ipa/certmonger/stop_pkicad -C '/usr/libexec/ipa/certmonger/renew_ca_cert "subsystemCert cert-pki-ca"'
   
   # getcert start-tracking -d /etc/pki/pki-tomcat/alias -n "Server-Cert cert-pki-ca" -c
   ```
3. Add tracking for the remaining IdM certificates, the HTTP, LDAP, IPA renewal agent and PKINIT certificates:

```bash
# getcert start-tracking -f /var/lib/ipa/certs/httpd.crt -k /var/lib/ipa/private/httpd.key -p /var/lib/ipa/passwds/idm.example.com-443-RSA -c IPA -C /usr/libexec/ipa/certmonger/restart_httpd

# getcert start-tracking -d /etc/dirsrv/slapd-IDM-EXAMPLE-COM -n "Server-Cert" -c IPA -p /etc/dirsrv/slapd-IDM-EXAMPLE-COM/pwdfile.txt -C '/usr/libexec/ipa/certmonger/restart_dirsrv "IDM-EXAMPLE-COM"'


# getcert start-tracking -f /var/kerberos/krb5kdc/kdc.crt -k /var/kerberos/krb5kdc/kdc.key -c dogtag-ipa-ca-renew-agent -B /usr/libexec/ipa/certmonger/renew_ra_cert_pre -C /usr/libexec/ipa/certmonger/renew_kdc_cert
```

4. Restart certmonger:

```bash
# systemctl restart certmonger
```

5. Wait for one minute after certmonger has started and then check the statuses of the new certificates:

```bash
# getcert list
```

### Additional resources

- If your IdM system certificates have all expired, follow the procedure described in [this Knowledge Centered Support (KCS) solution](https://example.com) to manually renew IdM system certificates on the IdM CA server that is also the CA renewal server and the CRL publisher server. Then follow the procedure described in [this KCS solution](https://example.com) to manually renew IdM system certificates on all the other CA servers in the topology.
CHAPTER 61. REQUESTING CERTIFICATES USING RHEL SYSTEM ROLES

You can use the Certificate System Role to issue and manage certificates.

This chapter covers the following topics:

- The Certificate System Role
- Requesting a new self-signed certificate using the Certificate System Role
- Requesting a new certificate from IdM CA using the Certificate System Role

61.1. THE CERTIFICATE SYSTEM ROLE

Using the Certificate System Role, you can manage issuing and renewing TLS and SSL certificates using Ansible Core.

The role uses certmonger as the certificate provider, and currently supports issuing and renewing self-signed certificates and using the IdM integrated certificate authority (CA).

You can use the following variables in your Ansible playbook with the Certificate System Role:

- **certificate_wait**
  to specify if the task should wait for the certificate to be issued.

- **certificate_requests**
  to represent each certificate to be issued and its parameters.

Additional resources

- For details about the parameters used in the certificate_requests variable and additional information about the certificate System Role, see the /usr/share/ansible/roles/rhel-system-roles.certificate/README.md file.

- For details about RHEL System Roles and how to apply them, see Getting started with RHEL System Roles.

61.2. REQUESTING A NEW SELF-SIGNED CERTIFICATE USING THE CERTIFICATE SYSTEM ROLE

With the Certificate System Role, you can use Ansible Core to issue self-signed certificates.

This process uses the certmonger provider and requests the certificate through the getcert command.

**NOTE**

By default, certmonger automatically tries to renew the certificate before it expires. You can disable this by setting the auto_renew parameter in the Ansible playbook to no.

Prerequisites

- The Ansible Core package is installed on the control machine.
You have the **rhel-system-roles** package installed on the system from which you want to run the playbook.

For details about RHEL System Roles and how to apply them, see [Getting started with RHEL System Roles](#).

### Procedure

1. **Optional:** Create an inventory file, for example `inventory.file`:
   ```
   $ touch inventory.file
   ```

2. Open your inventory file and define the hosts on which you want to request the certificate, for example:
   ```
   [webserver]
   server.idm.example.com
   ```

3. Create a playbook file, for example `request-certificate.yml`:
   - Set `hosts` to include the hosts on which you want to request the certificate, such as `webserver`.
   - Set the `certificate_requests` variable to include the following:
     - Set the `name` parameter to the desired name of the certificate, such as `mycert`.
     - Set the `dns` parameter to the domain to be included in the certificate, such as `*.example.com`.
     - Set the `ca` parameter to `self-sign`.
   - Set the `rhel-system-roles.certificate` role under `roles`. This is the playbook file for this example:
     ```yaml
     ---
     - hosts: webserver
       vars:
         certificate_requests:
           - name: mycert
             dns: "*.example.com"
             ca: self-sign
       roles:
         - rhel-system-roles.certificate
     ```

4. Save the file.

5. Run the playbook:
   ```
   $ ansible-playbook -i inventory.file request-certificate.yml
   ```

### Additional resources

For details about the parameters used in the `certificate_requests` variable and additional Red Hat Enterprise Linux 9.0 Beta Configuring and managing Identity Management 512
61.3. REQUESTING A NEW CERTIFICATE FROM IDM CA USING THE CERTIFICATE SYSTEM ROLE

With the Certificate System Role, you can use ansible-core to issue certificates while using an IdM server with an integrated certificate authority (CA). Therefore, you can efficiently and consistently manage the certificate trust chain for multiple systems when using IdM as the CA.

This process uses the certmonger provider and requests the certificate through the getcert command.

**NOTE**

By default, certmonger automatically tries to renew the certificate before it expires. You can disable this by setting the auto_renew parameter in the Ansible playbook to no.

**Procedure**

1. *Optional:* Create an inventory file, for example inventory.file:

   ```bash
   $ touch inventory.file
   ```

2. Open your inventory file and define the hosts on which you want to request the certificate, for example:

   ```ini
   [webserver]
   server.idm.example.com
   ```

3. Create a playbook file, for example request-certificate.yml:

   ```yaml
   - name: Request a new certificate
     hosts: webserver
     vars:
       certificate_requests:
         name: mycert
         dns: www.example.com
         principal: example@domain.com
   ```
Set the principal parameter to specify the Kerberos principal, such as HTTP/www.example.com@EXAMPLE.COM.

- Set the ca parameter to ipa.

- Set the rhel-system-roles.certificate role under roles.

This is the playbook file for this example:

```yaml
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
dns: www.example.com
    principal: HTTP/www.example.com@EXAMPLE.COM
    ca: ipa
  roles:
    - rhel-system-roles.certificate

4. Save the file.

5. Run the playbook:

   $ ansible-playbook -i inventory.file request-certificate.yml

Additional resources

- For details about the parameters used in the certificate_requests variable and additional information about the certificate System Role, see the /usr/share/ansible/roles/rhel-system-roles.certificate/README.md file.

- For details about the ansible-playbook command, see the ansible-playbook(1) man page.

61.4. SPECIFYING COMMANDS TO RUN BEFORE OR AFTER CERTIFICATE ISSUANCE USING THE CERTIFICATE SYSTEM ROLE

With the Certificate System Role, you can use Ansible Core to execute a command before and after a certificate is issued or renewed.

In the following example, the administrator ensures stopping the httpd service before a self-signed certificate for www.example.com is issued or renewed, and restarting it afterwards.

**NOTE**

By default, certmonger automatically tries to renew the certificate before it expires. You can disable this by setting the auto_renew parameter in the Ansible playbook to no.

Prerequisites

- The Ansible Core package is installed on the control machine.

- You have the rhel-system-roles package installed on the system from which you want to run the playbook.
For details about RHEL System Roles and how to apply them, see Getting started with RHEL System Roles.

**Procedure**

1. **Optional:** Create an inventory file, for example `inventory.file`:

   ```
   $ touch inventory.file
   ```

2. Open your inventory file and define the hosts on which you want to request the certificate, for example:

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

3. Create a playbook file, for example `request-certificate.yml`:

   - Set `hosts` to include the hosts on which you want to request the certificate, such as `webserver`.
   - Set the `certificate_requests` variable to include the following:
     - Set the `name` parameter to the desired name of the certificate, such as `mycert`.
     - Set the `dns` parameter to the domain to be included in the certificate, such as `www.example.com`.
     - Set the `ca` parameter to the CA you want to use to issue the certificate, such as `self-sign`.
     - Set the `run_before` parameter to the command you want to execute before this certificate is issued or renewed, such as `systemctl stop httpd.service`.
     - Set the `run_after` parameter to the command you want to execute after this certificate is issued or renewed, such as `systemctl start httpd.service`.
   - Set the `rhel-system-roles.certificate` role under `roles`. This is the playbook file for this example:

   ```yaml
   ---
   - hosts: webserver
     vars:
       certificate_requests:
         - name: mycert
dns: www.example.com
ca: self-sign
run_before: systemctl stop httpd.service
run_after: systemctl start httpd.service

   roles:
     - rhel-system-roles.certificate
   ```

4. Save the file.

5. Run the playbook:

   ```
   ```
$ ansible-playbook -i inventory.file request-certificate.yml

Additional resources

- For details about the parameters used in the `certificate_requests` variable and additional information about the `certificate` System Role, see the `/usr/share/ansible/roles/rhel-system-roles.certificate/README.md` file.

- For details about the `ansible-playbook` command, see the `ansible-playbook(1)` man page.
CHAPTER 62. RESTRICTING AN APPLICATION TO TRUST ONLY A SUBSET OF CERTIFICATES

If your Identity Management (IdM) installation is configured with the integrated Certificate System (CS) certificate authority (CA), you are able to create lightweight sub-CAs. All sub-CAs you create are subordinated to the primary CA of the certificate system, the **ipa CA**.

A **lightweight sub-CA** in this context means *a sub-CA issuing certificates for a specific purpose*. For example, a lightweight sub-CA enables you to configure a service, such as a virtual private network (VPN) gateway and a web browser, to accept only certificates issued by *sub-CA A*. By configuring other services to accept certificates only issued by *sub-CA B*, you prevent them from accepting certificates issued by *sub-CA A*, the primary CA, that is the **ipa CA**, and any intermediate sub-CA between the two.

If you revoke the intermediate certificate of a sub-CA, **all certificates issued by this sub-CA are automatically considered invalid** by correctly configured clients. All the other certificates issued directly by the root CA, **ipa**, or another sub-CA, remain valid.

This section uses the example of the Apache web server to illustrate how to restrict an application to trust only a subset of certificates. Complete this section to restrict the web server running on your IdM client to use a certificate issued by the **webserver-ca** IdM sub-CA, and to require the users to authenticate to the web server using user certificates issued by the **webclient-ca** IdM sub-CA.

The steps you need to take are:

1. Create an IdM sub-CA
2. Download the sub-CA certificate from IdM WebUI
3. Create a CA ACL specifying the correct combination of users, services and CAs, and the certificate profile used
4. Request a certificate for the web service running on an IdM client from the IdM sub-CA
5. Set up a single-instance Apache HTTP Server
6. Add TLS encryption to the Apache HTTP Server
7. Set the supported TLS protocol versions on an Apache HTTP Server
8. Set the supported ciphers on the Apache HTTP Server
9. Configure TLS client certificate authentication on the web server
10. Request a certificate for the user from the IdM sub-CA and export it to the client
11. Import the user certificate into the browser and configure the browser to trust the sub-CA certificate

62.1. MANAGING LIGHTWEIGHT SUB-CAS

This section describes how to manage lightweight subordinate certificate authorities (sub-CAs). All sub-CAs you create are subordinated to the primary CA of the certificate system, the **ipa CA**. You can also disable and delete sub-CAs.
NOTE

- If you delete a sub-CA, revocation checking for that sub-CA will no longer work. Only delete a sub-CA when there are no more certificates that were issued by that sub-CA whose `notAfter` expiration time is in the future.

- You should only disable sub-CAs while there are still non-expired certificates that were issued by that sub-CA. If all certificates that were issued by a sub-CA have expired, you can delete that sub-CA.

- You cannot disable or delete the IdM CA.

For details on managing sub-CAs, see:

- Section 62.1.1, “Creating a sub-CA from the IdM WebUI”
- Section 62.1.2, “Deleting a sub-CA from the IdM WebUI”
- Section 62.1.3, “Creating a sub-CA from the IdM CLI”
- Section 62.1.4, “Disabling a sub-CA from the IdM CLI”
- Section 62.1.5, “Deleting a sub-CA from the IdM CLI”

62.1.1. Creating a sub-CA from the IdM WebUI

This procedure describes how to use the IdM WebUI to create new sub-CAs named `webserver-ca` and `webclient-ca`.

Prerequisites

- Make sure you have obtained the administrator’s credentials.

Procedure

1. In the Authentication menu, click Certificates.

2. Select Certificate Authorities and click Add.

3. Enter the name of the `webserver-ca` sub-CA. Enter the Subject DN, for example `CN=WEBSERVER,O=IDM.EXAMPLE.COM`, in the Subject DN field. Note that the Subject DN must be unique in the IdM CA infrastructure.

4. Enter the name of the `webclient-ca` sub-CA. Enter the Subject DN `CN=WEBCLIENT,O=IDM.EXAMPLE.COM` in the Subject DN field.

5. In the command-line interface, run the `ipa-certupdate` command to create a certmonger tracking request for the `webserver-ca` and `webclient-ca` sub-CA certificates:

   ```bash
   [root@ipaserver ~]# ipa-certupdate
   ```
IMPORTANT

Forgetting to run the `ipa-certupdate` command after creating a sub-CA means that if the sub-CA certificate expires, end-entity certificates issued by the sub-CA are considered invalid even if the end-entity certificate has not expired.

Verification

- Verify that the signing certificate of the new sub-CA has been added to the IdM database:

  ```bash
  [root@ipaserver ~]# certutil -d /etc/pki/pki-tomcat/alias/ -L
  Certificate Nickname                      Trust Attributes
  SSL,S/MIME,JAR/XPI                        CTu,Cu,Cu
  caSigningCert cert-pki-ca                 u,u,u
  Server-Cert cert-pki-ca                   u,u,u
  auditSigningCert cert-pki-ca              u,u,Pu
  caSigningCert cert-pki-ca ba83f324-5e50-4114-b109-acca05d6f1dc u,u,u
  ocspSigningCert cert-pki-ca               u,u,u
  subsystemCert cert-pki-ca                 u,u,u
  ```

NOTE

The new sub-CA certificate is automatically transferred to all the replicas that have a certificate system instance installed.

62.1.2. Deleting a sub-CA from the IdM WebUI

This procedure describes how to delete lightweight sub-CAs in the IdM WebUI.

NOTE

- If you delete a sub-CA, revocation checking for that sub-CA will no longer work. Only delete a sub-CA when there are no more certificates that were issued by that sub-CA whose `notAfter` expiration time is in the future.

- You should only disable sub-CAs while there are still non-expired certificates that were issued by that sub-CA. If all certificates that were issued by a sub-CA have expired, you can delete that sub-CA.

- You cannot disable or delete the IdM CA.

Prerequisites

- Make sure you have obtained the administrator’s credentials.

- You have disabled the sub-CA in the IdM CLI. See Disabling a sub-CA from the IdM CLI

Procedure

1. In the IdM WebUI, open the Authentication tab, and select the Certificates subtab.

2. Select Certificate Authorities.
3. Select the sub-CA to remove and click **Delete**.

**Figure 62.1. Deleting a sub-CA in the IdM Web UI**

4. Click **Delete** to confirm.

The sub-CA is removed from the list of **Certificate Authorities**.

### 62.1.3. Creating a sub-CA from the IdM CLI

This procedure describes how to use the IdM CLI to create new sub-CAs named `webserver-ca` and `webclient-ca`.

**Prerequisites**

- Make sure that you have obtained the administrator’s credentials.
- Make sure you are logged in to an IdM server that is a CA server.

**Procedure**

1. Enter the `ipa ca-add` command, and specify the name of the `webserver-ca` sub-CA and its Subject Distinguished Name (DN):

   ```bash
   [root@ipaserver ~]# ipa ca-add webserver-ca --subject="CN=WEBSERVER,O=IDM.EXAMPLE.COM"
   -------------------
   Created CA "webserver-ca"
   -------------------
   Name: webserver-ca
   Authority ID: ba83f324-5e50-4114-b109-acca05d6f1dc
   Subject DN: CN=WEBSERVER,O=IDM.EXAMPLE.COM
   Issuer DN: CN=Certificate Authority,O=IDM.EXAMPLE.COM
   
   **Name**
   
   Name of the CA.
   
   **Authority ID**
   
   Automatically created, individual ID for the CA.
   
   **Subject DN**
   
   Subject Distinguished Name (DN). The Subject DN must be unique in the IdM CA infrastructure.
Issuer DN

Parent CA that issued the sub-CA certificate. All sub-CAs are created as a child of the IdM root CA.

2. Create the webclient-ca sub-CA for issuing certificates to web clients:

```
[root@ipaserver ~]# ipa ca-add webclient-ca --subject="CN=WEBCLIENT,O=IDM.EXAMPLE.COM"
```

Created CA "webclient-ca"

Name: webclient-ca
Authority ID: 8a479f3a-0454-4a4d-8ade-fd3b5a54ab2e
Subject DN: CN=WEBCLIENT,O=IDM.EXAMPLE.COM
Issuer DN: CN=Certificate Authority,O=IDM.EXAMPLE.COM

3. Run the `ipa-certupdate` command to create a certmonger tracking request for the webserver-ca and webclient-ca sub-CAs certificates:

```
[root@ipaserver ~]# ipa-certupdate
```

**IMPORTANT**

If you forget to run the `ipa-certupdate` command after creating a sub-CA and the sub-CA certificate expires, end-entity certificates issued by that sub-CA are considered invalid even though the end-entity certificate has not expired.

Verification steps

- Verify that the signing certificate of the new sub-CA has been added to the IdM database:

```
[root@ipaserver ~]# certutil -d /etc/pki/pki-tomcat/alias/ -L
```

<table>
<thead>
<tr>
<th>Certificate Nickname</th>
<th>Trust Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL,S/MIME,JAR/XPI</td>
<td></td>
</tr>
<tr>
<td>caSigningCert cert-pki-ca</td>
<td>CTu,Cu,Cu</td>
</tr>
<tr>
<td>Server-Cert cert-pki-ca</td>
<td>u,u,u</td>
</tr>
<tr>
<td>auditSigningCert cert-pki-ca</td>
<td>u,u,Pu</td>
</tr>
<tr>
<td>caSigningCert cert-pki-ca ba83f324-5e50-4114-b109-acca05d6f1dc</td>
<td>u,u,u</td>
</tr>
<tr>
<td>ocspSigningCert cert-pki-ca</td>
<td>u,u,u</td>
</tr>
<tr>
<td>subsystemCert cert-pki-ca</td>
<td>u,u,u</td>
</tr>
</tbody>
</table>

**NOTE**

The new sub-CA certificate is automatically transferred to all the replicas that have a certificate system instance installed.

62.1.4. Disabling a sub-CA from the IdM CLI

This procedure describes how to disable a sub-CA from the IdM CLI. If there are still non-expired certificates that were issued by a sub-CA, you should not delete it but you can disable it. If you delete the sub-CA, revocation checking for that sub-CA will no longer work.
Prerequisites

- Make sure you have obtained the administrator’s credentials.

Procedure

1. Run the `ipa ca-find` command to determine the name of the sub-CA you are deleting:

   ```
   [root@ipaserver ~]# ipa ca-find
   3 CAs matched
   Name: ipa
   Description: IPA CA
   Authority ID: 5195deaf-3b61-4aab-b608-317aff38497c
   Subject DN: CN=Certificate Authority,O=IPA.TEST
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Name: webclient-ca
   Authority ID: 605a472c-9c6e-425e-b959-f1955209b092
   Subject DN: CN=WEBCLIENT,O=IDM.EXAMPLE.COM
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Name: webserver-ca
   Authority ID: 02d537f9-c178-4433-98ea-53aa92126fc3
   Subject DN: CN=WEBSERVER,O=IDM.EXAMPLE.COM
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Number of entries returned 3
   ```

2. Run the `ipa ca-disable` command to disable your sub-CA, in this example, the `webserver-ca`:

   ```
   ipa ca-disable webserver-ca
   Disabled CA "webserver-ca"
   ```

62.15. Deleting a sub-CA from the IdM CLI

This procedure describes how to delete lightweight sub-CAs from the IdM CLI.

**NOTE**

- If you delete a sub-CA, revocation checking for that sub-CA will no longer work. Only delete a sub-CA when there are no more certificates that were issued by that sub-CA whose `notAfter` expiration time is in the future.

- You should only disable sub-CAs while there are still non-expired certificates that were issued by that sub-CA. If all certificates that were issued by a sub-CA have expired, you can delete that sub-CA.

- You cannot disable or delete the IdM CA.
Prerequisites

- Make sure you have obtained the administrator’s credentials.

Procedure

1. To display a list of sub-CAs and CAs, run the `ipa ca-find` command:

   ```
   # ipa ca-find
   3 CAs matched
   Name: ipa
   Description: IPA CA
   Authority ID: 5195deaf-3b61-4aab-b608-317aff38497c
   Subject DN: CN=Certificate Authority,O=IPA.TEST
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Name: webclient-ca
   Authority ID: 605a472c-9c6e-425e-b959-f1955209b092
   Subject DN: CN=WEBCLIENT,O=IDM.EXAMPLE.COM
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Name: webserver-ca
   Authority ID: 02d537f9-c178-4433-98ea-53aa92126fc3
   Subject DN: CN=WEBSERVER,O=IDM.EXAMPLE.COM
   Issuer DN: CN=Certificate Authority,O=IPA.TEST
   
   Number of entries returned 3
   ```

2. Run the `ipa ca-disable` command to disable your sub-CA, in this example, the `webserver-ca`:

   ```
   # ipa ca-disable webserver-ca
   Disabled CA "webserver-ca"
   ```

3. Delete the sub-CA, in this example, the `webserver-ca`:

   ```
   # ipa ca-del webserver-ca
   Deleted CA "webserver-ca"
   ```

Verification

- Run `ipa ca-find` to display the list of CAs and sub-CAs. The `webserver-ca` is no longer on the list.

   ```
   # ipa ca-find
   2 CAs matched
   ```
62.2. DOWNLOADING THE SUB-CA CERTIFICATE FROM IDM WEBUI

Prerequisites

- Make sure that you have obtained the IdM administrator’s credentials.

Procedure

1. In the Authentication menu, click Certificates > Certificates.

   Figure 62.2. sub-CA certificate in the list of certificates

   268173326  CN=WEBSERVER,O=IDM.EXAMPLE.COM  ipa  VALID
   268238849  CN=dm_user,O=IDM.EXAMPLE.COM  ipa  VALID

2. Click the serial number of the sub-CA certificate to open the certificate information page.

3. In the certificate information page, click Actions > Download.

4. In the CLI, move the sub-CA certificate to the /etc/pki/tls/private/ directory:

   # mv path/to/the/downloaded/certificate /etc/pki/tls/private/sub-ca.crt

62.3. CREATING CA ACLS FOR WEB SERVER AND CLIENT AUTHENTICATION

Certificate authority access control list (CA ACL) rules define which profiles can be used to issue certificates to which users, services, or hosts. By associating profiles, principals, and groups, CA ACLs permit principals or groups to request certificates using particular profiles.

For example, using CA ACLs, the administrator can restrict the use of a profile intended for employees working from an office located in London only to users that are members of the London office-related group.

62.3.1. Viewing CA ACLs in IdM CLI

Complete this section to view the list of certificate authority access control lists (CA ACLs) available in your IdM deployment and the details of a specific CA ACL.
Procedure

1. To view all the CA ACLs in your IdM environment, enter the **ipa caacl-find** command:

   ```
   $ ipa caacl-find
   -----------------  
   1 CA ACL matched
   -----------------  
   ACL name: hosts_services_calIPAserviceCert
   Enabled: TRUE
   ```

2. To view the details of a CA ACL, enter the **ipa caacl-show** command, and specify the CA ACL name. For example, to view the details of the **hosts_services_calIPAserviceCert** CA ACL, enter:

   ```
   $ ipa caacl-show hosts_services_calIPAserviceCert
   ACL name: hosts_services_calIPAserviceCert
   Enabled: TRUE
   Host category: all
   Service category: all
   CAs: ipa
   Profiles: calIPAserviceCert
   Users: admin
   ```

62.3.2. Creating a CA ACL for web servers authenticating to web clients using certificates issued by webserver-ca

This section describes how to create a CA ACL that requires the system administrator to use the `webserver-ca` sub-CA and the `calIPAserviceCert` profile when requesting a certificate for the `HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM` service. If the user requests a certificate from a different sub-CA or of a different profile, the request fails. The only exception is when there is another matching CA ACL that is enabled. To view the available CA ACLs, see View CA ACLs in IdM CLI.

Prerequisites

- Make sure that the `HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM` service is part of IdM.
- Make sure you have obtained IdM administrator’s credentials.

Procedure

1. Create a CA ACL using the **ipa caacl** command, and specify its name:

   ```
   $ ipa caacl-add TLS_web_server_authentication
   --------------------------------------------
   Added CA ACL "TLS_web_server_authentication"
   --------------------------------------------
   ACL name: TLS_web_server_authentication
   Enabled: TRUE
   ```

2. Modify the CA ACL using the **ipa caacl-mod** command to specify the description of the CA ACL:

   ```
   $ ipa caacl-mod TLS_web_server_authentication
   --------------------------------------------
   Added CA ACL "TLS_web_server_authentication"
   --------------------------------------------
   ACL name: TLS_web_server_authentication
   Enabled: TRUE
   ```
3. Add the `webserver-ca` sub-CA to the CA ACL:

```
$ ipa caacl-add-ca TLS_web_server_authentication --ca=webserver-ca
ACL name: TLS_web_server_authentication
Description: CAACL for web servers authenticating to web clients using certificates issued by webserver-ca
Enabled: TRUE
CAs: webserver-ca
```

Number of members added 1

4. Use the `ipa caacl-add-service` command to specify the service whose principal will be able to request a certificate:

```
$ ipa caacl-add-service TLS_web_server_authentication --service=HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM
ACL name: TLS_web_server_authentication
Description: CAACL for web servers authenticating to web clients using certificates issued by webserver-ca
Enabled: TRUE
CAs: webserver-ca
Services: HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM
```

Number of members added 1

5. Use the `ipa caacl-add-profile` command to specify the certificate profile for the requested certificate:

```
$ ipa caacl-add-profile TLS_web_server_authentication --certprofiles=calIPAServiceCert
ACL name: TLS_web_server_authentication
Description: CAACL for web servers authenticating to web clients using certificates issued by webserver-ca
Enabled: TRUE
CAs: webserver-ca
Profiles: calIPAServiceCert
Services: HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM
```

Number of members added 1

You can use the newly-created CA ACL straight away. It is enabled after its creation by default.
NOTE

The point of CA ACLs is to specify which CA and profile combinations are allowed for requests coming from particular principals or groups. CA ACLs do not affect certificate validation or trust. They do not affect how the issued certificates will be used.

62.3.3. Creating a CA ACL for user web browsers authenticating to web servers using certificates issued by webclient-ca

This section describes how to create a CA ACL that requires the system administrator to use the webclient-ca sub-CA and the IECUserRoles profile when requesting a certificate. If the user requests a certificate from a different sub-CA or of a different profile, the request fails. The only exception is when there is another matching CA ACL that is enabled. To view the available CA ACLs, see Viewing CA ACLs in IdM CLI.

Prerequisites

- Make sure that you have obtained IdM administrator’s credentials.

Procedure

1. Create a CA ACL using the `ipa caacl` command and specify its name:

```
$ ipa caacl-add TLS_web_client_authentication
--------------------------------------------
Added CA ACL "TLS_web_client_authentication"
--------------------------------------------
ACL name: TLS_web_client_authentication
Enabled: TRUE
```

2. Modify the CA ACL using the `ipa caacl-mod` command to specify the description of the CA ACL:

```
$ ipa caacl-mod TLS_web_client_authentication --desc="CAACL for user web browsers authenticating to web servers using certificates issued by webclient-ca"
-----------------------------------------------
Modified CA ACL "TLS_web_client_authentication"
-----------------------------------------------
ACL name: TLS_web_client_authentication
Description: CAACL for user web browsers authenticating to web servers using certificates issued by webclient-ca
Enabled: TRUE
```

3. Add the webclient-ca sub-CA to the CA ACL:

```
$ ipa caacl-add-ca TLS_web_client_authentication --ca=webclient-ca
ACL name: TLS_web_client_authentication
Description: CAACL for user web browsers authenticating to web servers using certificates issued by webclient-ca
Enabled: TRUE
CAs: webclient-ca
-------------------------
Number of members added 1
-------------------------
```
4. Use the `ipa caacl-add-profile` command to specify the certificate profile for the requested certificate:

```
$ ipa caacl-add-profile TLS_web_client_authentication --certprofiles=IECUserRoles
ACL name: TLS_web_client_authentication
Description: CAACL for user web browsers authenticating to web servers using certificates issued by webclient-ca
Enabled: TRUE
CAs: webclient-ca
Profiles: IECUserRoles
```

```
Number of members added 1
```

5. Modify the CA ACL using the `ipa caacl-mod` command to specify that the CA ACL applies to all IdM users:

```
$ ipa caacl-mod TLS_web_client_authentication --usercat=all
```

```
Modified CA ACL "TLS_web_client_authentication"
```

```
ACL name: TLS_web_client_authentication
Description: CAACL for user web browsers authenticating to web servers using certificates issued by webclient-ca
Enabled: TRUE
User category: all
CAs: webclient-ca
Profiles: IECUserRoles
```

You can use the newly-created CA ACL straight away. It is enabled after its creation by default.

**NOTE**

The point of CA ACLs is to specify which CA and profile combinations are allowed for requests coming from particular principals or groups. CA ACLs do not affect certificate validation or trust. They do not affect how the issued certificates will be used.

### 62.4. Obtaining an IdM Certificate for a Service Using Certmonger

To ensure that communication between browsers and the web service running on your IdM client is secure and encrypted, use a TLS certificate. If you want to restrict web browsers to trust certificates issued by the `webserver-ca` sub-CA but no other IdM sub-CA, obtain the TLS certificate for your web service from the `webserver-ca` sub-CA.

This section describes how to use `certmonger` to obtain an IdM certificate for a service (HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM) running on an IdM client.

Using `certmonger` to request the certificate automatically means that `certmonger` manages and renews the certificate when it is due for a renewal.

For a visual representation of what happens when `certmonger` requests a service certificate, see Section 62.5, “Communication flow for certmonger requesting a service certificate”.
Prerequisites

- The web server is enrolled as an IdM client.
- You have root access to the IdM client on which you are running the procedure.
- The service for which you are requesting a certificate does not have to pre-exist in IdM.

Procedure

1. On the my_company.idm.example.com IdM client on which the HTTP service is running, request a certificate for the service corresponding to the HTTP/my_company.idm.example.com@IDM.EXAMPLE.COM principal, and specify that
   - The certificate is to be stored in the local /etc/pki/tls/certs/httpd.pem file
   - The private key is to be stored in the local /etc/pki/tls/private/httpd.key file
   - The webserver-ca sub-CA is to be the issuing certificate authority
   - That an extensionRequest for a SubjectAltName be added to the signing request with the DNS name of my_company.idm.example.com:

```
# ipa-getcert request -K HTTP/my_company.idm.example.com -k /etc/pki/tls/private/httpd.key -f /etc/pki/tls/certs/httpd.pem -g 2048 -D my_company.idm.example.com -X webserver-ca -C "systemctl restart httpd"
```

New signing request "20190604065735" added.

In the command above:
- The ipa-getcert request command specifies that the certificate is to be obtained from the IdM CA. The ipa-getcert request command is a shortcut for getcert request -c IPA.
- The -g option specifies the size of key to be generated if one is not already in place.
- The -D option specifies the SubjectAltName DNS value to be added to the request.
- The -X option specifies that the issuer of the certificate must be webserver-ca, not ipa.
- The -C option instructs certmonger to restart the httpd service after obtaining the certificate.
- To specify that the certificate be issued with a particular profile, use the -T option.

2. Optionally, to check the status of your request:

```
# ipa-getcert list -f /etc/pki/tls/certs/httpd.pem
Number of certificates and requests being tracked: 3.
Request ID '20190604065735':
    status: MONITORING
    stuck: no
    key pair storage: type=FILE,location="/etc/pki/tls/private/httpd.key"
    certificate: type=FILE,location="/etc/pki/tls/certs/httpd.crt"
CA: IPA
```
The output shows that the request is in the **MONITORING** status, which means that a certificate has been obtained. The locations of the key pair and the certificate are those requested.

### 62.5. COMMUNICATION FLOW FOR CERTMONGER REQUESTING A SERVICE CERTIFICATE

The diagrams in this section show the stages of what happens when **certmonger** requests a service certificate from Identity Management (IdM) certificate authority (CA) server. The sequence consists of these diagrams:

- Figure 62.3, “Unencrypted communication”
- Figure 62.4, “Certmonger requesting a service certificate”
- Figure 62.5, "IdM CA issuing the service certificate"
- Figure 62.6, “Certmonger applying the service certificate”
- Figure 62.7, “Certmonger requesting a new certificate when the old one is nearing expiration”

In the diagrams, the **webserver-ca** sub-CA is represented by the generic **IdM CA server**.

Figure 62.3, “Unencrypted communication” shows the initial situation: without an HTTPS certificate, the communication between the web server and the browser is unencrypted.

Figure 62.4, “Certmonger requesting a service certificate” shows the system administrator using **certmonger** to manually request an HTTPS certificate for the Apache web server. Note that when
requesting a web server certificate, certmonger does not communicate directly with the CA. It proxies through IdM.

**Figure 62.4. Certmonger requesting a service certificate**

**Figure 62.5. “IdM CA issuing the service certificate”** shows an IdM CA issuing an HTTPS certificate for the web server.

**Figure 62.5. IdM CA issuing the service certificate**

**Figure 62.6. “Certmonger applying the service certificate”** shows certmonger placing the HTTPS
certificate in appropriate locations on the IdM client and, if instructed to do so, restarting the `httpd` service. The Apache server subsequently uses the HTTPS certificate to encrypt the traffic between itself and the browser.

**Figure 62.6. Certmonger applying the service certificate**

![Diagram showing the flow of a HTTPS connection](image)

**IdM client**

- **Apache web server**
  - my_company.idm.example.com
  - HTTPS certificate

- **Certmonger**
  - HTTPS certificate

**Figure 62.7. Certmonger requesting a new certificate when the old one is nearing expiration** shows `certmonger` automatically requesting a renewal of the service certificate from the IdM CA before the expiration of the certificate. The IdM CA issues a new certificate.
62.6. SETTING UP A SINGLE-INSTANCE APACHE HTTP SERVER

This section describes how to set up a single-instance Apache HTTP Server to serve static HTML content.

Follow the procedure in this section if the web server should provide the same content for all domains associated with the server. If you want to provide different content for different domains, set up name-based virtual hosts. For details, see Configuring Apache name-based virtual hosts.

Procedure

1. Install the httpd package:
   
   # dnf install httpd

2. Open the TCP port 80 in the local firewall:
   
   # firewall-cmd --permanent --add-port=80/tcp
   
   # firewall-cmd --reload

3. Enable and start the httpd service:
   
   # systemctl enable --now httpd

NOTE

When adding content to `/var/www/html/`, files and directories must be readable by the user under which `httpd` runs by default. The content owner can be the either the root user and root user group, or another user or group of the administrator’s choice. If the content owner is the root user and root user group, the files must be readable by other users. The SELinux context for all the files and directories must be `httpd_sys_content_t`, which is applied by default to all content within the `/var/www` directory.

**Verification steps**

  - If the `/var/www/html/` directory is empty or does not contain an `index.html` or `index.htm` file, Apache displays the Red Hat Enterprise Linux Test Page. If `/var/www/html/` contains HTML files with a different name, you can load them by entering the URL to that file, such as [http://server_IP/example.html](http://server_IP/example.html) or [http://my_company.idm.example.com/example.html](http://my_company.idm.example.com/example.html).

**Additional resources**

- For further details about configuring Apache and adapting the service to your environment, refer to the Apache manual. For details about installing the manual, see Installing the Apache HTTP Server manual.

- For details about using or adjusting the `httpd systemctl` service, see the `httpd.service(8) man` page.

### 62.7. ADDING TLS ENCRYPTION TO AN APACHE HTTP SERVER

This section describes how to enable TLS encryption on the `my_company.idm.example.com` Apache HTTP Server for the `idm.example.com` domain.

**Prerequisites**

- The `my_company.idm.example.com` Apache HTTP Server is installed and running.

- You have obtained the TLS certificate from the `webserver-ca` sub-CA, and stored it in the `/etc/pki/tls/certs/httpd.pem` file as described in Section 62.4, “Obtaining an IdM certificate for a service using certmonger”. If you use a different path, adapt the corresponding steps of the procedure.

- The corresponding private key is stored in the `/etc/pki/tls/private/httpd.key` file. If you use a different path, adapt the corresponding steps of the procedure.

- The `webserver-ca` CA certificate is stored in the `/etc/pki/tls/private/sub-ca.crt` file. If you use a different path, adapt the corresponding steps of the procedure.

- Clients and the `my_company.idm.example.com` web server resolve the host name of the server to the IP address of the web server.

**Procedure**

1. Install the `mod_ssl` package:

   ```bash
   # dnf install mod_ssl
   ```
2. Edit the `/etc/httpd/conf.d/ssl.conf` file and add the following settings to the `<VirtualHost _default_:443>` directive:

   a. Set the server name:

```
    ServerName my_company.idm.example.com
```

   **IMPORTANT**
   The server name must match the entry set in the **Common Name** field of the certificate.

   b. Optional: If the certificate contains additional host names in the **Subject Alt Names (SAN)** field, you can configure **mod_ssl** to provide TLS encryption also for these host names. To configure this, add the **ServerAliases** parameter with corresponding names:

```
    ServerAlias www.my_company.idm.example.com server.my_company.idm.example.com
```

   c. Set the paths to the private key, the server certificate, and the CA certificate:

```
    SSLCertificateKeyFile  "/etc/pki/tls/private/httpd.key"
    SSLCertificateFile  "/etc/pki/tls/certs/httpd.pem"
    SSLCACertificateFile  "/etc/pki/tls/certs/ca.crt"
```

3. For security reasons, configure that only the **root** user can access the private key file:

```
# chown root:root /etc/pki/tls/private/httpd.key
# chmod 600 /etc/pki/tls/private/httpd.key
```

   **WARNING**
   If the private key was accessed by unauthorized users, revoke the certificate, create a new private key, and request a new certificate. Otherwise, the TLS connection is no longer secure.

4. Open port **443** in the local firewall:

```
# firewall-cmd --permanent --add-port=443/tcp
# firewall-cmd --reload
```

5. Restart the **httpd** service:

```
# systemctl restart httpd
```
NOTE

If you protected the private key file with a password, you must enter this password each time when the httpd service starts.

- Use a browser and connect to https://my_company.idm.example.com.

Additional resources

- For further details about configuring TLS, refer to the SSL/TLS Encryption documentation in the Apache manual. For details about installing the manual, see Installing the Apache HTTP Server manual.
- Security considerations for TLS in RHEL 9-beta

62.8. SETTING THE SUPPORTED TLS PROTOCOL VERSIONS ON AN APACHE HTTP SERVER

By default, the Apache HTTP Server on RHEL uses the system-wide crypto policy that defines safe default values, which are also compatible with recent browsers. For example, the DEFAULT policy defines that only the TLSv1.2 and TLSv1.3 protocol versions are enabled in apache.

This section describes how to manually configure which TLS protocol versions your my_company.idm.example.com Apache HTTP Server supports. Follow the procedure if your environment requires to enable only specific TLS protocol versions, for example:

- If your environment requires that clients can also use the weak TLS1 (TLSv1.0) or TLS1.1 protocol.
- If you want to configure that Apache only supports the TLSv1.2 or TLSv1.3 protocol.

Prerequisites

- TLS encryption is enabled on the my_company.idm.example.com server as described in Section 62.7, “Adding TLS encryption to an Apache HTTP Server”.

Procedure

1. Edit the /etc/httpd/conf/httpd.conf file, and add the following setting to the <VirtualHost> directive for which you want to set the TLS protocol version. For example, to enable only the TLSv1.3 protocol:

   SSLProtocol -All TLSv1.3

2. Restart the httpd service:

   # systemctl restart httpd

Verification steps

1. Use the following command to verify that the server supports TLSv1.3:

   # openssl s_client -connect example.com:443 -tls1_3
2. Use the following command to verify that the server does not support TLSv1.2:

```bash
# openssl s_client -connect example.com:443 -tls1_2
```

If the server does not support the protocol, the command returns an error:

```
140111600609088:error:1409442E:SSL routines:ssl3_read_bytes:tlsv1 alert protocol version:ssl/record/rec_layer_s3.c:1543:SSL alert number 70
```

3. Optional: Repeat the command for other TLS protocol versions.

Additional resources

- For further details about the system-wide crypto policy, see the `update-crypto-policies(8)` man page and Using system-wide cryptographic policies.
- For further details about the `SSLProtocol` parameter, refer to the `mod_ssl` documentation in the Apache manual. For details about installing the manual, see Installing the Apache HTTP Server manual.

### 62.9. SETTING THE SUPPORTED CIPHERS ON AN APACHE HTTP SERVER

By default, the Apache HTTP Server uses the system-wide crypto policy that defines safe default values, which are also compatible with recent browsers. For the list of ciphers the system-wide crypto allows, see the `/etc/crypto-policies/back-ends/openssl.config` file.

This section describes how to manually configure which ciphers the `my_company.idm.example.com` Apache HTTP server supports. Follow the procedure if your environment requires specific ciphers.

**Prerequisites**

- TLS encryption is enabled on the `my_company.idm.example.com` server as described in Section 62.7, “Adding TLS encryption to an Apache HTTP Server”.

**Procedure**

1. Edit the `/etc/httpd/conf/httpd.conf` file, and add the `SSLCipherSuite` parameter to the `<VirtualHost>` directive for which you want to set the TLS ciphers:

   ```
   SSLCipherSuite
   "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH:!SHA1:!SHA256"
   ```

   This example enables only the EECDH+AESGCM, EDH+AESGCM, AES256+EECDH, and AES256+EDH ciphers and disables all ciphers which use the SHA1 and SHA256 message authentication code (MAC).

2. Restart the `httpd` service:

   ```bash
   # systemctl restart httpd
   ```

**Verification steps**
1. To display the list of ciphers the Apache HTTP Server supports:
   a. Install the nmap package:
      # dnf install nmap
   b. Use the nmap utility to display the supported ciphers:
      # nmap --script ssl-enum-ciphers -p 443 example.com
      ...
      PORT    STATE SERVICE
      443/tcp open  https
      | ssl-enum-ciphers:
      |   TLSv1.2:
      |     ciphers:
      |       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
      |       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
      |       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
      ...

Additional resources

- For further details about the system-wide crypto policy, see the update-crypto-policies(8) man page and Using system-wide cryptographic policies.
- For further details about the SSLCipherSuite parameter, refer to the mod_ssl documentation in the Apache manual. For details about installing the manual, see Installing the Apache HTTP Server manual.

62.10. CONFIGURING TLS CLIENT CERTIFICATE AUTHENTICATION

Client certificate authentication enables administrators to allow only users who authenticate using a certificate to access resources on the my_company.idm.example.com web server. This section describes how to configure client certificate authentication for the /var/www/html/Example/ directory.

**IMPORTANT**

If the my_company.idm.example.com Apache server uses the TLS 1.3 protocol, certain clients require additional configuration. For example, in Firefox, set the security.tls.enable_post_handshake_auth parameter in the about:config menu to true. For further details, see Transport Layer Security version 1.3 in Red Hat Enterprise Linux 8.

Prerequisites

- TLS encryption is enabled on the my_company.idm.example.com server as described in Section 62.7, “Adding TLS encryption to an Apache HTTP Server”.

Procedure

1. Edit the /etc/httpd/conf/httpd.conf file and add the following settings to the <VirtualHost> directive for which you want to configure client authentication:
The SSLVerifyClient require setting defines that the server must successfully validate the client certificate before the client can access the content in the /var/www/html/Example/ directory.

2. Restart the **httpd** service:

```bash
# systemctl restart httpd
```

**Verification steps**

1. Use the **curl** utility to access the **https://my_company.idm.example.com/Example/** URL without client authentication:

   ```bash
   $ curl https://my_company.idm.example.com/Example/  
   curl: (56) OpenSSL SSL_read: error:1409445C:SSL routines:ssl3_read_bytes:tlsv13 **alert certificate required**, errno 0
   
   The error indicates that the **my_company.idm.example.com** web server requires a client certificate authentication.
   ```

2. Pass the client private key and certificate, as well as the CA certificate to **curl** to access the same URL with client authentication:

   ```bash
   $ curl --cacert ca.crt --key client.key --cert client.crt https://my_company.idm.example.com/Example/  
   
   If the request succeeds, **curl** displays the **index.html** file stored in the **/var/www/html/Example/** directory.
   ```

**Additional resources**

- For further details about client authentication, see the [mod_ssl Configuration How-To documentation](#) in the Apache manual. For details about installing the manual, see [Installing the Apache HTTP Server manual](#).

### 62.11. REQUESTING A NEW USER CERTIFICATE AND EXPORTING IT TO THE CLIENT

As an Identity Management (IdM) administrator, you can configure a web server running on an IdM client to request users that use web browsers to access the server to authenticate with certificates issued by a specific IdM sub-CA. Complete this section to request a user certificate from a specific IdM sub-CA and to export the certificate and the corresponding private key on to the host from which the user wants to access the web server using a web browser. Afterwards, import the certificate and the private key into the browser.

**Procedure**
1. Optionally, create a new directory, for example `~/certdb/`, and make it a temporary certificate database. When asked, create an NSS Certificate DB password to encrypt the keys to the certificate to be generated in a subsequent step:

```
# mkdir ~/certdb/
# certutil -N -d ~/certdb/
```

Enter a password which will be used to encrypt your keys. The password should be at least 8 characters long, and should contain at least one non-alphabetic character.

Enter new password:
Re-enter password:

2. Create the certificate signing request (CSR) and redirect the output to a file. For example, to create a CSR with the name `certificate_request.csr` for a 4096 bit certificate for the `idm_user` user in the `IDM.EXAMPLE.COM` realm, setting the nickname of the certificate private keys to `idm_user` for easy findability, and setting the subject to `CN=idm_user,O=IDM.EXAMPLE.COM`:

```
# certutil -R -d ~/certdb/ -a -g 4096 -n idm_user -s "CN=idm_user,O=IDM.EXAMPLE.COM"
```

3. When prompted, enter the same password that you entered when using `certutil` to create the temporary database. Then continue typing randomly until told to stop:

```
Enter Password or Pin for "NSS Certificate DB":

A random seed must be generated that will be used in the creation of your key. One of the easiest ways to create a random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter is full. DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!

Continue typing until the progress meter is full:
```

4. Submit the certificate request file to the server. Specify the Kerberos principal to associate with the newly-issued certificate, the output file to store the certificate, and optionally the certificate profile. Specify the IdM sub-CA that you want to issue the certificate. For example, to obtain a certificate of the `IECUserRoles` profile, a profile with added user roles extension, for the `idm_user@IDM.EXAMPLE.COM` principal from `webclient-ca`, and save the certificate in the `~/idm_user.pem` file:

```
# ipa cert-request certificate_request.csr --principal=idm_user@IDM.EXAMPLE.COM --profile-id=IECUserRoles --ca=webclient-ca --certificate-out=~/.idm_user.pem
```

5. Add the certificate to the NSS database. Use the `-n` option to set the same nickname that you used when creating the CSR previously so that the certificate matches the private key in the NSS database. The `-t` option sets the trust level. For details, see the `certutil(1)` man page. The `-i` option specifies the input certificate file. For example, to add to the NSS database a certificate with the `idm_user` nickname that is stored in the `~/idm_user.pem` file in the `~/certdb/` database:
6. Verify that the key in the NSS database does not show (orphan) as its nickname. For example, to verify that the certificate stored in the ~/certdb/ database is not orphaned:

```bash
# certutil -A -d ~/certdb/ -n idm_user -t "P," -i ~/idm_user.pem
```

7. Use the `pk12util` command to export the certificate from the NSS database to the PKCS12 format. For example, to export the certificate with the `idm_user` nickname from the /root/certdb NSS database into the ~/idm_user.p12 file:

```bash
# pk12util -d ~/certdb -o ~/idm_user.p12 -n idm_user
```

8. Transfer the certificate to the host on which you want the certificate authentication for `idm_user` to be enabled:

```bash
# scp ~/idm_user.p12 idm_user@client.idm.example.com:/home/idm_user/
```

9. On the host to which the certificate has been transferred, make the directory in which the .pkcs12 file is stored inaccessible to the 'other' group for security reasons:

```bash
# chmod o-rwx /home/idm_user/
```

10. For security reasons, remove the temporary NSS database and the .pkcs12 file from the server:

```bash
# rm ~/certdb/
# rm ~/idm_user.p12
```

### 62.12. CONFIGURING A BROWSER TO ENABLE CERTIFICATE AUTHENTICATION

To be able to authenticate with a certificate when using the WebUI to log into Identity Management (IdM), you need to import the user and the relevant certificate authority (CA) certificates into the Mozilla Firefox or Google Chrome browser. The host itself on which the browser is running does not have to be part of the IdM domain.

IdM supports the following browsers for connecting to the WebUI:

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

The following procedure shows how to configure the Mozilla Firefox 57.0.1 browser.

**Prerequisites**
- You have the **user certificate** that you want to import to the browser at your disposal in the PKCS#12 format.

- You have downloaded the **sub-CA certificate** and have it at your disposal in the PEM format.

**Procedure**

1. Open Firefox, then navigate to **Preferences → Privacy & Security**.

   ![Figure 62.8. Privacy and Security section in Preferences](image)

2. Click **View Certificates**.

   ![Figure 62.9. View Certificates in Privacy and Security](image)

3. In the **Your Certificates** tab, click **Import**. Locate and open the certificate of the user in the PKCS12 format, then click **OK** and **OK**.

4. To make sure that your IdM sub-CA is recognized by Firefox as a trusted authority, import the IdM sub-CA certificate that you saved in **Section 62.2, "Downloading the sub-CA certificate from IdM WebUI"** as a trusted certificate authority certificate:
   a. Open Firefox, navigate to Preferences and click **Privacy & Security**.
b. Click **View Certificates**.

![Figure 62.11. View Certificates in Privacy and Security](image)

Figure 62.11. View Certificates in Privacy and Security

- In the **Authorities** tab, click **Import**. Locate and open the sub-CA certificate. Trust the certificate to identify websites, then click **OK** and **OK**.
CHAPTER 63. INVALIDATING A SPECIFIC GROUP OF RELATED CERTIFICATES QUICKLY

As a system administrator, if you want to be able to invalidate a specific group of related certificates quickly:

- Design your applications so that they only trust certificates that were issued by a specific lightweight Identity Management (IdM) sub-CA. Afterwards, you will be able to invalidate all these certificates by only revoking the certificate of the Identity Management (IdM) sub-CA that issued these certificates. For details on how to create and use a lightweight sub-CA in IdM, see Invalidating a specific group of related certificates quickly.

- To ensure that all the certificates that have been issued by the to-be-revoked IdM sub-CA are immediately invalid, configure applications that rely on such certificates to use the IdM OCSP responders. For example, to configure the Firefox browser to use OCSP responders, make sure that the **Query OCSP responder servers to confirm the current validity of certificates** checkbox is checked in Firefox Preferences.

In IdM, the certificate revocation list (CRL) is updated every four hours.

To invalidate all the certificates issued by an IdM sub-CA, **revoke the IdM sub-CA certificate**. In addition, **disable the relevant CA ACLs**, and consider **disabling the IdM sub-CA**. Disabling the sub-CA prevents the sub-CA from issuing new certificates, but allows Online Certificate Status Protocol (OCSP) responses to be produced for previously issued certificates because the sub-CA’s signing keys are retained.

**IMPORTANT**

Do not delete the sub-CA if you use OCSP in your environment. Deleting the sub-CA deletes the signing keys of the sub-CA, preventing production of OCSP responses for certificates issued by that sub-CA.

The only scenario when deleting a sub-CA is preferable to disabling it is when you want to create a new sub-CA with the same Subject distinguished name (DN) but a new signing key.

63.1. DISABLING CA ACLS IN IDM CLI

When you want to retire an IdM service or a group of IdM services, consider disabling any existing corresponding CA ACLs.

Complete this section to disable the **TLS_web_server_authentication** CA ACL that restricts the web server running on your IdM client to request a certificate to be issued by the **webserver-ca** IdM sub-CA, and to disable the **TLS_web_client_authentication** CA ACL that restricts IdM users to request a user certificate to be issued by the **webclient-ca** IdM sub-CA.

**Procedure**

1. Optionally, to view all the CA ACLs in your IdM environment, enter the `ipa caacl-find` command:

   ```
   $ ipa caacl-find
   -----------------  
   3 CA ACLs matched
   -----------------
   ACL name: hosts_services_calIPAserviceCert
   ```
2. Optionally, to view the details of a CA ACL, enter the `ipa caacl-show` command, and specify the CA ACL name:

```bash
$ ipa caacl-show TLS_web_server_authentication
```

```
ACL name: TLS_web_server_authentication
Description: CAACL for web servers authenticating to web clients using certificates issued by webserver-ca
Enabled: TRUE
CAs: webserver-ca
Profiles: calIPAserviceCert
Services: HTTP/rhel8server.idm.example.com@IDM.EXAMPLE.COM
```

3. To disable a CA ACL, enter the `ipa caacl-disable` command, and specify the CA ACL name.

- To disable the `TLS_web_server_authentication` CA ACL, enter:

```bash
$ ipa caacl-disable TLS_web_server_authentication
```

```
Disabled CA ACL "TLS_web_server_authentication"
```

- To disable the `TLS_web_client_authentication` CA ACL, enter:

```bash
$ ipa caacl-disable TLS_web_client_authentication
```

```
Disabled CA ACL "TLS_web_client_authentication"
```

The only enabled CA ACL now is the `hosts_services_calIPAserviceCert` CA ACL.

**IMPORTANT**

Be extremely careful about disabling the `hosts_services_calIPAserviceCert` CA ACL. Disabling `hosts_services_calIPAserviceCert`, without another CA ACL granting IdM servers use of the `ipa` CA with the `calIPAserviceCert` profile means that certificate renewal of the IdM HTTP and LDAP certificates will fail. The expired IdM HTTP and LDAP certificates will eventually cause IdM system failure.

### 63.2. DISABLING AN IDM SUB-CA

After revoking the CA certificate of an IdM sub-CA in order to invalidate all the certificates issued by that sub-CA, consider disabling the IdM sub-CA if you no longer need it. You can re-enable the sub-CA at a later time.
Disabling the sub-CA prevents the sub-CA from issuing new certificates, but allows Online Certificate Status Protocol (OCSP) responses to be produced for previously issued certificates because the sub-CA's signing keys are retained.

**Prerequisites**

- You are logged in as IdM administrator.

**Procedure**

- Enter the `ipa ca-disable` command and specify the name of the sub-CA:

```bash
$ ipa ca-disable webserver-CA
---------------------
Disabled CA "webserver-CA"
---------------------
```
CHAPTER 64. VAULTS IN IDM

This chapter describes vaults in Identity Management (IdM). It introduces the following topics:

- The concept of the vault.
- The different roles associated with a vault.
- The different types of vaults available in IdM based on the level of security and access control.
- The different types of vaults available in IdM based on ownership.
- The concept of vault containers.
- The basic commands for managing vaults in IdM.
- Installing the key recovery authority (KRA), which is a prerequisite for using vaults in IdM.

64.1. VAULTS AND THEIR BENEFITS

A vault is a useful feature for those Identity Management (IdM) users who want to keep all their sensitive data stored securely but conveniently in one place. This section explains the various types of vaults and their uses, and which vault you should choose based on your requirements.

A vault is a secure location in (IdM) for storing, retrieving, sharing, and recovering a secret. A secret is security-sensitive data, usually authentication credentials, that only a limited group of people or entities can access. For example, secrets include:

- passwords
- PINs
- private SSH keys

A vault is comparable to a password manager. Just like a password manager, a vault typically requires a user to generate and remember one primary password to unlock and access any information stored in the vault. However, a user can also decide to have a standard vault. A standard vault does not require the user to enter any password to access the secrets stored in the vault.

NOTE

The purpose of vaults in IdM is to store authentication credentials that allow you to authenticate to external, non-IdM-related services.

Other important characteristics of the IdM vaults are:

- Vaults are only accessible to the vault owner and those IdM users that the vault owner selects to be the vault members. In addition, the IdM administrator has access to the vault.
- If a user does not have sufficient privileges to create a vault, an IdM administrator can create the vault and set the user as its owner.
- Users and services can access the secrets stored in a vault from any machine enrolled in the IdM domain.
One vault can only contain one secret, for example, one file. However, the file itself can contain multiple secrets such as passwords, keytabs or certificates.

NOTE

Vault is only available from the IdM command line (CLI), not from the IdM Web UI.

64.2. VAULT OWNERS, MEMBERS, AND ADMINISTRATORS

Identity Management (IdM) distinguishes the following vault user types:

Vault owner

A vault owner is a user or service with basic management privileges on the vault. For example, a vault owner can modify the properties of the vault or add new vault members. Each vault must have at least one owner. A vault can also have multiple owners.

Vault member

A vault member is a user or service that can access a vault created by another user or service.

Vault administrator

Vault administrators have unrestricted access to all vaults and are allowed to perform all vault operations.

NOTE

Symmetric and asymmetric vaults are protected with a password or key and apply special access control rules (see Vault types). The administrator must meet these rules to:

- Access secrets in symmetric and asymmetric vaults.
- Change or reset the vault password or key.

A vault administrator is any user with the Vault Administrators privilege. In the context of the role-based access control (RBAC) in IdM, a privilege is a group of permissions that you can apply to a role.

Vault User

The vault user represents the user in whose container the vault is located. The Vault user information is displayed in the output of specific commands, such as `ipa vault-show`:

```bash
$ ipa vault-show my_vault
Vault name: my_vault
Type: standard
Owner users: user
Vault user: user
```

For details on vault containers and user vaults, see Vault containers.

Additional resources

- Certain owner and member privileges depend on the type of the vault. See Standard, symmetric and asymmetric vaults for details.
64.3. STANDARD, SYMMETRIC, AND ASYMMETRIC VAULTS

Based on the level of security and access control, IdM classifies vaults into the following types:

**Standard vaults**

Vault owners and vault members can archive and retrieve the secrets without having to use a password or key.

**Symmetric vaults**

Secrets in the vault are protected with a symmetric key. Vault owners and members can archive and retrieve the secrets, but they must provide the vault password.

**Asymmetric vaults**

Secrets in the vault are protected with an asymmetric key. Users archive the secret using a public key and retrieve it using a private key. Vault members can only archive secrets, while vault owners can do both, archive and retrieve secrets.

64.4. USER, SERVICE, AND SHARED VAULTS

Based on ownership, IdM classifies vaults into several types. The table below contains information about each type, its owner and use.

Table 64.1. IdM vaults based on ownership

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Owner</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>User vault</td>
<td>A private vault for a user</td>
<td>A single user</td>
<td>Any user can own one or more user vaults if allowed by IdM administrator</td>
</tr>
<tr>
<td>Service vault</td>
<td>A private vault for a service</td>
<td>A single service</td>
<td>Any service can own one or more user vaults if allowed by IdM administrator</td>
</tr>
<tr>
<td>Shared vault</td>
<td>A vault shared by multiple users and services</td>
<td>The vault administrator who created the vault</td>
<td>Users and services can own one or more user vaults if allowed by IdM administrator. The vault administrators other than the one that created the vault also have full access to the vault.</td>
</tr>
</tbody>
</table>

64.5. VAULT CONTAINERS

A vault container is a collection of vaults. The table below lists the default vault containers that Identity Management (IdM) provides.

Table 64.2. Default vault containers in IdM

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>User container</td>
<td>A private container for a user</td>
<td>Stores user vaults for a particular user</td>
</tr>
<tr>
<td>Service container</td>
<td>A private container for a service</td>
<td>Stores service vaults for a particular service</td>
</tr>
</tbody>
</table>
IdM creates user and service containers for each user or service automatically when the first private vault for the user or service is created. After the user or service is deleted, IdM removes the container and its contents.

### 64.6. BASIC IDM VAULT COMMANDS

This section describes basic commands you can use to manage Identity Management (IdM) vaults. The table below contains a list of `ipa vault-*` commands with the explanation of their purpose.

**NOTE**

Before running any `ipa vault-*` command, install the Key Recovery Authority (KRA) certificate system component on one or more of the servers in your IdM domain. For details, see [Installing the Key Recovery Authority in IdM](#).

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ipa help vault</code></td>
<td>Displays conceptual information about IdM vaults and sample vault commands.</td>
</tr>
<tr>
<td><code>ipa vault-add --help</code>, <code>ipa vault-find --help</code></td>
<td>Adding the <code>--help</code> option to a specific <code>ipa vault-*</code> command displays the options and detailed help available for that command.</td>
</tr>
<tr>
<td><code>ipa vault-show user_vault --user idm_user</code></td>
<td>When accessing a vault as a vault member, you must specify the vault owner. If you do not specify the vault owner, IdM informs you that it did not find the vault:</td>
</tr>
<tr>
<td><code>ipa vault-show shared_vault --shared</code></td>
<td>When accessing a shared vault, you must specify that the vault you want to access is a shared vault. Otherwise, IdM informs you it did not find the vault:</td>
</tr>
</tbody>
</table>

**64.7. INSTALLING THE KEY RECOVERY AUTHORITY IN IDM**

This section describes how you can enable vaults in Identity Management (IdM) by installing the Key Recovery Authority (KRA) Certificate System (CS) component.

**Prerequisites**

- You are logged in as IdM administrator.
You are logged in as root on an IdM client.

Procedure

- Install the KRA:

  # ipa-kra-install

IMPORTANT

You can install the first KRA of an IdM cluster on a hidden replica. However, installing additional KRAs requires temporarily activating the hidden replica before you install the KRA clone on a non-hidden replica. Then you can hide the originally hidden replica again.

NOTE

To make the vault service highly available, install the KRA on two IdM servers or more.

Additional resources

- For more information on how to activate an IdM replica and how to hide it, see Demoting or promoting hidden replicas.

- For more information on hidden replicas in IdM, see The hidden replica mode.
CHAPTER 65. USING IDM USER VAULTS: STORING AND RETRIEVING SECRETS

This chapter describes how to use user vaults in Identity Management. Specifically, it describes how a user can store a secret in an IdM vault, and how the user can retrieve it. The user can do the storing and the retrieving from two different IdM clients.

Prerequisites

- The Key Recovery Authority (KRA) Certificate System component has been installed on one or more of the servers in your IdM domain. For details, see Installing the Key Recovery Authority in IdM.

65.1. STORING A SECRET IN A USER VAULT

This section shows how a user can create a vault container with one or more private vaults to securely store files with sensitive information. In the example used in the procedure below, the idm_user user creates a vault of the standard type. The standard vault type ensures that idm_user will not be required to authenticate when accessing the file. idm_user will be able to retrieve the file from any IdM client to which the user is logged in.

In the procedure:

- idm_user is the user who wants to create the vault.
- my_vault is the vault used to store the user’s certificate.
- The vault type is standard, so that accessing the archived certificate does not require the user to provide a vault password.
- secret.txt is the file containing the certificate that the user wants to store in the vault.

Prerequisites

- You know the password of idm_user.
- You are logged in to a host that is an IdM client.

Procedure

1. Obtain the Kerberos ticket granting ticket (TGT) for idm_user:

   $ kinit idm_user

2. Use the ipa vault-add command with the --type standard option to create a standard vault:

   $ ipa vault-add my_vault --type standard

   ----------------------
   Added vault "my_vault"
   ----------------------
   Vault name: my_vault
   Type: standard
   Owner users: idm_user
   Vault user: idm_user
IMPORTANT

Make sure the first user vault for a user is created by the same user. Creating the first vault for a user also creates the user's vault container. The agent of the creation becomes the owner of the vault container.

For example, if another user, such as admin, creates the first user vault for user1, the owner of the user's vault container will also be admin, and user1 will be unable to access the user vault or create new user vaults.

3. Use the `ipa vault-archive` command with the `--in` option to archive the `secret.txt` file into the vault:

```
$ ipa vault-archive my_vault --in secret.txt
-----------------------------------
Archived data into vault "my_vault"
-----------------------------------
```

65.2. RETRIEVING A SECRET FROM A USER VAULT

As an Identity Management (IdM), you can retrieve a secret from your user private vault onto any IdM client to which you are logged in.

This section shows how to retrieve, as an IdM user named `idm_user`, a secret from the user private vault named `my_vault` onto `idm_client.idm.example.com`.

Prerequisites

- `idm_user` is the owner of `my_vault`.
- `idm_user` has archived a secret in the vault.
- `my_vault` is a standard vault, which means that `idm_user` does not have to enter any password to access the contents of the vault.

Procedure

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

```
$ ssh idm_user@idm_client.idm.example.com
```

2. Log in as `idm_user`:

```
$ kinit user
```

3. Use the `ipa vault-retrieve --out` command with the `--out` option to retrieve the contents of the vault and save them into the `secret_exported.txt` file.

```
$ ipa vault-retrieve my_vault --out secret_exported.txt
--------------------------------------
Retrieved data from vault "my_vault"
--------------------------------------
```
Additional resources

- You can use Ansible to automate the process of managing IdM user vaults. For more information, see Using Ansible to manage IdM user vaults: storing and retrieving secrets.
CHAPTER 66. USING ANSIBLE TO MANAGE IDM USER VAULTS: STORING AND RETRIEVING SECRETS

This chapter describes how to manage user vaults in Identity Management using the Ansible vault module. Specifically, it describes how a user can use Ansible playbooks to perform the following three consecutive actions:

- Create an user vault in IdM.
- Store a secret in the vault.
- Retrieve a secret from the vault.

The user can do the storing and the retrieving from two different IdM clients.

Prerequisites

- The Key Recovery Authority (KRA) Certificate System component has been installed on one or more of the servers in your IdM domain. For details, see Installing the Key Recovery Authority in IdM.

66.1. ENSURING THE PRESENCE OF A STANDARD USER VAULT IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) user can use an Ansible playbook to create a vault container with one or more private vaults to securely store sensitive information. In the example used in the procedure below, the idm_user user creates a vault of the standard type named my_vault. The standard vault type ensures that idm_user will not be required to authenticate when accessing the file. idm_user will be able to retrieve the file from any IdM client to which the user is logged in.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller, that is the host on which you execute the steps in the procedure.
- You know the password of idm_user.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/vault directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/vault

2. Create an inventory file, for example inventory.file:

   $ touch inventory.file

3. Open inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

   [ipaserver]
   server.idm.example.com
4. Make a copy of the `ensure-standard-vault-is-present.yml` Ansible playbook file. For example:

```
$ cp ensure-standard-vault-is-present.yml ensure-standard-vault-is-present-copy.yml
```

5. Open the `ensure-standard-vault-is-present-copy.yml` file for editing.

6. Adapt the file by setting the following variables in the `ipavault` task section:
   - Set the `ipaadmin_principal` variable to `idm_user`.
   - Set the `ipaadmin_password` variable to the password of `idm_user`.
   - Set the `user` variable to `idm_user`.
   - Set the `name` variable to `my_vault`.
   - Set the `vault_type` variable to `standard`.

This the modified Ansible playbook file for the current example:

```
---
- name: Tests
  hosts: ipaserver
  become: true
  gather_facts: false
  tasks:
    - ipavault:
        ipaadmin_principal: idm_user
        ipaadmin_password: idm_user_password
        user: idm_user
        name: my_vault
        vault_type: standard
```

7. Save the file.

8. Run the playbook:

```
$ ansible-playbook -v -i inventory.file ensure-standard-vault-is-present-copy.yml
```

### 66.2. ARCHIVING A SECRET IN A STANDARD USER VAULT IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) user can use an Ansible playbook to store sensitive information in a personal vault. In the example used, the `idm_user` user archives a file with sensitive information named `password.txt` in a vault named `my_vault`.

**Prerequisites**

- You have installed the `ansible-freeipa` package on the Ansible controller, that is the host on which you execute the steps in the procedure.
- You know the password of `idm_user`.
- `idm_user` is the owner, or at least a member user of `my_vault`.
You have access to `password.txt`, the secret that you want to archive in `my_vault`.

### Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/vault` directory:
   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/vault
   ```
2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:
   ```plaintext
   [ipaserver]
   server.idm.example.com
   ```
3. Make a copy of the `data-archive-in-symmetric-vault.yml` Ansible playbook file but replace "symmetric" by "standard". For example:
   ```bash
   ```
4. Open the `data-archive-in-standard-vault-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipavault` task section:
   - Set the `ipaadmin_principal` variable to `idm_user`.
   - Set the `ipaadmin_password` variable to the password of `idm_user`.
   - Set the `user` variable to `idm_user`.
   - Set the `name` variable to `my_vault`.
   - Set the `in` variable to the full path to the file with sensitive information.
   - Set the `action` variable to `member`.
   This the modified Ansible playbook file for the current example:
   ```yaml
   ---
   - name: Tests
     hosts: ipaserver
     become: true
     gather_facts: false
     tasks:
     - ipavault:
         ipaadmin_principal: idm_user
         ipaadmin_password: idm_user_password
         user: idm_user
         name: my_vault
         in: /usr/share/doc/ansible-freeipa/playbooks/vault/password.txt
         action: member
   ```
6. Save the file.
7. Run the playbook:
ansible-playbook -v -i inventory.file data-archive-in-standard-vault-copy.yml

66.3. RETRIEVING A SECRET FROM A STANDARD USER VAULT IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) user can use an Ansible playbook to retrieve a secret from the user personal vault. In the example used in the procedure below, the idm_user user retrieves a file with sensitive data from a vault of the standard type named my_vault onto an IdM client named host01. idm_user does not have to authenticate when accessing the file. idm_user can use Ansible to retrieve the file from any IdM client on which Ansible is installed.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the password of idm_user.
- idm_user is the owner of my_vault.
- idm_user has stored a secret in my_vault.
- Ansible can write into the directory on the IdM host into which you want to retrieve the secret.
- idm_user can read from the directory on the IdM host into which you want to retrieve the secret.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/vault directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/vault

2. Open your inventory file and mention, in a clearly defined section, the IdM client onto which you want to retrieve the secret. For example, to instruct Ansible to retrieve the secret onto host01.idm.example.com, enter:

   [ipahost]
   host01.idm.example.com

3. Make a copy of the retrieve-data-symmetric-vault.yml Ansible playbook file. Replace "symmetric" with "standard". For example:

   $ cp retrieve-data-symmetric-vault.yml retrieve-data-standard-vault.yml-copy.yml

4. Open the retrieve-data-standard-vault.yml-copy.yml file for editing.

5. Adapt the file by setting the hosts variable to ipahost.

6. Adapt the file by setting the following variables in the ipavault task section:
   - Set the ipaadmin_principal variable to idm_user.
   - Set the ipaadmin_password variable to the password of idm_user.
- Set the **user** variable to **idm_user**.
- Set the **name** variable to **my_vault**.
- Set the **out** variable to the full path of the file into which you want to export the secret.
- Set the **state** variable to **retrieved**.

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

```yaml
---
- name: Tests
  hosts: ipahost
  become: true
  gather_facts: false
  tasks:
    - ipavault:
        ipaadmin_principal: idm_user
        ipaadmin_password: idm_user_password
        user: idm_user
        name: my_vault
        out: /tmp/password_exported.txt
        state: retrieved
```

7. Save the file.

8. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file retrieve-data-standard-vault.yml-copy.yml
   ```

**Verification steps**

1. **SSH** to **host01** as **user01**:

   ```bash
   $ ssh user01@host01.idm.example.com
   ```

2. View the file specified by the **out** variable in the Ansible playbook file:

   ```bash
   $ vim /tmp/password_exported.txt
   ```

You can now see the exported secret.

- For more information about using Ansible to manage IdM vaults and user secrets and about playbook variables, see the README-vault.md Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory and the sample playbooks available in the `/usr/share/doc/ansible-freeipa/playbooks/vault/` directory.
CHAPTER 67. MANAGING IDM SERVICE SECRETS: STORING AND RETRIEVING SECRETS

This section shows how an administrator can use the `ansible-freeipa vault` module to securely store a service secret in a centralized location. The vault used in the example is asymmetric, which means that in order to use it, the administrator needs to perform the following steps:

1. Generate a private key using, for example, the `openssl` utility.
2. Generate a public key based on the private key.

The service secret is encrypted with the public key when an administrator archives it into the vault. Afterwards, a service instance hosted on a specific machine in the domain retrieves the secret using the private key. Only the service and the administrator are allowed to access the secret.

If the secret is compromised, the administrator can replace it in the service vault and then redistribute it to those individual service instances that have not been compromised.

Prerequisites

- The Key Recovery Authority (KRA) Certificate System component has been installed on one or more of the servers in your IdM domain. For details, see Installing the Key Recovery Authority in IdM.

This section includes these procedure

1. Storing an IdM service secret in an asymmetric vault
2. Retrieving a service secret for an IdM service instance
3. Changing an IdM service vault secret when compromised

Terminology used

In the procedures:

- `admin` is the administrator who manages the service password.
- `private-key-to-an-externally-signed-certificate.pem` is the file containing the service secret, in this case a private key to an externally signed certificate. Do not confuse this private key with the private key used to retrieve the secret from the vault.
- `secret_vault` is the vault created for the service.
- `HTTP/webserver.idm.example.com` is the service whose secret is being archived.
- `service-public.pem` is the service public key used to encrypt the password stored in `password_vault`.
- `service-private.pem` is the service private key used to decrypt the password stored in `secret_vault`.

67.1. STORING AN IDM SERVICE SECRET IN AN ASYMMETRIC VAULT

This section describes how to create an asymmetric vault and use it to archive a service secret.
Prerequisites

- You know the IdM administrator password.

Procedure

1. Log in as the administrator:

   ```
   $ kinit admin
   ```

2. Obtain the public key of the service instance. For example, using the `openssl` utility:

   a. Generate the `service-private.pem` private key.

   ```
   $ openssl genrsa -out service-private.pem 2048
   Generating RSA private key, 2048 bit long modulus
   ...........................................+++ e is 65537 (0x10001)
   ```

   b. Generate the `service-public.pem` public key based on the private key.

   ```
   $ openssl rsa -in service-private.pem -out service-public.pem -pubout
   writing RSA key
   ```

3. Create an asymmetric vault as the service instance vault, and provide the public key:

   ```
   $ ipa vault-add secret_vault --service HTTP/webserver.idm.example.com --type asymmetric --public-key-file service-public.pem
   -------------------
   Added vault "secret_vault"
   -------------------
   Vault name: secret_vault
   Type: asymmetric
   Public key: LS0tLS1C...S0tLS0tCg==
   Owner users: admin
   Vault service: HTTP/webserver.idm.example.com@IDM.EXAMPLE.COM
   ```

   The password archived into the vault will be protected with the key.

4. Archive the service secret into the service vault:

   ```
   $ ipa vault-archive secret_vault --service HTTP/webserver.idm.example.com --in private-key-to-an-externally-signed-certificate.pem
   -------------------
   Archived data into vault "secret_vault"
   -------------------
   ```

   This encrypts the secret with the service instance public key.

Repeat these steps for every service instance that requires the secret. Create a new asymmetric vault for each service instance.
67.2. RETRIEVING A SERVICE SECRET FOR AN IDM SERVICE INSTANCE

This section describes how a service instance can retrieve the service vault secret using a locally-stored service private key.

Prerequisites

- You have access to the keytab of the service principal owning the service vault, for example HTTP/webserver.idm.example.com.
- You have created an asymmetric vault and archived a secret in the vault.
- You have access to the private key used to retrieve the service vault secret.

Procedure

1. Log in as the administrator:
   
   $ kinit admin

2. Obtain a Kerberos ticket for the service:
   
   # kinit HTTP/webserver.idm.example.com -k -t /etc/httpd/conf/ipa.keytab

3. Retrieve the service vault password:
   
   $ ipa vault-retrieve secret_vault --service HTTP/webserver.idm.example.com --private-key-file service-private.pem --out secret.txt
   
   Retrieved data from vault "secret_vault"

67.3. CHANGING AN IDM SERVICE VAULT SECRET WHEN COMPROMISED

This section describes how to isolate a compromised service instance by changing the service vault secret.

Prerequisites

- You know the IdM administrator password.
- You have created an asymmetric vault to store the service secret.
- You have generated the new secret and have access to it, for example in the new-private-key-to-an-externally-signed-certificate.pem file.

Procedure

1. Archive the new secret into the service instance vault:
   
   $ ipa vault-archive secret_vault --service HTTP/webserver.idm.example.com --in new-
private-key-to-an-externally-signed-certificate.pem
-----------------------------------
Archived data into vault "secret_vault"
-----------------------------------
This overwrites the current secret stored in the vault.

2. Retrieve the new secret on non-compromised service instances only. For details, see Retrieving a service secret for an IdM service instance.

Additional resources

- You can use Ansible to automate the process of managing IdM service vaults. For more information, see Using Ansible to manage IdM service vaults: storing and retrieving secrets.
CHAPTER 68. USING ANSIBLE TO MANAGE IDM SERVICE VAULTS: STORING AND RETRIEVING SECRETS

This section shows how an administrator can use the ansible-freeipa vault module to securely store a service secret in a centralized location. The vault used in the example is asymmetric, which means that in order to use it, the administrator needs to perform the following steps:

1. Generate a private key using, for example, the openssl utility.

2. Generate a public key based on the private key.

The service secret is encrypted with the public key when an administrator archives it into the vault. Afterwards, a service instance hosted on a specific machine in the domain retrieves the secret using the private key. Only the service and the administrator are allowed to access the secret.

If the secret is compromised, the administrator can replace it in the service vault and then redistribute it to those individual service instances that have not been compromised.

Prerequisites

- The Key Recovery Authority (KRA) Certificate System component has been installed on one or more of the servers in your IdM domain. For details, see Installing the Key Recovery Authority in IdM.

This section includes these procedures:

- Ensuring the presence of an asymmetric service vault in IdM using Ansible
- Storing an IdM service secret in an asymmetric vault using Ansible
- Retrieving a service secret for an IdM service using Ansible
- Changing an IdM service vault secret when compromised using Ansible

In the procedures:

- admin is the administrator who manages the service password.

- private-key-to-an-externally-signed-certificate.pem is the file containing the service secret, in this case a private key to an externally signed certificate. Do not confuse this private key with the private key used to retrieve the secret from the vault.

- secret_vault is the vault created to store the service secret.

- HTTP/webserver1.idm.example.com is the service that is the owner of the vault.

- HTTP/webserver2.idm.example.com and HTTP/webserver3.idm.example.com are the vault member services.

- service-public.pem is the service public key used to encrypt the password stored in password_vault.

- service-private.pem is the service private key used to decrypt the password stored in secret_vault.
68.1. ENSURING THE PRESENCE OF AN ASYMMETRIC SERVICE VAULT IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to create a service vault container with one or more private vaults to securely store sensitive information. In the example used in the procedure below, the administrator creates an asymmetric vault named `secret_vault`. This ensures that the vault members have to authenticate using a private key in order to retrieve the secret in the vault. The vault members will be able to retrieve the file from any IdM client.

Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/vault` directory:

   ```bash
   cd /usr/share/doc/ansible-freeipa/playbooks/vault
   ``

2. Obtain the public key of the service instance. For example, using the `openssl` utility:

   a. Generate the `service-private.pem` private key.

      ```bash
      openssl genrsa -out service-private.pem 2048
      Generating RSA private key, 2048 bit long modulus
                        ...........................................+++e is 65537 (0x10001)
      ```

   b. Generate the `service-public.pem` public key based on the private key.

      ```bash
      openssl rsa -in service-private.pem -out service-public.pem -pubout
      writing RSA key
      ```

3. Optional: Create an inventory file if it does not exist, for example `inventory.file`:

   ```bash
   touch inventory.file
   ``

4. Open your inventory file and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

5. Make a copy of the `ensure-asymmetric-vault-is-present.yml` Ansible playbook file. For example:

   ```bash
   cp ensure-asymmetric-vault-is-present.yml ensure-asymmetric-service-vault-is-present-copy.yml
   ```
6. Open the `ensure-asymmetric-vault-is-present-copy.yml` file for editing.

7. Add a task that copies the `service-public.pem` public key from the Ansible controller to the `server.idm.example.com` server.

8. Modify the rest of the file by setting the following variables in the `ipavault` task section:
   - Set the `ipaadmin_password` variable to the IdM administrator password.
   - Define the name of the vault using the `name` variable, for example `secret_vault`.
   - Set the `vault_type` variable to `asymmetric`.
   - Set the `service` variable to the principal of the service that owns the vault, for example `HTTP/webserver1.idm.example.com`.
   - Set the `public_key_file` to the location of your public key.

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

   ```yaml
   ---
   - name: Tests
     hosts: ipaserver
     become: true
     gather_facts: false
     tasks:
       - name: Copy public key to ipaserver.
         copy:
           src: /path/to/service-public.pem
           dest: /usr/share/doc/ansible-freeipa/playbooks/vault/service-public.pem
           mode: 0600
       - name: Add data to vault, from a LOCAL file.
         ipavault:
           ipaadmin_password: Secret123
           name: secret_vault
           vault_type: asymmetric
           service: HTTP/webserver1.idm.example.com
   ```

9. Save the file.

10. Run the playbook:

    ```bash
    $ ansible-playbook -v -i inventory.file ensure-asymmetric-service-vault-is-present-copy.yml
    ```

68.2. ADDING MEMBER SERVICES TO AN ASYMMETRIC VAULT USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to add member services to a service vault so that they can all retrieve the secret stored in the vault. In the example used in the procedure below, the IdM administrator adds the `HTTP/webserver2.idm.example.com` and `HTTP/webserver3.idm.example.com` service principals to the `secret_vault` vault that is owned by `HTTP/webserver1.idm.example.com`. 
Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the **IdM administrator** password.
- You have created an asymmetric vault to store the service secret.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/vault` directory:

   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/vault
   ```

2. Optional: Create an inventory file if it does not exist, for example `inventory.file`:

   ```bash
   $ touch inventory.file
   ```

3. Open your inventory file and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

4. Make a copy of the `data-archive-in-asymmetric-vault.yml` Ansible playbook file. For example:

   ```bash
   $ cp data-archive-in-asymmetric-vault.yml add-services-to-an-asymmetric-vault.yml
   ```

5. Open the `data-archive-in-asymmetric-vault-copy.yml` file for editing.

6. Modify the file by setting the following variables in the `ipavault` task section:

   - Set the `ipaadmin_password` variable to the IdM administrator password.
   - Set the `name` variable to the name of the vault, for example `secret_vault`.
   - Set the `service` variable to the service owner of the vault, for example `HTTP/webserver1.idm.example.com`.
   - Define the services that you want to have access to the vault secret using the `services` variable.
   - Set the `action` variable to `member`.

   This the modified Ansible playbook file for the current example:

   ```
   ---
   - name: Tests
     hosts: ipaserver
     become: true
     gather_facts: false
     
     tasks:
     - ipavault:
       ipaadmin_password: Secret123
   ```
name: secret_vault
service: HTTP/webserver1.idm.example.com
services:
- HTTP/webserver2.idm.example.com
- HTTP/webserver3.idm.example.com
action: member

7. Save the file.

8. Run the playbook:

```
$ ansible-playbook -v -i inventory.file add-services-to-an-asymmetric-vault.yml
```

### 68.3. STORING AN IDM SERVICE SECRET IN AN ASYMMETRIC VAULT USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to store a secret in a service vault so that it can be later retrieved by the service. In the example used in the procedure below, the administrator stores a **PEM** file with the secret in an asymmetric vault named **secret_vault**. This ensures that the service will have to authenticate using a private key in order to retrieve the secret from the vault. The vault members will be able to retrieve the file from any IdM client.

**Prerequisites**

- You have installed the **ansible-freeipa** package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the **IdM administrator** password.
- You have created an asymmetric vault to store the service secret.
- The secret is stored locally on the Ansible controller, for example in the `/usr/share/doc/ansible-freeipa/playbooks/vault/private-key-to-an-externally-signed-certificate.pem` file.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/vault` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/vault
```

2. Optional: Create an inventory file if it does not exist, for example **inventory.file**:

```
$ touch inventory.file
```

3. Open your inventory file and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

4. Make a copy of the **data-archive-in-asymmetric-vault.yml** Ansible playbook file. For example:
$ cp data-archive-in-asymmetric-vault.yml data-archive-in-asymmetric-vault-copy.yml

5. Open the `data-archive-in-asymmetric-vault-copy.yml` file for editing.

6. Modify the file by setting the following variables in the `ipavault` task section:
   - Set the `ipaadmin_password` variable to the IdM administrator password.
   - Set the `name` variable to the name of the vault, for example `secret_vault`.
   - Set the `service` variable to the service owner of the vault, for example `HTTP/webserver1.idm.example.com`.
   - Set the `in` variable to `"{{ lookup('file', 'private-key-to-an-externally-signed-certificate.pem') | b64encode }}"`. This ensures that Ansible retrieves the file with the private key from the working directory on the Ansible controller rather than from the IdM server.
   - Set the `action` variable to `member`.
      This the modified Ansible playbook file for the current example:

\[
---
- name: Tests
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
  - ipavault:
      ipaadmin_password: Secret123
      name: secret_vault
      service: HTTP/webserver1.idm.example.com
      in: "{{ lookup('file', 'private-key-to-an-externally-signed-certificate.pem') | b64encode }}"
      action: member

7. Save the file.

8. Run the playbook:

\[
$ ansible-playbook -v -i inventory.file data-archive-in-asymmetric-vault-copy.yml
\]

## 68.4. RETRIEVING A SERVICE SECRET FOR AN IDM SERVICE USING ANSIBLE

This section shows how an Identity Management (IdM) user can use an Ansible playbook to retrieve a secret from a service vault on behalf of the service. In the example used in the procedure below, running the playbook retrieves a PEM file with the secret from an asymmetric vault named `secret_vault`, and stores it in the specified location on all the hosts listed in the Ansible inventory file as `ipaservers`.

The services authenticate to IdM using keytabs, and they authenticate to the vault using a private key. You can retrieve the file on behalf of the service from any IdM client on which `ansible-freeipa` is installed.

**Prerequisites**
You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.

- You know the IdM administrator password.
- You have created an asymmetric vault to store the service secret.
- You have archived the secret in the vault.
- You have stored the private key used to retrieve the service vault secret in the location specified by the private_key_file variable on the Ansible controller.

**Procedure**

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/vault directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/vault
   ```

2. Optional: Create an inventory file if it does not exist, for example inventory.file:

   ```
   $ touch inventory.file
   ```

3. Open your inventory file and define the following hosts:
   - Define your IdM server in the [ipaserver] section.
   - Define the hosts onto which you want to retrieve the secret in the [webservers] section. For example, to instruct Ansible to retrieve the secret to webserver1.idm.example.com, webserver2.idm.example.com, and webserver3.idm.example.com, enter:

     ```
     [ipaserver]
     server.idm.example.com
     
     [webservers]
     webserver1.idm.example.com
     webserver2.idm.example.com
     webserver3.idm.example.com
     ```

4. Make a copy of the retrieve-data-asymmetric-vault.yml Ansible playbook file. For example:

   ```
   $ cp retrieve-data-asymmetric-vault.yml retrieve-data-asymmetric-vault-copy.yml
   ```

5. Open the retrieve-data-asymmetric-vault-copy.yml file for editing.

6. Modify the file by setting the following variables in the ipavault task section:
   - Set the ipaadmin_password variable to your IdM administrator password.
   - Set the name variable to the name of the vault, for example secret_vault.
   - Set the service variable to the service owner of the vault, for example HTTP/webserver1.idm.example.com.
   - Set the private_key_file variable to the location of the private key used to retrieve the service vault secret.
Set the **out** variable to the location on the IdM server where you want to retrieve the `private-key-to-an-externally-signed-certificate.pem` secret, for example the current working directory.

Set the **action** variable to **member**. This the modified Ansible playbook file for the current example:

```yaml
---
- name: Retrieve data from vault
  hosts: ipaserver
  become: no
  gather_facts: false
  tasks:
    - name: Retrieve data from the service vault
      ipavault:
        ipaadmin_password: Secret123
        name: secret_vault
        service: HTTP/webserver1.idm.example.com
        vault_type: asymmetric
        private_key: "{{ lookup('file', 'service-private.pem') | b64encode }}"
        out: private-key-to-an-externally-signed-certificate.pem
        state: retrieved

7. Add a section to the playbook that retrieves the data file from the IdM server to the Ansible controller:

```yaml
---
- name: Retrieve data from vault
  hosts: ipaserver
  become: no
  gather_facts: false
  tasks:
    [...]
    - name: Retrieve data file
      fetch:
        src: private-key-to-an-externally-signed-certificate.pem
        dest: ./
        flat: yes
        mode: 0600
```

8. Add a section to the playbook that transfers the retrieved `private-key-to-an-externally-signed-certificate.pem` file from the Ansible controller on to the webservers listed in the **webservers** section of the inventory file:

```yaml
---
- name: Send data file to webservers
  become: no
  gather_facts: no
  hosts: webservers
  tasks:
    - name: Send data to webservers
      copy:
```
src: private-key-to-an-externally-signed-certificate.pem
dest: /etc/pki/tls/private/httpd.key
mode: 0444

9. Save the file.
10. Run the playbook:

$ ansible-playbook -v -i inventory.file retrieve-data-asymmetric-vault-copy.yml

68.5. CHANGING AN IDM SERVICE VAULT SECRET WHEN COMPROMISED USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can reuse an Ansible playbook to change the secret stored in a service vault when a service instance has been compromised. The scenario in the following example assumes that on webserver3.idm.example.com, the retrieved secret has been compromised, but not the key to the asymmetric vault storing the secret. In the example, the administrator reuses the Ansible playbooks used when storing a secret in an asymmetric vault and retrieving a secret from the asymmetric vault onto IdM hosts. At the start of the procedure, the IdM administrator stores a new PEM file with a new secret in the asymmetric vault, adapts the inventory file so as not to retrieve the new secret on to the compromised web server, webserver3.idm.example.com, and then re-runs the two procedures.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
- You have created an asymmetric vault to store the service secret.
- You have generated a new httpd key for the web services running on IdM hosts to replace the compromised old key.
- The new httpd key is stored locally on the Ansible controller, for example in the /usr/share/doc/ansible-freeipa/playbooks/vault/private-key-to-an-externally-signed-certificate.pem file.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/vault directory:

$ cd /usr/share/doc/ansible-freeipa/playbooks/vault

2. Open your inventory file and make sure that the following hosts are defined correctly:

- The IdM server in the [ipaserver] section.
- The hosts onto which you want to retrieve the secret in the [webservers] section. For example, to instruct Ansible to retrieve the secret to webserver1.idm.example.com and webserver2.idm.example.com, enter:

  [ipaserver]
server.idm.example.com

[webservers]
webserver1.idm.example.com
webserver2.idm.example.com

IMPORTANT

Make sure that the list does not contain the compromised webserver, in the
current example webserver3.idm.example.com.

3. Open the `data-archive-in-asymmetric-vault-copy.yml` file for editing.

4. Modify the file by setting the following variables in the `ipavault` task section:
   - Set the `ipaadmin_password` variable to the IdM administrator password.
   - Set the `name` variable to the name of the vault, for example `secret_vault`.
   - Set the `service` variable to the service owner of the vault, for example `HTTP/webserver.idm.example.com`.
   - Set the `in` variable to `"{{ lookup('file', 'new-private-key-to-an-externally-signed-certificate.pem') | b64encode }}"`. This ensures that Ansible retrieves the file with the private key from the working directory on the Ansible controller rather than from the IdM server.
   - Set the `action` variable to `member`.

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

   ```yaml
   ---
   - name: Tests
     hosts: ipaserver
     become: true
     gather_facts: false

     tasks:
     - ipavault:
         ipaadmin_password: Secret123
         name: secret_vault
         service: HTTP/webserver.idm.example.com
         in: "{{ lookup('file', 'new-private-key-to-an-externally-signed-certificate.pem') | b64encode }}"
         action: member
   ```

5. Save the file.

6. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file data-archive-in-asymmetric-vault-copy.yml
   ```

7. Open the `retrieve-data-asymmetric-vault-copy.yml` file for editing.

8. Modify the file by setting the following variables in the `ipavault` task section:
- Set the **ipaadmin_password** variable to your IdM administrator password.
- Set the **name** variable to the name of the vault, for example **secret_vault**.
- Set the **service** variable to the service owner of the vault, for example **HTTP/webserver1.idm.example.com**.
- Set the **private_key_file** variable to the location of the private key used to retrieve the service vault secret.
- Set the **out** variable to the location on the IdM server where you want to retrieve the **new-private-key-to-an-externally-signed-certificate.pem** secret, for example the current working directory.
- Set the **action** variable to **member**.

This the modified Ansible playbook file for the current example:

```
---
- name: Retrieve data from vault
  hosts: ipaserver
  become: no
  gather_facts: false

  tasks:
  - name: Retrieve data from the service vault
    ipavault:
      ipaadmin_password: Secret123
      name: secret_vault
      service: HTTP/webserver1.idm.example.com
      vault_type: asymmetric
      private_key: "{{ lookup('file', 'service-private.pem') | b64encode }}"
      out: new-private-key-to-an-externally-signed-certificate.pem
      state: retrieved
```

9. Add a section to the playbook that retrieves the data file from the IdM server to the Ansible controller:

```
---
- name: Retrieve data file
  hosts: ipaserver
  become: yes
  gather_facts: false
  tasks:
  [...] 
  - name: Retrieve data file
    fetch:
      src: new-private-key-to-an-externally-signed-certificate.pem
      dest: ./
      flat: yes
      mode: 0600
```

10. Add a section to the playbook that transfers the retrieved **new-private-key-to-an-externally-signed-certificate.pem** file from the Ansible controller on to the webservers listed in the **webservers** section of the inventory file:
---
- name: Send data file to webservers
  become: yes
  gather_facts: no
  hosts: webservers
  tasks:
  - name: Send data to webservers
    copy:
      src: new-private-key-to-an-externally-signed-certificate.pem
      dest: /etc/pki/tls/private/httpd.key
      mode: 0444

11. Save the file.

12. Run the playbook:

```
$ ansible-playbook -v -i inventory.file retrieve-data-asymmetric-vault-copy.yml
```

Additional resources

- For more information about using Ansible to manage IdM vaults and service secrets and about playbook variables, see the README-vault.md Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory and the sample playbooks available in the `/usr/share/doc/ansible-freeipa/playbooks/vault/` directory.
CHAPTER 69. ENSURING THE PRESENCE AND ABSENCE OF SERVICES IN IDM USING ANSIBLE

With the Ansible service module, Identity Management (IdM) administrator can ensure that specific services that are not native to IdM are present or absent in IdM. For example, you can use the service module to:

- Check that a manually installed service is present on an IdM client and automatically install that service if it is absent. For details, see:
  - Ensuring the presence of an HTTP service in IdM on an IdM client.
  - Ensuring the presence of an HTTP service in IdM on a non-IdM client.
  - Ensuring the presence of an HTTP service on an IdM client without DNS.

- Check that a service enrolled in IdM has a certificate attached and automatically install that certificate if it is absent. For details, see:
  - Ensuring the presence of an externally-signed certificate in an IdM service entry.

- Allow IdM users and hosts to retrieve and create the service keytab. For details, see:
  - Allowing IdM users, groups, hosts, or host groups to create a keytab of a service.
  - Allowing IdM users, groups, hosts, or host groups to retrieve a keytab of a service.

- Allow IdM users and hosts to add a Kerberos alias to a service. For details, see:
  - Ensuring the presence of a Kerberos principal alias for a service.

- Check that a service is not present on an IdM client and automatically remove that service if it is present. For details, see:
  - Ensuring the absence of an HTTP service in IdM on an IdM client.

69.1. ENSURING THE PRESENCE OF AN HTTP SERVICE IN IDM USING AN ANSIBLE PLAYBOOK

This section describes how to ensure the presence of an HTTP server in IdM using an Ansible playbook.

Prerequisites

- The system to host the HTTP service is an IdM client.
- You have the IdM administrator password.

Procedure

1. Create an inventory file, for example inventory.file:

   ```bash
   $ touch inventory.file
   ```

2. Open the inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:
3. Make a copy of the `/usr/share/doc/ansible-freeipa/playbooks/service/service-is-present.yml` Ansible playbook file. For example:

```bash
$ cp /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present.yml /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-copy.yml
```

4. Open the `/usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-copy.yml` Ansible playbook file for editing:

```yaml
---
- name: Playbook to manage IPA service.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
  # Ensure service is present
  - ipaservice:
     ipaadmin_password: Secret123
     name: HTTP/client.idm.example.com
```

5. Adapt the file:

- Change the IdM administrator password defined by the `ipaadmin_password` variable.
- Change the name of your IdM client on which the HTTP service is running, as defined by the `name` variable of the `ipaservice` task.

6. Save and exit the file.

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

```bash
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-copy.yml
```

**Verification steps**

1. Log into the IdM Web UI as IdM administrator.

2. Navigate to **Identity → Services**.

If `HTTP/client.idm.example.com@IDM.EXAMPLE.COM` is listed in the **Services** list, the Ansible playbook has been successfully added to IdM.

**Additional resources**

- You can secure the communication between the HTTP server and browser clients by adding **TLS encryption to an Apache HTTP Server**.
You can request a certificate for the HTTP service from an IdM certificate authority. For more information, see the procedure described in Obtaining an IdM certificate for a service using certmonger.

69.2. ENSURING THE PRESENCE OF AN HTTP SERVICE IN IDM ON A NON-IDM CLIENT USING AN ANSIBLE PLAYBOOK

This section describes how to ensure the presence of an HTTP server in IdM on a host that is not an IdM client using an Ansible playbook. By adding the HTTP server to IdM you are also adding the host to IdM.

Prerequisites

- You have installed an HTTP service on your host.
- The host on which you have set up HTTP is not an IdM client. Otherwise, follow the steps in Ensuring the presence of an HTTP service in IdM using an Ansible playbook.
- You have the IdM administrator password.
- The DNS A record - or the AAAA record if IPv6 is used - for the host is available.

Procedure

1. Create an inventory file, for example `inventory.file`:

   ```
   $ touch inventory.file
   ```

2. Open the `inventory.file` and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `/usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-without-host-check.yml` Ansible playbook file. For example:

   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-without-host-check.yml /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-without-host-check-copy.yml
   ```

4. Open the copied file, `/usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-without-host-check-copy.yml`, for editing. Locate the `ipaadmin_password` and `name` variables in the `ipaservice` task:

   ```
   ---
   - name: Playbook to manage IPA service.
     hosts: ipaserver
     become: true
     gather_facts: false

     tasks:
     # Ensure service is present
     - ipaservice:
   ```
5. Adapt the file:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `name` variable to the name of the host on which the HTTP service is running.

6. Save and exit the file.

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

   ```bash
   ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-without-host-check-copy.yml
   ```

**Verification steps**

1. Log into the IdM Web UI as IdM administrator.
2. Navigate to **Identity → Services**.

You can now see **HTTP/client.idm.example.com@IDM.EXAMPLE.COM** listed in the Services list.

**Additional resources**

- You can secure the communication between the HTTP server and the browser clients by adding TLS encryption to an Apache HTTP Server.

---

**69.3. ENSURING THE PRESENCE OF AN HTTP SERVICE ON AN IDM CLIENT WITHOUT DNS USING AN ANSIBLE PLAYBOOK**

This section describes how to ensure the presence of an HTTP server running on an IdM client that has no DNS entry using an Ansible playbook. The scenario implied is that the IdM host has no DNS A entry available – or no DNS AAAA entry if IPv6 is used instead of IPv4.

**Prerequisites**

- The system to host the HTTP service is enrolled in IdM.
- The DNS A or DNS AAAA record for the host may not exist. Otherwise, if the DNS record for the host does exist, follow the procedure in **Ensuring the presence of an HTTP service in IdM using an Ansible playbook**.
- You have the IdM administrator password.

**Procedure**

1. Create an inventory file, for example `inventory.file`:

   ```bash
   touch inventory.file
   ```
2. Open the.inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-with-host-force.yml Ansible playbook file. For example:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-with-host-force.yml /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-with-host-force-copy.yml
```

4. Open the copied file, /usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-with-host-force-copy.yml, for editing. Locate the ipaadmin_password and name variables in the ipaservice task:

```
---
- name: Playbook to manage IPA service.
  hosts: ipaserver
  become: true
  gather_facts: false
  tasks:
    # Ensure service is present
    - ipaservice:
        ipaadmin_password: MyPassword123
        name: HTTP/ihavenodns.info
        force: yes
```

5. Adapt the file:

- Set the ipaadmin_password variable to your IdM administrator password.
- Set the name variable to the name of the host on which the HTTP service is running.

6. Save and exit the file.

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

```
$ ansible-playbook -v -i /path_to_inventory_directory/inventory.file
/usr/share/doc/ansible-freeipa/playbooks/service/service-is-present-with-host-force-copy.yml
```

**Verification steps**

1. Log into the IdM Web UI as IdM administrator.
2. Navigate to Identity → Services.

You can now see HTTP/client.idm.example.com@IDM.EXAMPLE.COM listed in the Services list.

**Additional resources**
You can secure the communication between the Apache HTTP server and browser clients by adding TLS encryption to the Apache HTTP Server.

69.4. ENSURING THE PRESENCE OF AN EXTERNALLY SIGNED CERTIFICATE IN AN IDM SERVICE ENTRY USING AN ANSIBLE PLAYBOOK

This section describes how to use the ansible-freeipa service module to ensure that a certificate issued by an external certificate authority (CA) is attached to the IdM entry of the HTTP service. Having the certificate of an HTTP service signed by an external CA rather than the IdM CA is particularly useful if your IdM CA uses a self-signed certificate.

Prerequisites

- You have installed an HTTP service on your host.
- You have enrolled the HTTP service to IdM.
- You have the IdM administrator password.
- You have an externally signed certificate whose Subject corresponds to the principal of the HTTP service.

Procedure

1. Create an inventory file, for example inventory.file:

   ```
   $ touch inventory.file
   ```

2. Open the inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the /usr/share/doc/ansible-freeipa/playbooks/service/service-member-certificate-present.yml file, for example:

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

4. Optional: If the certificate is in the Privacy Enhanced Mail (PEM) format, convert the certificate to the Distinguished Encoding Rules (DER) format for easier handling through the command-line interface (CLI):

   ```
   $ openssl x509 -outform der -in cert1.pem -out cert1.der
   ```

5. Decode the DER file to standard output using the base64 command. Use the -w0 option to disable wrapping:

   ```
   $ base64 cert1.der -w0
   ```
6. Copy the certificate from the standard output to the clipboard.

7. Open the `/usr/share/doc/ansible-freeipa/playbooks/service/service-member-certificate-present-copy.yml` file for editing and view its contents:

```
---
- name: Service certificate present.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
    # Ensure service certificate is present
    - ipaservice:
      ipaadmin_password: MyPassword123
      name: HTTP/www.example.com
      certificate: |
        MIICBjCCAW8CFHnm32VcXaUDGfEGdDL/...
        ...
        action: member
        state: present
```

8. Adapt the file:

- Replace the certificate, defined using the `certificate` variable, with the certificate you copied from the CLI. Note that if you use the `certificate:` variable with the `|` pipe character as indicated, you can enter the certificate THIS WAY rather than having it to enter it in a single line. This makes reading the certificate easier.

- Change the IdM administrator password, defined by the `ipaadmin_password` variable.

- Change the name of your IdM client on which the HTTP service is running, defined by the `name` variable.

- Change any other relevant variables.

9. Save and exit the file.

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

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-member-certificate-present-copy.yml
```

**Verification steps**

1. Log into the IdM Web UI as IdM administrator.

2. Navigate to **Identity → Services**.

3. Click the name of the service with the newly added certificate, for example `HTTP/client.idm.example.com`.

In the **Service Certificate** section on the right, you can now see the newly added certificate.
69.5. USING AN ANSIBLE PLAYBOOK TO ALLOW IDM USERS, GROUPS, HOSTS, OR HOST GROUPS TO CREATE A KEYTAB OF A SERVICE

A keytab is a file containing pairs of Kerberos principals and encrypted keys. Keytab files are commonly used to allow scripts to automatically authenticate using Kerberos, without requiring human interaction or access to password stored in a plain-text file. The script is then able to use the acquired credentials to access files stored on a remote system.

As an Identity Management (IdM) administrator, you can allow other users to retrieve or even create a keytab for a service running in IdM. By allowing specific users and user groups to create keytabs, you can delegate the administration of the service to them without sharing the IdM administrator password. This delegation provides a more fine-grained system administration.

This section describes how you can allow specific IdM users, user groups, hosts, and host groups to create a keytab for the HTTP service running on an IdM client. Specifically, it describes how you can allow the user01 IdM user to create a keytab for the HTTP service running on an IdM client named client.idm.example.com.

Prerequisites

- You know the IdM administrator password.
- You have installed the ansible-freeipa package on the Ansible controller.
- You have enrolled the HTTP service to IdM.
- The system to host the HTTP service is an IdM client.
- The IdM users and user groups that you want to allow to create the keytab exist in IdM.
- The IdM hosts and host groups that you want to allow to create the keytab exist in IdM.

Procedure

1. Create an inventory file, for example inventory.file:

   ```
   $ touch inventory.file
   ```

2. Open the inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the /usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_create_keytab-present.yml Ansible playbook file. For example:

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

4. Open the /usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_create_keytab-present-copy.yml Ansible playbook file for editing.
5. Adapt the file by changing the following:

- The IdM administrator password specified by the `ipaadmin_password` variable.
- The name of your IdM client on which the HTTP service is running. In the current example, it is `HTTP/client.idm.example.com`.
- The names of IdM users that are listed in the `allow_create_keytab_user:` section. In the current example, it is `user01`.
- The names of IdM user groups that are listed in the `allow_create_keytab_group:` section.
- The names of IdM hosts that are listed in the `allow_create_keytab_host:` section.
- The names of IdM host groups that are listed in the `allow_create_keytab_hostgroup:` section.
- The name of the task specified by the `name` variable in the `tasks` section.

After being adapted for the current example, the copied file looks like this:

```yaml
---
- name: Service member allow_create_keytab present
  hosts: ipaserver
  become: true

  tasks:
  - name: Service HTTP/client.idm.example.com members allow_create_keytab present for user01
    ipaservice:
      ipaadmin_password: Secret123
      name: HTTP/client.idm.example.com
      allow_create_keytab_user:
      - user01
      action: member
```

6. Save the file.

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

```bash
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_create_keytab-present-copy.yml
```

Verification steps

1. SSH to an IdM server as an IdM user that has the privilege to create a keytab for the particular HTTP service:

   ```bash
   $ ssh user01@server.idm.example.com
   Password:
   ```

2. Use the `ipa-getkeytab` command to generate the new keytab for the HTTP service:

   ```bash
   $ ipa-getkeytab -s server.idm.example.com -p HTTP/client.idm.example.com -k /etc/httpd/conf/krb5.keytab
   ```
The `-s` option specifies a Key Distribution Center (KDC) server to generate the keytab.

The `-p` option specifies the principal whose keytab you want to create.

The `-k` option specifies the keytab file to append the new key to. The file will be created if it does not exist.

If the command does not result in an error, you have successfully created a keytab of `HTTP/client.idm.example.com` as `user01`.

### 69.6. USING AN ANSIBLE PLAYBOOK TO ALLOW IDM USERS, GROUPS, HOSTS, OR HOST GROUPS TO RETRIEVE A KEYTAB OF A SERVICE

A keytab is a file containing pairs of Kerberos principals and encrypted keys. Keytab files are commonly used to allow scripts to automatically authenticate using Kerberos, without requiring human interaction or access to a password stored in a plain-text file. The script is then able to use the acquired credentials to access files stored on a remote system.

As IdM administrator, you can allow other users to retrieve or even create a keytab for a service running in IdM.

This section describes how you can allow specific IdM users, user groups, hosts, and host groups to retrieve a keytab for the HTTP service running on an IdM client. Specifically, it describes how to allow the `user01` IdM user to retrieve the keytab of the HTTP service running on `client.idm.example.com`.

#### Prerequisites

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.
- You have enrolled the HTTP service to IdM.
- The IdM users and user groups that you want to allow to retrieve the keytab exist in IdM.
- The IdM hosts and host groups that you want to allow to retrieve the keytab exist in IdM.

#### Procedure

1. Create an inventory file, for example `inventory.file`:

   ```
   $ touch inventory.file
   ```

2. Open the `inventory.file` and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `/usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_retrieve_keytab-present.yml` Ansible playbook file. For example:
4. Open the copied file, `/usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_retrieve_keytab-present-copy.yml`, for editing:

5. Adapt the file:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `name` variable of the `ipaservice` task to the principal of the HTTP service. In the current example, it is `HTTP/client.idm.example.com`
   - Specify the names of IdM users in the `allow_retrieve_keytab_group:` section. In the current example, it is `user01`.
   - Specify the names of IdM user groups in the `allow_retrieve_keytab_group:` section.
   - Specify the names of IdM hosts in the `allow_retrieve_keytab_group:` section.
   - Specify the names of IdM host groups in the `allow_retrieve_keytab_group:` section.
   - Specify the name of the task using the `name` variable in the `tasks` section.

After being adapted for the current example, the copied file looks like this:

```
---
- name: Service member allow_retrieve_keytab present
  hosts: ipaserver
  become: true

  tasks:
  - name: Service HTTP/client.idm.example.com members allow_retrieve_keytab present for user01
    ipaservice:
      ipaadmin_password: Secret123
      name: HTTP/client.idm.example.com
      allow_retrieve_keytab_user:
        - user01
      action: member
```

6. Save the file.

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

```
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-member-allow_retrieve_keytab-present-copy.yml
```

**Verification steps**

1. SSH to an IdM server as an IdM user with the privilege to retrieve a keytab for the HTTP service:

```
$ ssh user01@server.idm.example.com
Password:
```
2. Use the `ipa-getkeytab` command with the `-r` option to retrieve the keytab:

   ```bash
   $ ipa-getkeytab -r -s server.idm.example.com -p HTTP/client.idm.example.com -k /etc/httpd/conf/krb5.keytab
   ``

   The `-s` option specifies a Key Distribution Center (KDC) server from which you want to retrieve the keytab.

   The `-p` option specifies the principal whose keytab you want to retrieve.

   The `-k` option specifies the keytab file to which you want to append the retrieved key. The file will be created if it does not exist.

   If the command does not result in an error, you have successfully retrieved a keytab of `HTTP/client.idm.example.com` as `user01`.

### 69.7. ENSURING THE PRESENCE OF A KERBEROS PRINCIPAL ALIAS OF A SERVICE USING AN ANSIBLE PLAYBOOK

In some scenarios, it is beneficial for IdM administrator to enable IdM users, hosts, or services to authenticate against Kerberos applications using a Kerberos principal alias. These scenarios include:

- The user name changed, but the user should be able to log into the system using both the previous and new user names.
- The user needs to log in using the email address even if the IdM Kerberos realm differs from the email domain.

This section describes how to create the principal alias of `HTTP/mycompany.idm.example.com` for the HTTP service running on `client.idm.example.com`.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible controller.
- You have set up an HTTP service on your host.
- You have enrolled the HTTP service to IdM.
- The host on which you have set up HTTP is an IdM client.

**Procedure**

1. Create an inventory file, for example `inventory.file`:

   ```bash
   $ touch inventory.file
   ```

2. Open the `inventory.file` and define the IdM server that you want to configure in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

   ```bash
   [ipaserver]
   server.idm.example.com
   ```
3. Make a copy of the `/usr/share/doc/ansible-freeipa/playbooks/service/service-member-principal-present.yml` Ansible playbook file. For example:

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

4. Open the `/usr/share/doc/ansible-freeipa/playbooks/service/service-member-principal-present-copy.yml` Ansible playbook file for editing.

5. Adapt the file by changing the following:

   - The IdM administrator password specified by the `ipaadmin_password` variable.
   - The name of the service specified by the `name` variable. This is the canonical principal name of the service. In the current example, it is `HTTP/client.idm.example.com`.
   - The Kerberos principal alias specified by the `principal` variable. This is the alias you want to add to the service defined by the `name` variable. In the current example, it is `host/mycompany.idm.example.com`.
   - The name of the task specified by the `name` variable in the `tasks` section.

   After being adapted for the current example, the copied file looks like this:

   ```yaml
   ---
   - name: Service member principal present
     hosts: ipaserver
     become: true

     tasks:
     - name: Service HTTP/client.idm.example.com member principals
       host/mycompany.idm.example.com present
       ipaservice:
         ipaadmin_password: Secret123
         name: HTTP/client.idm.example.com
         principal:
           - host/mycompany.idm.example.com
           action: member
   ```

6. Save the file.

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

   ```
   $ ansible-playbook -v -i path_to_inventory_directory/inventory.file /usr/share/doc/ansible-freeipa/playbooks/service/service-member-principal-present-copy.yml
   ```

If running the playbook results in 0 unreachable and 0 failed tasks, you have successfully created the `host/mycompany.idm.example.com` Kerberos principal for the `HTTP/client.idm.example.com` service.

**Additional resources**

- For more information on Kerberos principal aliases and managing them without Ansible, see [Managing Kerberos principal aliases for users, hosts, and services](#).
69.8. ENSURING THE ABSENCE OF AN HTTP SERVICE IN IDM USING AN ANSIBLE PLAYBOOK

This section describes how to unenroll a service from IdM. More specifically, it describes how to use an Ansible playbook to ensure the absence of an HTTP server named HTTP/client.idm.example.com in IdM.

Prerequisites

- You have the IdM administrator password.

Procedure

1. Create an inventory file, for example inventory.file:
   
   ```
   $ touch inventory.file
   ```

2. Open the inventory.file and define the IdM server that you want to configure in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:
   
   ```
   [ipaserver]
   server.idm.example.com
   ```

3. Make a copy of the /usr/share/doc/ansible-freeipa/playbooks/service/service-is-absent.yml Ansible playbook file. For example:
   
   ```
   $ cp /usr/share/doc/ansible-freeipa/playbooks/service/service-is-absent.yml /usr/share/doc/ansible-freeipa/playbooks/service/service-is-absent-copy.yml
   ```

4. Open the /usr/share/doc/ansible-freeipa/playbooks/service/service-is-absent-copy.yml Ansible playbook file for editing.

5. Adapt the file by changing the following:
   
   - The IdM administrator password defined by the ipaadmin_password variable.
   - The Kerberos principal of the HTTP service, as defined by the name variable of the ipaservice task.

   After being adapted for the current example, the copied file looks like this:

   ```
   ---
   - name: Playbook to manage IPA service.
     hosts: ipaserver
     become: true
     gather_facts: false

     tasks:
     # Ensure service is absent
     - ipaservice:
         ipaadmin_password: Secret123
         name: HTTP/client.idm.example.com
         state: absent
   ```

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

```bash
$ ansible-playbook -v -i path_to_inventory_directory/inventory.file
/usr/share/doc/ansible-freeipa/playbooks/service/service-is-absent-copy.yml
```

**Verification steps**

1. Log into the IdM Web UI as IdM administrator.

2. Navigate to **Identity → Services**.

If you cannot see the `HTTP/client.idm.example.com@IDM.EXAMPLE.COM` service in the **Services** list, you have successfully ensured its absence in IdM.

**Additional resources**

- You can see sample Ansible playbooks for ensuring the presence and absence of services in IdM including a list of possible variables in the **README-service.md** Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory.

- You can see sample Ansible playbooks for ensuring the presence and absence of services in IdM in the `/usr/share/doc/ansible-freeipa/playbooks/config` directory.
CHAPTER 70. ENABLING AD USERS TO ADMINISTER IDM

70.1. ID OVERIDES 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.

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

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

This procedure describes creating and using an ID override for an AD user to give that user rights identical to those of an IdM user. During this procedure, work on an IdM server that is configured as a trust controller or a trust agent. For details on trust controllers and trust agents, see Trust controllers and trust agents in Planning Identity Management.

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
```

70.3. 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
```
CHAPTER 71. 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

71.1. 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_sufffix 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.
71.2. 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)
```

71.3. SETTING THE DOMAIN RESOLUTION ORDER FOR AN ID VIEW ON AN IDM SERVER

This procedure sets the domain resolution order for an ID view that you can apply to a specific set of IdM servers and clients. This example creates an ID view named `ADsubdomain1_first` for IdM host `client1.idm.example.com`, and sets the domain resolution order to search for users and groups in the following order:

1. Active Directory (AD) child domain subdomain1.ad.example.com
2. AD root domain `ad.example.com`

3. IdM domain `idm.example.com`

NOTE

The domain resolution order set in an ID view overrides the global domain resolution order, but it does not override any domain resolution order set locally in the SSSD configuration.

Prerequisites

- You have configured a trust with an AD environment.

Procedure

1. Create an ID view with the `--domain-resolution-order` option set.

   ```
   [user@server ~]$ ipa idview-add ADsubdomain1_first --desc "ID view for resolving AD subdomain1 first on client1.idm.example.com" --domain-resolution-order subdomain1.ad.example.com:ad.example.com:idm.example.com
   ---------------------------------
   Added ID View "ADsubdomain1_first"
   ---------------------------------
   ID View Name: ADsubdomain1_first
   Description: ID view for resolving AD subdomain1 first on client1.idm.example.com
   Domain Resolution Order:
   subdomain1.ad.example.com:ad.example.com:idm.example.com
   ```

2. Apply the ID view to IdM hosts.

   ```
   [user@server ~]$ ipa idview-apply ADsubdomain1_first --hosts client1.idm.example.com
   -----------------------------------
   Applied ID View "ADsubdomain1_first"
   -----------------------------------
   hosts: client1.idm.example.com
   ---------------------------------------------
   Number of hosts the ID View was applied to: 1
   ---------------------------------------------
   ```

Verification steps

- Display the details of the ID view.

  ```
  [user@server ~]$ ipa idview-show ADsubdomain1_first --show-hosts
  ID View Name: ADsubdomain1_first
  Description: ID view for resolving AD subdomain1 first on client1.idm.example.com
  Hosts the view applies to: client1.idm.example.com
  Domain resolution order:
  subdomain1.ad.example.com:ad.example.com:idm.example.com
  ```

- Verify you can retrieve user information for a user from the `subdomain1.ad.example.com` domain using only a short name.
71.4. 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.

```bash
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.

```bash
[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.

```bash
[root@client2 ~]# id <user_from_subdomain1>
uid=1916901106(user_from_subdomain1) gid=1916900513(domain users)
groups=1916900513(domain users)
```
CHAPTER 72. ENABLING AUTHENTICATION USING AD USER PRINCIPAL NAMES IN IDM

72.1. 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

72.2. 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.

### 72.3. 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

- For additional troubleshooting steps, see the Knowledgebase article Using rpcclient to gather troubleshooting data for AD UPN authentication issues.
CHAPTER 73. USING CANONICALIZED DNS HOST NAMES IN IDM

DNS canonicalization is disabled by default on Identity Management (IdM) clients to avoid potential security risks. For example, if an attacker controls the DNS server and a host in the domain, the attacker can cause the short host name, such as demo, to resolve to a compromised host, such as malicious.example.com. In this case, the user connects to a different server than expected.

This section describes how to use canonicalized host names on IdM clients.

73.1. ADDING AN ALIAS TO A HOST PRINCIPAL

By default, Identity Management (IdM) clients enrolled by using the ipa-client-install command do not allow to use short host names in service principals. For example, users can use only host/demo.example.com@EXAMPLE.COM instead of host/demo@EXAMPLE.COM when accessing a service.

This section explains how to add an alias to a Kerberos principal. Note that you can alternatively enable canonicalization of host names in the /etc/krb5.conf file. For details, see Section 73.2, “Enabling canonicalization of host names in service principals on clients”.

Prerequisites

- The IdM client is installed.
- The host name is unique in the network.

Procedure

1. Authenticate to IdM as the admin user:

   $ kinit admin

2. Add the alias to the host principal. For example, to add the demo alias to the demo.example.com host principal:

   $ ipa host-add-principal demo.example.com --principal=demo

73.2. ENABLING CANONICALIZATION OF HOST NAMES IN SERVICE PRINCIPALS ON CLIENTS

This section describes how you enable canonicalization of host names in services principals on clients.

Note that if you use host principal aliases, as described in Section 73.1, “Adding an alias to a host principal”, you do not need to enable canonicalization.

Prerequisites

- The Identity Management (IdM) client is installed.
- You are logged in to the IdM client as the root user.
- The host name is unique in the network.
Procedure

1. Set the `dns_canonicalize_hostname` parameter in the `[libdefaults]` section in the `/etc/krb5.conf` file to `false`:

   ```
   [libdefaults]
   ...
   dns_canonicalize_hostname = true
   ```

73.3. OPTIONS FOR USING HOST NAMES WITH DNS HOST NAME CANONICALIZATION ENABLED

If you set `dns_canonicalize_hostname = true` in the `/etc/krb5.conf` file as explained in Section 73.2, “Enabling canonicalization of host names in service principals on clients”, you have the following options when you use a host name in a service principal:

- In Identity Management (IdM) environments, you can use the full host name in a service principal, such as `host/demo.example.com@EXAMPLE.COM`.

- In environments without IdM, but if the RHEL host as a member of an Active Directory (AD) domain, no further considerations are required, because AD domain controllers (DC) automatically create service principals for NetBIOS names of the machines enrolled into AD.
CHAPTER 74. MANAGING GLOBAL DNS CONFIGURATION IN IDM USING ANSIBLE PLAYBOOKS

Using the Red Hat Ansible Engine `dnsconfig` module, you can configure global configuration for Identity Management (IdM) DNS. Settings defined in global DNS configuration are applied to all IdM DNS servers. However, the global configuration has lower priority than the configuration for a specific IdM DNS zone.

The `dnsconfig` module supports the following variables:

- The global forwarders, specifically their IP addresses and the port used for communication.
- The global forwarding policy: only, first, or none. For more details on these types of DNS forward policies, see DNS forward policies in IdM.
- The synchronization of forward lookup and reverse lookup zones.

Prerequisites

- DNS service is installed on the IdM server. For more information about how to install an IdM server with integrated DNS, see one of the following links:
  - Installing an IdM server: With integrated DNS, with an integrated CA as the root CA
  - Installing an IdM server: With integrated DNS, with an external CA as the root CA
  - Installing an IdM server: With integrated DNS, without a CA

This chapter includes the following sections:

- How IdM ensures that global forwarders from `/etc/resolv.conf` are not removed by NetworkManager
- Ensuring the presence of a DNS global forwarder in IdM using Ansible
- Ensuring the absence of a DNS global forwarder in IdM using Ansible
- An introduction to DNS forward policies in IdM
- Using an Ansible playbook to ensure that the forward first policy is set in IdM DNS global configuration
- Using an Ansible playbook to ensure that global forwarders are disabled in IdM DNS
- Using an Ansible playbook to ensure that synchronization of forward and reverse lookup zones is disabled in IdM DNS

74.1. HOW IDM ENSURES THAT GLOBAL FORWARDERS FROM /ETC/RESOLV.CONF ARE NOT REMOVED BY NETWORKMANAGER

Installing Identity Management (IdM) with integrated DNS configures the `/etc/resolv.conf` file to point to the `127.0.0.1` localhost address:
In certain environments, such as networks that use **Dynamic Host Configuration Protocol** (DHCP), the NetworkManager service may revert changes to the `/etc/resolv.conf` file. To make the DNS configuration persistent, the IdM DNS installation process also configures the NetworkManager service in the following way:

1. The DNS installation script creates an `/etc/NetworkManager/conf.d/zzz-ipa.conf` NetworkManager configuration file to control the search order and DNS server list:

   ```
   # auto-generated by IPA installer
   [main]
   dns=default

   [global-dns]
   searches=$DOMAIN

   [global-dns-domain-*]
   servers=127.0.0.1
   
   # Generated by NetworkManager
   search idm.example.com
   nameserver 127.0.0.1
   ```

2. The NetworkManager service is reloaded, which always creates the `/etc/resolv.conf` file with the settings from the last file in the `/etc/NetworkManager/conf.d/` directory. This is in this case the `zzz-ipa.conf` file.

   **IMPORTANT**

   Do not modify the `/etc/resolv.conf` file manually.

### 74.2. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the presence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of **7.7.9.9** and IP v6 address of **2001:db8::1:0** on port **53**.

**Prerequisites**

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```
2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the forwarders-absent.yml Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-of-a-global-forwarder.yml
```

4. Open the ensure-presence-of-a-global-forwarder.yml file for editing.

5. Adapt the file by setting the following variables:
   a. Change the name variable for the playbook to Playbook to ensure the presence of a global forwarder in IdM DNS.
   b. In the tasks section, change the name of the task to Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53.
   c. In the forwarders section of the ipadnsconfig portion:
      i. Change the first ip_address value to the IPv4 address of the global forwarder: 7.7.9.9.
      ii. Change the second ip_address value to the IPv6 address of the global forwarder: 2001:db8::1:0.
      iii. Verify the port value is set to 53.
   d. Change the state to present.
      This the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the presence of a global forwarder in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53
    ipadnsconfig:
      forwarders:
        - ip_address: 7.7.9.9
        - ip_address: 2001:db8::1:0
      port: 53
      state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook -v -i inventory.file ensure-presence-of-a-global-forwarder.yml
```

Additional resources
You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsconfig` module in the `README-dnsconfig.md` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of `ipdnsconfig` variables.

### 74.3. ENSURING THE ABSENCE OF A DNSGLOBAL FORWARDER IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the absence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS global forwarder with an Internet Protocol (IP) v4 address of `8.8.6.6` and IP v6 address of `2001:4860:4860::8800` on port `53`.

**Prerequisites**

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `forwarders-absent.yml` Ansible playbook file. For example:

   ```bash
   $ cp forwarders-absent.yml ensure-absence-of-a-global-forwarder.yml
   ```

4. Open the `ensure-absence-of-a-global-forwarder.yml` file for editing.

5. Adapt the file by setting the following variables:

   a. Change the `name` variable for the playbook to **Playbook to ensure the absence of a global forwarder in IdM DNS**.

   b. In the `tasks` section, change the `name` of the task to **Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53**.

   c. In the `forwarders` section of the `ipdnsconfig` portion:

      i. Change the first `ip_address` value to the IPv4 address of the global forwarder: `8.8.6.6`.

      ii. Change the second `ip_address` value to the IPv6 address of the global forwarder: `2001:4860:4860::8800`.

      iii. Verify the `port` value is set to `53`. 
d. Set the `action` variable to `member`.

e. Verify the `state` is set to `absent`.

This the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to ensure the absence of a global forwarder in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53
    ipadnsconfig:
      forwarders:
        - ip_address: 8.8.6.6
        - ip_address: 2001:4860:4860::8800
        port: 53
      action: member
      state: absent

6. Save the file.

7. Run the playbook:

   ```
   $ ansible-playbook -v -i inventory.file ensure-absence-of-a-global-forwarder.yml
   ```

Additional resources

- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsconfig` module in the README-dnsconfig.md Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of `ipadnsconfig` variables.

## 74.4. DNS FORWARD POLICIES IN IDM

IdM supports the **first** and **only** standard BIND forward policies, as well as the **none** IdM-specific forward policy.

**Forward first** *(default)*

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The **forward first** policy is the default policy, and it is suitable for optimizing DNS traffic.

**Forward only**

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND returns an error to the client. The **forward only** policy is recommended for environments with split DNS configuration.

**None** *(forwarding disabled)*

DNS queries are not forwarded with the **none** forwarding policy. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.
NOTE

You cannot use forwarding to combine data in IdM with data from other DNS servers. You can only forward queries for specific subzones of the primary zone in IdM DNS.

By default, the BIND service does not forward queries to another server if the queried DNS name belongs to a zone for which the IdM server is authoritative. In such a situation, if the queried DNS name cannot be found in the IdM database, the **NXDOMAIN** answer is returned. Forwarding is not used.

Example 74.1. Example Scenario

The IdM server is authoritative for the **test.example**. DNS zone. BIND is configured to forward queries to the DNS server with the **192.0.2.254** IP address.

When a client sends a query for the **nonexistent.test.example**. DNS name, BIND detects that the IdM server is authoritative for the **test.example**. zone and does not forward the query to the **192.0.2.254** server. As a result, the DNS client receives the **NXDomain** error message, informing the user that the queried domain does not exist.

74.5. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT THE FORWARD FIRST POLICY IS SET IN IDM DNS GLOBAL CONFIGURATION

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that global forwarding policy in IdM DNS is set to **forward first**.

If you use the **forward first** DNS forwarding policy, DNS queries are forwarded to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The forward first policy is the default policy. It is suitable for traffic optimization.

Prerequisites

- You have installed the **ansible-freeipa** package on the Ansible controller, the host on which you execute the procedure. For more information, see **Installing the ansible-freeipa package**.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure the **server.idm.example.com**, enter:
3. Make a copy of the `set-configuration.yml` Ansible playbook file. For example:

```bash
$ cp set-configuration.yml set-forward-policy-to-first.yml
```

4. Open the `set-forward-policy-to-first.yml` file for editing.

5. Adapt the file by setting the following variables in the `ipadnsconfig` task section:
   
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `forward_policy` variable to `first`.
     
   Delete all the other lines of the original playbook that are irrelevant. This is the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to set global forwarding policy to first
  hosts: ipaserver
  become: true

  tasks:
  - name: Set global forwarding policy to first.
    ipadnsconfig:
      ipaadmin_password: Secret123
      forward_policy: first
```

6. Save the file.

7. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file set-forward-policy-to-first.yml
   ```

Additional resources

- For more information on forwarding policy types available in IdM DNS, see [DNS forward policies in IdM](#).
- For more sample Ansible playbooks using the `ansible-freeipa ipadnsconfig` module, see the `README-dnsconfig.md` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `ipadnsconfig` variables.
- For more sample Ansible playbooks using the `ipadnsconfig` module, see the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory.

### 74.6. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT GLOBAL FORWARDERS ARE DISABLED IN IDM DNS

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that global forwarders are disabled in IdM DNS. The disabling is done by setting the `forward_policy` variable to `none`. 
Disabling global forwarders causes DNS queries not to be forwarded. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller, the host on which you execute the procedure. For more information, see Installing the ansible-freeipa package.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the disable-global-forwarders.yml Ansible playbook file. For example:

   ```
   $ cp disable-global-forwarders.yml disable-global-forwarders-copy.yml
   ```

4. Open the disable-global-forwarders-copy.yml file for editing.

5. Adapt the file by setting the following variables in the ipadnsconfig task section:

   - Set the ipaadmin_password variable to your IdM administrator password.
   - Set the forward_policy variable to none.

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

   ```
   ---
   - name: Playbook to disable global DNS forwarders
     hosts: ipaserver
     become: true

     tasks:
     - name: Disable global forwarders.
       ipadnsconfig:
         ipaadmin_password: Secret123
         forward_policy: none
   ```

6. Save the file.

7. Run the playbook:
74.7. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT SYNCHRONIZATION OF FORWARD AND REVERSE LOOKUP ZONES IS DISABLED IN IDM DNS

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that forward and reverse lookup zones are not synchronized in IdM DNS.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller, the host on which you execute the procedure. For more information, see Installing the ansible-freeipa package.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the disallow-reverse-sync.yml Ansible playbook file. For example:

```
$ cp disallow-reverse-sync.yml disallow-reverse-sync-copy.yml
```

4. Open the disallow-reverse-sync-copy.yml file for editing.

5. Adapt the file by setting the following variables in the ipadnsconfig task section:

- Set the ipaadmin_password variable to your IdM administrator password.
- Set the `allow_sync_ptr` variable to `no`. 
  This is the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to disallow reverse record synchronization 
  hosts: ipaserver 
  become: true 

  tasks: 
  - name: Disallow reverse record synchronization. 
    ipadnsconfig: 
      ipaadmin_password: Secret123 
      allow_sync_ptr: no 

6. Save the file.
7. Run the playbook:

   $ ansible-playbook -v -i inventory.file disallow-reverse-sync-copy.yml

Additional resources

- For more sample Ansible playbooks using the `ansible-freeipa ipadnsconfig` module, see the README-dnsconfig.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the `ipadnsconfig` variables.

- For more sample Ansible playbooks using the `ipadnsconfig` module, see the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory.
CHAPTER 75. MANAGING DNS ZONES IN IDM

As Identity Management (IdM) administrator, you can manage how IdM DNS zones work. The chapter describes the following topics and procedures:

- What DNS zone types are supported in IdM
  - How to add primary IdM DNS zones using the IdM Web UI
  - How to add primary IdM DNS zones using the IdM CLI
  - How to remove primary IdM DNS zones using the IdM Web UI
  - How to remove primary IdM DNS zones using the IdM CLI
- What DNS attributes you can configure in IdM
  - How you can configure these attributes in the IdM Web UI
  - How you can configure these attributes in the IdM CLI
- How zone transfers work in IdM
  - How you can allow zone transfers in the IdM Web UI
  - How you can allow zone transfers in the IdM CLI

Prerequisites

- DNS service is installed on the IdM server. For more information about how to install an IdM server with integrated DNS, see one of the following links:
  - Installing an IdM server: With integrated DNS, with an integrated CA as the root CA
  - Installing an IdM server: With integrated DNS, with an external CA as the root CA
  - Installing an IdM server: With integrated DNS, without a CA

75.1. SUPPORTED DNS ZONE TYPES

Identity Management (IdM) supports two types of DNS zones: primary and forward zones. This section describes these two types of zones and includes an example scenario for DNS forwarding.

NOTE

This guide uses the BIND terminology for zone types which is different from the terminology used for Microsoft Windows DNS. Primary zones in BIND serve the same purpose as forward lookup zones and reverse lookup zones in Microsoft Windows DNS. Forward zones in BIND serve the same purpose as conditional forwarders in Microsoft Windows DNS.

Primary DNS zones

Primary DNS zones contain authoritative DNS data and can accept dynamic DNS updates. This behavior is equivalent to the type master setting in standard BIND configuration. You can manage primary zones using the ipa dnszone-* commands.
In compliance with standard DNS rules, every primary zone must contain **start of authority** (SOA) and **nameserver** (NS) records. IdM generates these records automatically when the DNS zone is created, but you must copy the NS records manually to the parent zone to create proper delegation.

In accordance with standard BIND behavior, queries for names for which the server is not authoritative are forwarded to other DNS servers. These DNS servers, so called forwarders, may or may not be authoritative for the query.

**Example 75.1. Example scenario for DNS forwarding**

The IdM server contains the `test.example.` primary zone. This zone contains an NS delegation record for the `sub.test.example.` name. In addition, the `test.example.` zone is configured with the `192.0.2.254` forwarder IP address for the `sub.text.example` subzone.

A client querying the name `nonexistent.test.example.` receives the `NXDomain` answer, and no forwarding occurs because the IdM server is authoritative for this name.

On the other hand, querying for the `host1.sub.test.example.` name is forwarded to the configured forwarder `192.0.2.254` because the IdM server is not authoritative for this name.

**Forward DNS zones**

From the perspective of IdM, forward DNS zones do not contain any authoritative data. In fact, a forward “zone” usually only contains two pieces of information:

- A domain name
- The IP address of a DNS server associated with the domain

All queries for names belonging to the domain defined are forwarded to the specified IP address. This behavior is equivalent to the **type forward** setting in standard BIND configuration. You can manage forward zones using the `ipa dnsforwardzone-*` commands.

Forward DNS zones are especially useful in the context of IdM-Active Directory (AD) trusts. If the IdM DNS server is authoritative for the `idm.example.com` zone and the AD DNS server is authoritative for the `ad.example.com` zone, then `ad.example.com` is a DNS forward zone for the `idm.example.com` primary zone. That means that when a query comes from an IdM client for the IP address of `somehost.ad.example.com`, the query is forwarded to an AD domain controller specified in the `ad.example.com` IdM DNS forward zone.

**75.2. ADDING A PRIMARY DNS ZONE IN IDM WEB UI**

This section describes how to add a primary DNS zone using the Identity Management (IdM) Web UI.

**Prerequisites**

- You are logged in as IdM administrator.

**Procedure**

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.
2. Click **Add** at the top of the list of all zones.

3. Provide the zone name.

4. Click **Add**.

### 75.3. ADDING A PRIMARY DNS ZONE IN IDM CLI

This section describes how to add a primary DNS zone in the Identity Management (IdM) command-line interface (CLI).

**Prerequisites**

- You are logged in as IdM administrator.

**Procedure**

- The `ipa dnszone-add` command adds a new zone to the DNS domain. Adding a new zone requires you to specify the name of the new subdomain. You can pass the subdomain name directly with the command:

  ```
  $ ipa dnszone-add newzone.idm.example.com
  ```

  If you do not pass the name to `ipa dnszone-add`, the script prompts for it automatically.
Additional resources

- The `ipa dnszone-add` command also accepts various command-line options. For a complete list of these options, run the `ipa dnszone-add --help` command.

### 75.4. REMOVING A PRIMARY DNS ZONE IN IDM WEB UI

This section describes how to remove a primary DNS zone from Identity Management (IdM) using the IdM Web UI.

#### Prerequisites

- You are logged in as IdM administrator.

#### Procedure

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.
2. Select the check box by the zone name and click **Delete**.

Figure 75.3. Removing a primary DNS Zone

3. In the **Remove DNS zones** dialog window, confirm that you want to delete the selected zone.

### 75.5. REMOVING A PRIMARY DNS ZONE IN IDM CLI

This section describes how to remove a primary DNS zone from Identity Management (IdM) using the IdM command-line interface (CLI).

#### Prerequisites

- You are logged in as IdM administrator.

#### Procedure

- To remove a primary DNS zone, enter the `ipa dnszone-del` command, followed by the name of the zone you want to remove. For example:

  ```
  $ ipa dnszone-del idm.example.com
  ```

### 75.6. DNS CONFIGURATION PRIORITIES

You can configure many DNS configuration options on the following levels. Each level has a different
You can configure many DNS configuration options on the following levels. Each level has a different priority.

**Zone-specific configuration**

The level of configuration specific for a particular zone defined in IdM has the highest priority. You can manage zone-specific configuration by using the `ipa dnszone-*` and `ipa dnsforwardzone-*` commands.

**Per-server configuration**

You are asked to define per-server forwarders during the installation of an IdM server. You can manage per-server forwarders by using the `ipa dnsserver-*` commands. If you do not want to set a per-server forwarder when installing a replica, you can use the `--no-forwarder` option.

**Global DNS configuration**

If no zone-specific configuration is defined, IdM uses global DNS configuration stored in LDAP. You can manage global DNS configuration using the `ipa dnsconfig-*` commands. Settings defined in global DNS configuration are applied to all IdM DNS servers.

**Configuration in /etc/named.conf**

Configuration defined in the `/etc/named.conf` file on each IdM DNS server has the lowest priority. It is specific for each server and must be edited manually. The `/etc/named.conf` file is usually only used to specify DNS forwarding to a local DNS cache. Other options are managed using the commands for zone-specific and global DNS configuration mentioned above.

You can configure DNS options on multiple levels at the same time. In such cases, configuration with the highest priority takes precedence over configuration defined at lower levels.

**Additional resources**

- The [Priority order of configuration](#) section in *Per Server Config in LDAP*

## 75.7. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES

Identity Management (IdM) creates a new zone with certain default configuration, such as the refresh periods, transfer settings, or cache settings. In *IdM DNS zone attributes*, you can find the attributes of the default zone configuration that you can modify using one of the following options:

- The `dnszone-mod` command in the command-line interface (CLI). For more information, see *Editing the configuration of a primary DNS zone in IdM CLI*.

- The IdM Web UI. For more information, see *Editing the configuration of a primary DNS zone in IdM Web UI*.

- An Ansible playbook that uses the `ipadnszone` module. For more information, see *Managing DNS zones in IdM*.

Along with setting the actual information for the zone, the settings define how the DNS server handles the *start of authority* (SOA) record entries and how it updates its records from the DNS name server.

**Table 75.1. IdM DNS zone attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Command-Line Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoritative name server</td>
<td><code>--name-server</code></td>
<td>Sets the domain name of the primary DNS name server, also known as SOA MNAME. By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using <code>--name-server</code> is ignored.</td>
</tr>
<tr>
<td>Administrator e-mail address</td>
<td><code>--admin-email</code></td>
<td>Sets the email address to use for the zone administrator. This defaults to the root account on the host.</td>
</tr>
<tr>
<td>SOA serial</td>
<td><code>--serial</code></td>
<td>Sets a serial number in the SOA record. Note that IdM sets the version number automatically and users are not expected to modify it.</td>
</tr>
<tr>
<td>SOA refresh</td>
<td><code>--refresh</code></td>
<td>Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.</td>
</tr>
<tr>
<td>SOA retry</td>
<td><code>--retry</code></td>
<td>Sets the time, in seconds, to wait before retrying a failed refresh operation.</td>
</tr>
<tr>
<td>SOA expire</td>
<td><code>--expire</code></td>
<td>Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.</td>
</tr>
<tr>
<td>SOA minimum</td>
<td><code>--minimum</code></td>
<td>Sets the time to live (TTL) value in seconds for negative caching according to RFC 2308.</td>
</tr>
<tr>
<td>SOA time to live</td>
<td><code>--ttl</code></td>
<td>Sets TTL in seconds for records at zone apex. In zone example.com, for instance, all records (A, NS, or SOA) under name example.com are configured, but no other domain names, like test.example.com, are affected.</td>
</tr>
<tr>
<td>Default time to live</td>
<td><code>--default-ttl</code></td>
<td>Sets the default time to live (TTL) value in seconds for negative caching for all values in a zone that never had an individual TTL value set before. Requires a restart of the named-pkcs11 service on all IdM DNS servers after changes to take effect.</td>
</tr>
<tr>
<td>BIND update policy</td>
<td><code>--update-policy</code></td>
<td>Sets the permissions allowed to clients in the DNS zone.</td>
</tr>
<tr>
<td>Dynamic update</td>
<td>`--dynamic-update=TRUE</td>
<td>FALSE`</td>
</tr>
<tr>
<td>Attribute</td>
<td>Command-Line Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Allow transfer</td>
<td>--allow-transfer=string</td>
<td>Gives a list of IP addresses or network names which are allowed to transfer the given zone, separated by semicolons (;). Zone transfers are disabled by default. The default <strong>--allow-transfer</strong> value is <strong>none</strong>.</td>
</tr>
<tr>
<td>Allow query</td>
<td>--allow-query</td>
<td>Gives a list of IP addresses or network names which are allowed to issue DNS queries, separated by semicolons (;).</td>
</tr>
<tr>
<td>Allow PTR sync</td>
<td>--allow-sync-ptr=1</td>
<td>0</td>
</tr>
<tr>
<td>Zone forwarders</td>
<td>--forwarder=IP_add ress</td>
<td>Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the IdM domain. To specify multiple forwarders, use the option multiple times.</td>
</tr>
<tr>
<td>Forward policy</td>
<td>--forward-policy=none</td>
<td>only</td>
</tr>
</tbody>
</table>

### 75.8. Editing the Configuration of a Primary DNS Zone in IDM Web UI

This section describes how to edit the configuration attributes of a primary Identity Management (IdM) DNS using the IdM Web UI.

**Prerequisites**

- You are logged in as IdM administrator.

**Procedure**

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.

---

![Figure 75.4. DNS primary zones management](image-url)
2. In the **DNS Zones** section, click on the zone name in the list of all zones to open the DNS zone page.

![Figure 75.5. Editing a primary zone](image)

3. Click **Settings**.

![Figure 75.6. The Settings tab in the primary zone edit page](image)

4. Change the zone configuration as required.
   For information about the available settings, see [IdM DNS zone attributes](#).

5. Click **Save** to confirm the new configuration.

   **NOTE**
   If you are changing the default time to live (TTL) of a zone, restart the `named-pkcs11` service on all IdM DNS servers to make the changes take effect. All other settings are automatically activated immediately.

### 75.9. Editing the Configuration of a Primary DNS Zone in IdM CLI

This section describes how to edit the configuration of a primary DNS zone using the Identity Management (IdM) command-line interface (CLI).

**Prerequisites**
You are logged in as IdM administrator.

Procedure

- To modify an existing primary DNS zone, use the `ipa dnszone-mod` command. For example, to set the time to wait before retrying a failed refresh operation to 1800 seconds:

  ```
  $ ipa dnszone-mod --retry 1800
  ```

  For more information about the available settings and their corresponding CLI options, see IdM DNS zone attributes.

  If a specific setting does not have a value in the DNS zone entry you are modifying, the `ipa dnszone-mod` command adds the value. If the setting does not have a value, the command overwrites the current value with the specified value.

  **NOTE**

  If you are changing the default time to live (TTL) of a zone, restart the `named-pkcs11` service on all IdM DNS servers to make the changes take effect. All other settings are automatically activated immediately.

Additional resources

- For detailed information about `ipa dnszone-mod` and its options, run the `ipa dnszone-mod --help` command.

75.10. ZONE TRANSFERS IN IDM

This section describes how zone transfers work in an Identity Management (IdM) deployment that has integrated DNS.

Name servers maintain authoritative data for their zones. If you make changes to the zone on a DNS server that is authoritative for the `zone A` DNS zone, you must distribute the changes among the other name servers in the IdM DNS domain that are outside `zone A`. A zone transfer copies all resource records from one name server to another.

**IMPORTANT**

The IdM-integrated DNS can be written to by different servers simultaneously. The Start of Authority (SOA) serial numbers in IdM zones are not synchronized among the individual IdM DNS servers. For this reason, configure your DNS servers outside the to-be-transferred zone to only use one specific DNS server inside the to-be-transferred zone. This prevents zone transfer failures caused by non-synchronized SOA serial numbers.

IdM supports zone transfers according to the RFC 5936 (AXFR) and RFC 1995 (IXFR) standards.

Additional resources

- For more information on how to proceed to enable zone transfers in the IdM Web UI, see Enabling zone transfers in IdM Web UI.
For more information on how to proceed to enable zone transfers in the IdM CLI, see Enabling zone transfers in IdM CLI.

75.11. ENABLING ZONE TRANSFERS IN IDM WEB UI

This section describes how to enable zone transfers in Identity Management (IdM) using the IdM Web UI.

Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.
2. Click **Settings**.
3. Under **Allow transfer**, specify the name servers to which you want to transfer the zone records.

![Figure 75.7. Enabling zone transfers](image)

4. Click **Save** at the top of the DNS zone page to confirm the new configuration.

75.12. ENABLING ZONE TRANSFERS IN IDM CLI

This section describes how to enable zone transfers in Identity Management (IdM) using the IdM command-line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.
- You have root access to the secondary DNS servers.

Procedure

- To enable zone transfers in the **BIND** service, enter the `ipa dnszone-mod` command, and specify the list of name servers that are outside the to-be-transferred zone to which the zone records will be transferred using the `--allow-transfer` option. For example:

```bash
$ ipa dnszone-mod --allow-transfer=192.0.2.1;198.51.100.1;203.0.113.1
idm.example.com
```
Verification steps

1. SSH to one of the DNS servers to which zone transfer has been enabled:

   ```
   $ ssh 192.0.2.1
   ```

2. Transfer the IdM DNS zone using a tool such as the `dig` utility:

   ```
   # dig @ipa-server zone_name AXFR
   ```

If the command returns no error, you have successfully enabled zone transfer for `zone_name`.

75.13. ADDITIONAL RESOURCES

- For more information about how to use Red Hat Ansible Engine to manage IdM DNS zones, see [Using Ansible playbooks to manage IdM DNS zones](#).
CHAPTER 76. USING ANSIBLE PLAYBOOKS TO MANAGE IDM DNS ZONES

As Identity Management (IdM) administrator, you can manage how IdM DNS zones work using the `dnszone` module available in the `ansible-freeipa` package. The chapter describes the following topics and procedures:

- What DNS zone types are supported in IdM
- What DNS attributes you can configure in IdM
- How to use an Ansible playbook to create a primary zone in IdM DNS
- How to use an Ansible playbook to ensure the presence of a primary IdM DNS zone with multiple variables
- How to use an Ansible playbook to ensure the presence of a zone for reverse DNS lookup when an IP address is given

Prerequisites

- DNS service is installed on the IdM server. For more information about how to use Red Hat Ansible Engine to install an IdM server with integrated DNS, see Installing an Identity Management server using an Ansible playbook.

76.1. SUPPORTED DNS ZONE TYPES

Identity Management (IdM) supports two types of DNS zones: `primary` and `forward` zones. This section describes these two types of zones and includes an example scenario for DNS forwarding.

**NOTE**

This guide uses the BIND terminology for zone types which is different from the terminology used for Microsoft Windows DNS. Primary zones in BIND serve the same purpose as forward lookup zones and reverse lookup zones in Microsoft Windows DNS. Forward zones in BIND serve the same purpose as conditional forwarders in Microsoft Windows DNS.

Primary DNS zones

Primary DNS zones contain authoritative DNS data and can accept dynamic DNS updates. This behavior is equivalent to the `type master` setting in standard BIND configuration. You can manage primary zones using the `ipa dnszone-*` commands.

In compliance with standard DNS rules, every primary zone must contain `start of authority` (SOA) and `nameserver` (NS) records. IdM generates these records automatically when the DNS zone is created, but you must copy the NS records manually to the parent zone to create proper delegation.

In accordance with standard BIND behavior, queries for names for which the server is not authoritative are forwarded to other DNS servers. These DNS servers, so called forwarders, may or may not be authoritative for the query.

**Example 76.1. Example scenario for DNS forwarding**
The IdM server contains the `test.example` primary zone. This zone contains an NS delegation record for the `sub.test.example` name. In addition, the `test.example` zone is configured with the `192.0.2.254` forwarder IP address for the `sub.text.example` subzone.

A client querying the name `nonexistent.test.example` receives the `NXDomain` answer, and no forwarding occurs because the IdM server is authoritative for this name.

On the other hand, querying for the `host1.sub.test.example` name is forwarded to the configured forwarder `192.0.2.254` because the IdM server is not authoritative for this name.

**Forward DNS zones**

From the perspective of IdM, forward DNS zones do not contain any authoritative data. In fact, a forward "zone" usually only contains two pieces of information:

- A domain name
- The IP address of a DNS server associated with the domain

All queries for names belonging to the domain defined are forwarded to the specified IP address. This behavior is equivalent to the **type forward** setting in standard BIND configuration. You can manage forward zones using the `ipa dnsforwardzone-*` commands.

Forward DNS zones are especially useful in the context of IdM-Active Directory (AD) trusts. If the IdM DNS server is authoritative for the `idm.example.com` zone and the AD DNS server is authoritative for the `ad.example.com` zone, then `ad.example.com` is a DNS forward zone for the `idm.example.com` primary zone. That means that when a query comes from an IdM client for the IP address of `somehost.ad.example.com`, the query is forwarded to an AD domain controller specified in the `ad.example.com` IdM DNS forward zone.

### 76.2. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES

Identity Management (IdM) creates a new zone with certain default configuration, such as the refresh periods, transfer settings, or cache settings. In IdM DNS zone attributes, you can find the attributes of the default zone configuration that you can modify using one of the following options:

- The `dnszone-mod` command in the command-line interface (CLI). For more information, see Editing the configuration of a primary DNS zone in IdM CLI.
- The IdM Web UI. For more information, see Editing the configuration of a primary DNS zone in IdM Web UI.
- An Ansible playbook that uses the `ipadnszone` module. For more information, see Managing DNS zones in IdM.

Along with setting the actual information for the zone, the settings define how the DNS server handles the **start of authority** (SOA) record entries and how it updates its records from the DNS name server.

**Table 76.1. IdM DNS zone attributes**
<table>
<thead>
<tr>
<th>Attribute</th>
<th>ansible-freeipa variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authoritative name server</td>
<td>name_server</td>
<td>Sets the domain name of the primary DNS name server, also known as SOA MNAME.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using \texttt{--name-server} is ignored.</td>
</tr>
<tr>
<td>Administrator email address</td>
<td>admin_email</td>
<td>Sets the email address to use for the zone administrator. This defaults to the root account on the host.</td>
</tr>
<tr>
<td>SOA serial</td>
<td>serial</td>
<td>Sets a serial number in the SOA record. Note that IdM sets the version number automatically and users are not expected to modify it.</td>
</tr>
<tr>
<td>SOA refresh</td>
<td>refresh</td>
<td>Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.</td>
</tr>
<tr>
<td>SOA retry</td>
<td>retry</td>
<td>Sets the time, in seconds, to wait before retrying a failed refresh operation.</td>
</tr>
<tr>
<td>SOA expire</td>
<td>expire</td>
<td>Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.</td>
</tr>
<tr>
<td>SOA minimum</td>
<td>minimum</td>
<td>Sets the time to live (TTL) value in seconds for negative caching according to RFC 2308.</td>
</tr>
<tr>
<td>SOA time to live</td>
<td>ttl</td>
<td>Sets TTL in seconds for records at zone apex. In zone example.com, for instance, all records (A, NS, or SOA) under name example.com are configured, but no other domain names, like test.example.com, are affected.</td>
</tr>
<tr>
<td>Default time to live</td>
<td>default_ttl</td>
<td>Sets the default time to live (TTL) value in seconds for negative caching for all values in a zone that never had an individual TTL value set before. Requires a restart of the named-pkcs11 service on all IdM DNS servers after changes to take effect.</td>
</tr>
<tr>
<td>BIND update policy</td>
<td>update_policy</td>
<td>Sets the permissions allowed to clients in the DNS zone.</td>
</tr>
<tr>
<td>Dynamic update</td>
<td>dynamic_update=TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that if this is set to false, IdM client machines will not be able to add or update their IP address.</td>
</tr>
<tr>
<td>Allow transfer</td>
<td>allow_transfer=string</td>
<td>Gives a list of IP addresses or network names which are allowed to transfer the given zone, separated by semicolons (;).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zone transfers are disabled by default. The default allow_transfer value is none.</td>
</tr>
</tbody>
</table>
### Attribute | ansible-freeipa variable | Description
--- | --- | ---
Allow query | allow_query | Gives a list of IP addresses or network names which are allowed to issue DNS queries, separated by semicolons (;).
Allow PTR sync | allow_sync_ptr=1|0 | Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.
Zone forwarders | forwarder=IP_address | Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the IdM domain.
|  |  | To specify multiple forwarders, use the option multiple times.
Forward policy | forward_policy=none|only|first | Specifies the forward policy. For information about the supported policies, see DNS forward policies in IdM.

### Additional resources
- You can see more definitions of the attributes of the ansible-freeipa ipadnszone module in the README-dnszone.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory.

### 76.3. USING ANSIBLE TO CREATE A PRIMARY ZONE IN IDM DNS

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that a primary DNS zone exists. In the example used in the procedure below, an IdM administrator ensures the presence of the zone.idm.example.com DNS zone.

#### Prerequisites
- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

#### Procedure
1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnszone directory:
   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
   ```
2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:
   ```bash
   [ipaserver]
   server.idm.example.com
   ```
3. Make a copy of the `dnszone-present.yml` Ansible playbook file. For example:

```
$ cp dnszone-present.yml dnszone-present-copy.yml
```

4. Open the `dnszone-present-copy.yml` file for editing.

5. Adapt the file by setting the following variables in the `ipadnszone` task section:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `zone_name` variable to `zone.idm.example.com`.

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

```
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure zone is present.
    ipadnszone:
      ipaadmin_password: Secret123
      zone_name: zone.idm.example.com
      state: present
```

6. Save the file.

7. Run the playbook:

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

Additional resources

- For more information on DNS zone, see Supported DNS zone types.

- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnszone` module in the `README-dnszone.md` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `ipadnszone` variables.

- You can see sample Ansible playbooks for the `ipadnszone` module in the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory.

76.4. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A PRIMARY DNS ZONE IN IDM WITH MULTIPLE VARIABLES

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that a primary DNS zone exists. In the example used in the procedure below, an IdM administrator ensures the presence of the `zone.idm.example.com` DNS zone. The Ansible playbook configures multiple parameters of the zone.

Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
   ```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `dnszone-all-params.yml` Ansible playbook file. For example:

   ```
   $ cp dnszone-all-params.yml dnszone-all-params-copy.yml
   ```

4. Open the `dnszone-all-params-copy.yml` file for editing.

5. Adapt the file by setting the following variables in the `ipadnszone` task section:

   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `zone_name` variable to `zone.idm.example.com`.
   - Set the `allow_sync_ptr` variable to true if you want to allow the synchronization of forward and reverse records, that is the synchronization of A and AAAA records with PTR records.
   - Set the `dynamic_update` variable to true to enable IdM client machines to add or update their IP addresses.
   - Set the `dnssec` variable to true to allow inline DNSSEC signing of records in the zone.
   - Set the `allow_transfer` variable to the IP addresses of secondary name servers in the zone.
   - Set the `allow_query` variable to the IP addresses or networks that are allowed to issue queries.
   - Set the `forwarders` variable to the IP addresses of global forwarders.
   - Set the `serial` variable to the SOA record serial number.
   - Define the `refresh`, `retry`, `expire`, `minimum`, `ttl`, and `default_ttl` values for DNS records in the zone.
   - Define the NSEC3PARAM record for the zone using the `nsec3param_rec` variable.
   - Set the `skip_overlap_check` variable to true to force DNS creation even if it overlaps with an existing zone.
   - Set the `skip_nameserver_check` to true to force DNS zone creation even if the nameserver is not resolvable.

   This is the modified Ansible playbook file for the current example:
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure zone is present.
    ipadnszone:
      ipaadmin_password: Secret123
      zone_name: zone.idm.example.com
      allow_sync_ptr: true
      dynamic_update: true
      dnssec: true
      allow_transfer:
        - 1.1.1.1
        - 2.2.2.2
      allow_query:
        - 1.1.1.1
        - 2.2.2.2
      forwarders:
        - ip_address: 8.8.8.8
        - ip_address: 8.8.4.4
      port: 52
      serial: 1234
      refresh: 3600
      retry: 900
      expire: 1209600
      minimum: 3600
      ttl: 60
      default_ttl: 90
      name_server: server.idm.example.com.
      admin_email: admin.admin@idm.example.com
      nsec3param_rec: "1 7 100 0123456789abcdef"
      skip_overlap_check: true
      skip_nameserver_check: true
      state: present

6. Save the file.

7. Run the playbook:

   ```
   $ ansible-playbook -v -i inventory.file dnszone-all-params-copy.yml
   ```

Additional resources

- For more information on DNS zone, see Supported DNS zone types.
- For more information on what DNS zone attributes you can configure in IdM, see Configuration attributes of primary IdM DNS zones.
- You can see more sample Ansible playbooks for the ansible-freeipa ipadnszone module in the README-dnszone.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the ipadnszone variables.
You can see sample Ansible playbooks for the `ipadnszone` module in the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory.

76.5. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A ZONE FOR REVERSE DNS LOOKUP WHEN AN IP ADDRESS IS GIVEN

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that a reverse DNS zone exists. In the example used in the procedure below, an IdM administrator ensures the presence of a reverse DNS lookup zone using the IP address and prefix length of an IdM host.

Providing the prefix length of the IP address of your DNS server using the `name_from_ip` variable allows you to control the zone name. If you do not state the prefix length, the system queries DNS servers for zones and, based on the `name_from_ip` value of `192.168.1.2`, the query can return any of the following DNS zones:

- `1.168.192.in-addr.arpa`
- `168.192.in-addr.arpa`
- `192.in-addr.arpa`

Because the zone returned by the query might not be what you expect, `name_from_ip` can only be used with the `state` option set to `present` to prevent accidental removals of zones.

Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory:
   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
   ```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:
   ```
   [ipaserver]
   server.idm.example.com
   ```

3. Make a copy of the `dnszone-reverse-from-ip.yml` Ansible playbook file. For example:
   ```bash
   $ cp dnszone-reverse-from-ip.yml dnszone-reverse-from-ip-copy.yml
   ```

4. Open the `dnszone-reverse-from-ip-copy.yml` file for editing.

5. Adapt the file by setting the following variables in the `ipadnszone` task section:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
- Set the `name_from_ip` variable to the IP of your IdM nameserver, and provide its prefix length.

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

```yaml
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure zone for reverse DNS lookup is present.
      ipadnszone:
        ipaadmin_password: Secret123
        name_from_ip: 192.168.1.2/24
        state: present
        register: result
    - name: Display inferred zone name.
      debug:
        msg: "Zone name: {{ result.dnszone.name }}"
```

The playbook creates a zone for reverse DNS lookup from the 192.168.1.2 IP address and its prefix length of 24. Next, the playbook displays the resulting zone name.

6. Save the file.

7. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file dnszone-reverse-from-ip-copy.yml
   ```

Additional resources

- For more information on DNS zone, see [Supported DNS zone types](#).

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnszone module in the README-dnszone.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the ipadnszone variables.

- You can see sample Ansible playbooks for the ipadnszone module in the /usr/share/doc/ansible-freeipa/playbooks/dnszone directory.
CHAPTER 77. MANAGING DNS LOCATIONS IN IDM

As Identity Management (IdM) administrator, you can manage Identity Management (IdM) DNS locations by using the IdM Web UI and IdM command-line interface (CLI). The chapter describes the following topics and procedures:

- DNS-based service discovery
- Deployment considerations for DNS locations
- DNS time to live (TTL)
- Creating DNS locations using the IdM Web UI
- Creating DNS locations using the IdM CLI
- Assigning an IdM server to a DNS location using the IdM Web UI
- Assigning an IdM server to a DNS location using the IdM CLI
- Configuring an IdM client to use IdM servers in the same location

77.1. DNS-BASED SERVICE DISCOVERY

DNS-based service discovery is a process in which a client uses the DNS protocol to locate servers in a network that offer a specific service, such as LDAP or Kerberos. One typical type of operation is to allow clients to locate authentication servers within the closest network infrastructure, because they provide a higher throughput and lower network latency, lowering overall costs.

The major advantages of service discovery are:

- No need for clients to be explicitly configured with names of nearby servers.
- DNS servers are used as central providers of policy. Clients using the same DNS server have access to the same policy about service providers and their preferred order.

In an Identity Management (IdM) domain, DNS service records (SRV records) exist for LDAP, Kerberos, and other services. For example, the following command queries the DNS server for hosts providing a TCP-based Kerberos service in an IdM DNS domain:

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
0 100 88 idmserver-01.idm.example.com.
0 100 88 idmserver-02.idm.example.com.
```

The output contains the following information:

- **0** (priority): Priority of the target host. A lower value is preferred.
- **100** (weight). Specifies a relative weight for entries with the same priority. For further information, see RFC 2782, section 3.
- **88** (port number): Port number of the service.
- Canonical name of the host providing the service.
In the example, the two host names returned have the same priority and weight. In this case, the client uses a random entry from the result list.

When the client is, instead, configured to query a DNS server that is configured in a DNS location, the output differs. For IdM servers that are assigned to a location, tailored values are returned. In the example below, the client is configured to query a DNS server in the location `germany`.

Example 77.2. DNS location-based results

```bash
$ dig -t SRV +short _kerberos._tcp.idm.example.com _kerberos._tcp.germany._locations.idm.example.com.
0 100 88 idmserver-01.idm.example.com.
50 100 88 idmserver-02.idm.example.com.
```

The IdM DNS server automatically returns a DNS alias (CNAME) pointing to a DNS location specific SRV record which prefers local servers. This CNAME record is shown in the first line of the output. In the example, the host `idmserver-01.idm.example.com` has the lowest priority value and is therefore preferred. The `idmserver-02.idm.example.com` has a higher priority and thus is used only as backup for cases when the preferred host is unavailable.

### 77.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS

Identity Management (IdM) can generate location-specific service (SRV) records when using the integrated DNS. Because each IdM DNS server generates location-specific SRV records, you have to install at least one IdM DNS server in each DNS location.

The client’s affinity to a DNS location is only defined by the DNS records received by the client. For this reason, you can combine IdM DNS servers with non-IdM DNS consumer servers and recursors if the clients doing DNS service discovery resolve location-specific records from IdM DNS servers.

In the majority of deployments with mixed IdM and non-IdM DNS services, DNS recursors select the closest IdM DNS server automatically by using round-trip time metrics. Typically, this ensures that clients using non-IdM DNS servers are getting records for the nearest DNS location and thus use the optimal set of IdM servers.

### 77.3. DNS TIME TO LIVE (TTL)

Clients can cache DNS resource records for an amount of time that is set in the zone’s configuration. Because of this caching, a client might not be able to receive the changes until the time to live (TTL) value expires. The default TTL value in Identity Management (IdM) is 1 day.

If your client computers roam between sites, you should adapt the TTL value for your IdM DNS zone. Set the value to a lower value than the time clients need to roam between sites. This ensures that cached DNS entries on the client expire before they reconnect to another site and thus query the DNS server to refresh location-specific SRV records.

Additional resources

- For further information how to modify the default TTL of a DNS zone, see Configuration attributes of primary IdM DNS zones.
77.4. CREATING DNS LOCATIONS USING THE IDM WEB UI

You can use DNS locations to increase the speed of communication between Identity Management (IdM) clients and servers. This section describes how to create a DNS location using the IdM Web UI.

Prerequisites

- Your IdM deployment has integrated DNS.
- You have a permission to create DNS locations in IdM. For example, you are logged in as IdM admin.

Procedure

1. Open the IPA Server tab.
2. Select Topology subtab.
3. Click IPA Locations in the navigation bar.
4. Click Add at the top of the locations list.
5. Fill in the location name.
6. Click the Add button to save the location.
7. Optional: Repeat the steps to add further locations.

Additional resources

- To configure specific servers for the IdM locations you have added, see Assigning an IdM server to a DNS location using the IdM Web UI.
- For more information on how to use an Ansible playbook to execute the procedure, see Using Ansible to ensure an IdM location is present.

77.5. CREATING DNS LOCATIONS USING THE IDM CLI

You can use DNS locations to increase the speed of communication between Identity Management (IdM) clients and servers. This section describes how to create DNS locations using the ipa location-add command in the IdM command-line interface (CLI).

Prerequisites

- Your IdM deployment has integrated DNS.
- You have a permission to create DNS locations in IdM. For example, you are logged in as IdM admin.

Procedure

1. For example, to create a new location germany, enter:

   $ ipa location-add germany

----------------------------
Added IPA location "germany"
----------------------------
Location name: germany

2. Optional: Repeat the step to add further locations.

Additional resources

- To configure specific servers for the IdM locations you have added, see Assigning an IdM Server to a DNS Location using the IdM CLI.
- For more information on how to use an Ansible playbook to execute the procedure, see Using Ansible to ensure an IdM location is present.

77.6. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM WEB UI

You can use Identity Management (IdM) DNS locations to increase the speed of communication between IdM clients and servers. This section describes how to assign IdM servers to DNS locations using the IdM Web UI.

Prerequisites

- Your IdM deployment has integrated DNS.
- You are logged in as a user with a permission to assign a server to a DNS location, for example the IdM admin user.
- You have root access to the host that you want to assign a DNS location to.
- You have created the IdM DNS locations to which you want to assign servers.

Procedure

1. Open the IPA Server tab.
2. Select the Topology subtab.
3. Click IPA Servers in the navigation.
4. Click on the IdM server name.
5. Select a DNS location, and optionally set a service weight:
6. Click **Save**.

7. In the command-line interface (CLI) of the host you assigned in the previous steps the DNS location to, restart the `named-pkcs11` service:

   ```
   [root@idmserver-01 ~]# systemctl restart named-pkcs11
   ```

8. Optional: Repeat the steps to assign DNS locations to further IdM servers.

**Additional resources**

- To continue, see [Configuring an IdM client to use IdM servers in the same location](#).

### 77.7. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM CLI

You can use Identity Management (IdM) DNS locations to increase the speed of communication between IdM clients and servers. This section describes how to assign IdM servers to DNS locations using the IdM command-line interface (CLI).

**Prerequisites**

- Your IdM deployment has integrated DNS.

- You are logged in as a user with a permission to assign a server to a DNS location, for example the IdM admin user.

- You have root access to the host that you want to assign a DNS location to.

- You have created the IdM DNS locations to which you want to assign servers.

**Procedure**

Figure 77.1. Assigning a server to a DNS location

<table>
<thead>
<tr>
<th>IPA Server: idmserver-01.idm.example.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Refresh, Revert, Save]</td>
</tr>
<tr>
<td><strong>Server name</strong></td>
</tr>
<tr>
<td><strong>Min domain level</strong></td>
</tr>
<tr>
<td><strong>Max domain level</strong></td>
</tr>
<tr>
<td><strong>Managed suffixes</strong></td>
</tr>
<tr>
<td><strong>Location</strong></td>
</tr>
<tr>
<td><strong>Service weight</strong></td>
</tr>
</tbody>
</table>

...
1. Optional: List all configured DNS locations:

```
[root@server ~]# ipa location-find
-----------------------
2 IPA locations matched
-----------------------
Location name: australia
Location name: germany
-----------------------
Number of entries returned: 2
-----------------------
```

2. Assign the server to the DNS location. For example, to assign the location `germany` to the server `idmserver-01.idm.example.com`, run:

```
# ipa server-mod idmserver-01.idm.example.com --location=germany
ipa: WARNING: Service named-pkcs11.service requires restart on IPA server idmserver-01.idm.example.com to apply configuration changes.
-----------------------
Modified IPA server "idmserver-01.idm.example.com"
-----------------------
Servername: idmserver-01.idm.example.com
Min domain level: 0
Max domain level: 1
Location: germany
Enabled server roles: DNS server, NTP server
```

3. Restart the `named-pkcs11` service on the host you assigned in the previous steps the DNS location to:

```
# systemctl restart named-pkcs11
```

4. Optional: Repeat the steps to assign DNS locations to further IdM servers.

Additional resources

- To continue, see Configuring an IdM client to use IdM servers in the same location.

77.8. CONFIGURING AN IDM CLIENT TO USE IDM SERVERS IN THE SAME LOCATION

Identity Management (IdM) servers are assigned to DNS locations as described in Assigning an IdM server to a DNS location using the IdM Web UI. Now you can configure the clients to use a DNS server that is in the same location as the IdM servers:

- If a **DHCP** server assigns the DNS server IP addresses to the clients, configure the **DHCP** service. For further details about assigning a DNS server in your **DHCP** service, see the **DHCP** service documentation.

- If your clients do not receive the DNS server IP addresses from a **DHCP** server, manually set the IPs in the client’s network configuration. For further details about configuring the network on Red Hat Enterprise Linux, see the Configuring Network Connection Settings section in the Red Hat Enterprise Linux Networking Guide.
NOTE

If you configure the client to use a DNS server that is assigned to a different location, the client contacts IdM servers in both locations.

Example 77.3. Different name server entries depending on the location of the client

The following example shows different name server entries in the `/etc/resolv.conf` file for clients in different locations:

Clients in Prague:

nameserver 10.10.0.1
nameserver 10.10.0.2

Clients in Paris:

nameserver 10.50.0.1
nameserver 10.50.0.3

Clients in Oslo:

nameserver 10.30.0.1

Clients in Berlin:

nameserver 10.30.0.1

If each of the DNS servers is assigned to a location in IdM, the clients use the IdM servers in their location.

77.9. ADDITIONAL RESOURCES

- For more information about how to use Red Hat Ansible Engine to manage IdM DNS locations, see Using Ansible to manage DNS locations in IdM.
CHAPTER 78. USING ANSIBLE TO MANAGE DNS LOCATIONS IN IDM

As Identity Management (IdM) administrator, you can manage IdM DNS locations using the `location` module available in the `ansible-freeipa` package. The chapter describes the following topics and procedures:

- DNS-based service discovery
- Deployment considerations for DNS locations
- DNS time to live (TTL)
- Using Ansible to ensure an IdM location is present
- Using Ansible to ensure an IdM location is absent

78.1. DNS-BASED SERVICE DISCOVERY

DNS-based service discovery is a process in which a client uses the DNS protocol to locate servers in a network that offer a specific service, such as LDAP or Kerberos. One typical type of operation is to allow clients to locate authentication servers within the closest network infrastructure, because they provide a higher throughput and lower network latency, lowering overall costs.

The major advantages of service discovery are:

- No need for clients to be explicitly configured with names of nearby servers.
- DNS servers are used as central providers of policy. Clients using the same DNS server have access to the same policy about service providers and their preferred order.

In an Identity Management (IdM) domain, DNS service records (SRV records) exist for LDAP, Kerberos, and other services. For example, the following command queries the DNS server for hosts providing a TCP-based Kerberos service in an IdM DNS domain:

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
0 100 88 idmserver-01.idm.example.com.
0 100 88 idmserver-02.idm.example.com.
```

The output contains the following information:

- **0** (priority): Priority of the target host. A lower value is preferred.
- **100** (weight). Specifies a relative weight for entries with the same priority. For further information, see RFC 2782, section 3.
- **88** (port number): Port number of the service.
- Canonical name of the host providing the service.

In the example, the two host names returned have the same priority and weight. In this case, the client uses a random entry from the result list.
When the client is, instead, configured to query a DNS server that is configured in a DNS location, the output differs. For IdM servers that are assigned to a location, tailored values are returned. In the example below, the client is configured to query a DNS server in the location `germany`:

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
    _kerberos._tcp.germany._locations.idm.example.com.
0 100 88 idmserver-01.idm.example.com.
50 100 88 idmserver-02.idm.example.com.
```

The IdM DNS server automatically returns a DNS alias (CNAME) pointing to a DNS location specific SRV record which prefers local servers. This CNAME record is shown in the first line of the output. In the example, the host `idmserver-01.idm.example.com` has the lowest priority value and is therefore preferred. The `idmserver-02.idm.example.com` has a higher priority and thus is used only as backup for cases when the preferred host is unavailable.

### 78.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS

Identity Management (IdM) can generate location-specific service (SRV) records when using the integrated DNS. Because each IdM DNS server generates location-specific SRV records, you have to install at least one IdM DNS server in each DNS location.

The client’s affinity to a DNS location is only defined by the DNS records received by the client. For this reason, you can combine IdM DNS servers with non-IdM DNS consumer servers and recursors if the clients doing DNS service discovery resolve location-specific records from IdM DNS servers.

In the majority of deployments with mixed IdM and non-IdM DNS services, DNS recursors select the closest IdM DNS server automatically by using round-trip time metrics. Typically, this ensures that clients using non-IdM DNS servers are getting records for the nearest DNS location and thus use the optimal set of IdM servers.

### 78.3. DNS TIME TO LIVE (TTL)

Clients can cache DNS resource records for an amount of time that is set in the zone’s configuration. Because of this caching, a client might not be able to receive the changes until the time to live (TTL) value expires. The default TTL value in Identity Management (IdM) is 1 day.

If your client computers roam between sites, you should adapt the TTL value for your IdM DNS zone. Set the value to a lower value than the time clients need to roam between sites. This ensures that cached DNS entries on the client expire before they reconnect to another site and thus query the DNS server to refresh location-specific SRV records.

**Additional resources**

- For further information how to modify the default TTL of a DNS zone, see Configuration attributes of primary IdM DNS zones.

### 78.4. USING ANSIBLE TO ENSURE AN IDM LOCATION IS PRESENT

As a system administrator of Identity Management (IdM), you can configure IdM DNS locations to allow clients to locate authentication servers within the closest network infrastructure.
The following procedure describes how to use an Ansible playbook to ensure a DNS location is present in IdM. The example describes how to ensure that the `germany` DNS location is present in IdM. As a result, you can assign particular IdM servers to this location so that local IdM clients can use them to reduce server response time.

**Prerequisites**

- You know the IdM administrator password.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- The example assumes that you have created and configured the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.
- You understand the deployment considerations for DNS locations.

**Procedure**

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

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

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

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

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

4. Adapt the file by setting the following variables in the `ipalocation` 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 location.

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

   ```
   ---
   - name: location present example
     hosts: ipaserver
     become: true
     tasks:
       - name: Ensure that the "germany" location is present
         ipalocation:
           ipaadmin_password: Secret123
           name: germany
   ```

5. Save the file.

6. Run the Ansible playbook specifying the playbook file and the inventory file:
78.5. USING ANSIBLE TO ENSURE AN IDM LOCATION IS ABSENT

As a system administrator of Identity Management (IdM), you can configure IdM DNS locations to allow clients to locate authentication servers within the closest network infrastructure.

The following procedure describes how to use an Ansible playbook to ensure that a DNS location is absent in IdM. The example describes how to ensure that the `germany` DNS location is absent in IdM. As a result, you cannot assign particular IdM servers to this location and local IdM clients cannot use them.

**Prerequisites**

- You know the IdM administrator password.
- No IdM server is assigned to the `germany` DNS location.
- You have installed the `ansible-freeipa` package on the Ansible control node.
- The example assumes that you have created and configured the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

**Procedure**

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

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

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

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

3. Open the `location-absent-copy.yml` Ansible playbook file for editing.

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

   ```yaml
   - 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 DNS location.
   - Make sure that the `state` variable is set to `absent`.
   ``

   This is the modified Ansible playbook file for the current example:
---
- name: location absent example
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure that the "germany" location is absent
    ipalocation:
      ipaadmin_password: Secret123
      name: germany
      state: absent

5. Save the file.

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

   $ ansible-playbook -v -i inventory location-absent-copy.yml

78.6. ADDITIONAL RESOURCES

- You can see more sample Ansible playbooks for the ansible-freeipa ipalocation module in the README-location.md file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the ipalocation variables.

- You can see sample Ansible playbooks for the ipalocation module in the /usr/share/doc/ansible-freeipa/playbooks/location directory.
CHAPTER 79. MANAGING DNS FORWARDING IN IDM

The following procedures describe how to configure DNS global forwarders and DNS forward zones in the Identity Management (IdM) Web UI, the IdM CLI, and using Ansible:

- Section 79.1, “The two roles of an IdM DNS server”
- Section 79.2, “DNS forward policies in IdM”
- Section 79.3, “Adding a global forwarder in the IdM Web UI”
- Section 79.4, “Adding a global forwarder in the CLI”
- Section 79.5, “Adding a DNS Forward Zone in the IdM Web UI”
- Section 79.6, “Adding a DNS Forward Zone in the CLI”
- Section 79.7, “Establishing a DNS Global Forwarder in IdM using Ansible”
- Section 79.8, “Ensuring the presence of a DNS global forwarder in IdM using Ansible”
- Section 79.9, “Ensuring the absence of a DNS global forwarder in IdM using Ansible”
- Section 79.10, “Ensuring DNS Global Forwarders are disabled in IdM using Ansible”
- Section 79.11, “Ensuring the presence of a DNS Forward Zone in IdM using Ansible”
- Section 79.12, “Ensuring a DNS Forward Zone has multiple forwarders in IdM using Ansible”
- Section 79.13, “Ensuring a DNS Forward Zone is disabled in IdM using Ansible”
- Section 79.14, “Ensuring the absence of a DNS Forward Zone in IdM using Ansible”

Additional resources

- DNS forwarding

79.1. THE TWO ROLES OF AN IDM DNS SERVER

DNS forwarding affects how a DNS service answers DNS queries. By default, the Berkeley Internet Name Domain (BIND) service integrated with IdM acts as both an authoritative and a recursive DNS server:

Authoritative DNS server

When a DNS client queries a name belonging to a DNS zone for which the IdM server is authoritative, BIND replies with data contained in the configured zone. Authoritative data always takes precedence over any other data.

Recursive DNS server

When a DNS client queries a name for which the IdM server is not authoritative, BIND attempts to resolve the query using other DNS servers. If forwarders are not defined, BIND asks the root servers on the Internet and uses a recursive resolution algorithm to answer the DNS query.

In some cases, it is not desirable to let BIND contact other DNS servers directly and perform the recursion based on data available on the Internet. You can configure BIND to use another DNS server, a forwarder, to resolve the query.
When you configure BIND to use a forwarder, queries and answers are forwarded back and forth between the IdM server and the forwarder, and the IdM server acts as the DNS cache for non-authoritative data.

### 79.2. DNS FORWARD POLICIES IN IDM

IdM supports the **first** and **only** standard BIND forward policies, as well as the **none** IdM-specific forward policy.

**Forward first (default)**

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The **forward first** policy is the default policy, and it is suitable for optimizing DNS traffic.

**Forward only**

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND returns an error to the client. The **forward only** policy is recommended for environments with split DNS configuration.

**None (forwarding disabled)**

DNS queries are not forwarded with the **none** forwarding policy. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.

**NOTE**

You cannot use forwarding to combine data in IdM with data from other DNS servers. You can only forward queries for specific subzones of the primary zone in IdM DNS.

By default, the BIND service does not forward queries to another server if the queried DNS name belongs to a zone for which the IdM server is authoritative. In such a situation, if the queried DNS name cannot be found in the IdM database, the **NXDOMAIN** answer is returned. Forwarding is not used.

**Example 79.1. Example Scenario**

The IdM server is authoritative for the **test.example** DNS zone. BIND is configured to forward queries to the DNS server with the **192.0.2.254** IP address.

When a client sends a query for the **nonexistent.test.example**. DNS name, BIND detects that the IdM server is authoritative for the **test.example** zone and does not forward the query to the **192.0.2.254** server. As a result, the DNS client receives the **NXDomain** error message, informing the user that the queried domain does not exist.

### 79.3. ADDING A GLOBAL FORWARDER IN THE IDM WEB UI

This section describes how to add a global DNS forwarder in the Identity Management (IdM) Web UI.

**Prerequisites**

- You are logged in to the IdM WebUI as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.
**Procedure**

1. In the IdM Web UI, select **Network Services → DNS Global Configuration → DNS**.

   ![Image of IdM Web UI showing Network Services and DNS Global Configuration sections]

2. In the **DNS Global Configuration** section, click **Add**.

   ![Image of IdM Web UI showing DNS Global Configuration with Add button highlighted]

3. Specify the IP address of the DNS server that will receive forwarded DNS queries.
4. Select the **Forward policy**.

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

**Verification steps**

1. Select **Network Services → DNS Global Configuration → DNS**.
Verify that the global forwarder, with the forward policy you specified, is present and enabled in the IdM Web UI.

This section describes how to add a global DNS forwarder from the command line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

Procedure

- Use the `ipa dnsconfig-mod` command to add a new global forwarder. Specify the IP address of the DNS forwarder with the `--forwarder` option.

```
[user@server ~]$ ipa dnsconfig-mod --forwarder=10.10.0.1
Server will check DNS forwarder(s).
```
Verification steps

- Use the `dnsconfig-show` command to display global forwarders.

```bash
[user@server ~]$ ipa dnsconfig-show
Global forwarders: 10.10.0.1
IPA DNS servers: server.example.com
```

79.5. ADDING A DNS FORWARD ZONE IN THE IDM WEB UI

This section describes how to add a DNS forward zone in the Identity Management (IdM) Web UI.

**IMPORTANT**

Do not use forward zones unless absolutely required. Forward zones are not a standard solution, and using them can lead to unexpected and problematic behavior. If you must use forward zones, limit their use to overriding a global forwarding configuration.

When creating a new DNS zone, Red Hat recommends to always use standard DNS delegation using nameserver (NS) records and to avoid forward zones. In most cases, using a global forwarder is sufficient, and forward zones are not necessary.

**Prerequisites**

- You are logged in to the IdM WebUI as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

**Procedure**

1. In the IdM Web UI, select **Network Services → DNS Forward Zones → DNS**.

2. In the **DNS Forward Zones** section, click **Add**.
3. In the Add DNS forward zone window, specify the forward zone name.

4. Click the Add button and specify the IP address of a DNS server to receive the forwarding request. You can specify multiple forwarders per forward zone.
5. Select the **Forward policy**.

6. Click **Add** at the bottom of the window to add the new forward zone.

**Verification steps**

1. In the IdM Web UI, select **Network Services → DNS Forward Zones → DNS**.
2. Verify that the forward zone you created, with the forwarders and forward policy you specified, is present and enabled in the IdM Web UI.

79.6. ADDING A DNS FORWARD ZONE IN THE CLI

This section describes how to add a DNS forward zone from the command line interface (CLI).

**IMPORTANT**

Do not use forward zones unless absolutely required. Forward zones are not a standard solution, and using them can lead to unexpected and problematic behavior. If you must use forward zones, limit their use to overriding a global forwarding configuration.

When creating a new DNS zone, Red Hat recommends to always use standard DNS delegation using nameserver (NS) records and to avoid forward zones. In most cases, using a global forwarder is sufficient, and forward zones are not necessary.

**Prerequisites**

- You are logged in as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

**Procedure**

- Use the `dnsforwardzone-add` command to add a new forward zone. Specify at least one forwarder with the `--forwarder` option if the forward policy is not `none`, and specify the forward policy with the `--forward-policy` option.
Verification steps

- Use the `ipa dnsforwardzone-show` command to display the DNS forward zone you just created.

```bash
[user@server ~]$ ipa dnsforwardzone-show forward.example.com.
```

Zone name: forward.example.com.
Zone forwarders: 10.10.0.14, 10.10.1.15
Forward policy: first

79.7. ESTABLISHING A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to establish a DNS Global Forwarder in IdM.

In the example procedure below, the IdM administrator creates a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of 8.8.6.6 and IPv6 address of 2001:4860:4860::8800 on port 53.

Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `set-configuration.yml` Ansible playbook file. For example:

   ```bash
   $ cp set-configuration.yml establish-global-forwarder.yml
   ```

4. Open the `establish-global-forwarder.yml` file for editing.
5. Adapt the file by setting the following variables:
   a. Change the name variable for the playbook to **Playbook to establish a global forwarder in IdM DNS**.
   b. In the tasks section, change the name of the task to **Create a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800**.
   c. In the forwarders section of the ipadnsconfig portion:
      i. Change the first ip_address value to the IPv4 address of the global forwarder: **8.8.6.6**.
      ii. Change the second ip_address value to the IPv6 address of the global forwarder: **2001:4860:4860::8800**.
      iii. Verify the port value is set to **53**.
   d. Change the forward_policy to **first**.
      This the modified Ansible playbook file for the current example:

```
---
- name: Playbook to establish a global forwarder in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Create a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800
    ipadnsconfig:
      forwarders:
        - ip_address: 8.8.6.6
        - ip_address: 2001:4860:4860::8800
      port: 53
      forward_policy: first
      allow_sync_ptr: yes
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook -v -i inventory.file establish-global-forwarder.yml
```

Additional resources

- You can see more sample Ansible playbooks for the **ansible-freeipa ipadnsconfig** module in the README-dnsconfig.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of **ipadnsconfig** variables.

### 79.8. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the presence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of **7.7.9.9** and IP v6 address of **2001:db8::1:0** on port **53**.
Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

   [ipaserver]
   server.idm.example.com

3. Make a copy of the forwarders-absent.yml Ansible playbook file. For example:

   $ cp forwarders-absent.yml ensure-presence-of-a-global-forwarder.yml

4. Open the ensure-presence-of-a-global-forwarder.yml file for editing.

5. Adapt the file by setting the following variables:

   a. Change the name variable for the playbook to **Playbook to ensure the presence of a global forwarder in IdM DNS**.
   
   b. In the tasks section, change the name of the task to **Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53**.
   
   c. In the forwarders section of the ipadnsconfig portion:
      
      i. Change the first ip_address value to the IPv4 address of the global forwarder: **7.7.9.9**.
      
      ii. Change the second ip_address value to the IPv6 address of the global forwarder: **2001:db8::1:0**.
      
      iii. Verify the port value is set to **53**.
   
   d. Change the state to present.

   This the modified Ansible playbook file for the current example:

   ```yaml
   ---
   - name: Playbook to ensure the presence of a global forwarder in IdM DNS
     hosts: ipaserver
     become: true

     tasks:
     - name: Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53
       ipadnsconfig:
   ```
forwarders:
- ip_address: 7.7.9.9
- ip_address: 2001:db8::1:0
  port: 53
  state: present

6. Save the file.

7. Run the playbook:

   $ ansible-playbook -v -i inventory.file ensure-presence-of-a-global-forwarder.yml

Additional resources

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsconfig module in the README-dnsconfig.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of ipadnsconfig variables.

79.9. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the absence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS global forwarder with an Internet Protocol (IP) v4 address of 8.8.6.6 and IP v6 address of 2001:4860:4860::8800 on port 53.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

   [ipaserver]
   server.idm.example.com

3. Make a copy of the forwards-absent.yml Ansible playbook file. For example:

   $ cp forwards-absent.yml ensure-absence-of-a-global-forwarder.yml

4. Open the ensure-absence-of-a-global-forwarder.yml file for editing.

5. Adapt the file by setting the following variables:
a. Change the name variable for the playbook to **Playbook to ensure the absence of a global forwarder in IdM DNS**.

b. In the tasks section, change the name of the task to **Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53**.

c. In the forwarders section of the ipadnsconfig portion:
   i. Change the first ip_address value to the IPv4 address of the global forwarder: **8.8.6.6**.
   ii. Change the second ip_address value to the IPv6 address of the global forwarder: **2001:4860:4860::8800**.
   iii. Verify the port value is set to **53**.

d. Set the action variable to **member**.

e. Verify the state is set to **absent**.

This the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to ensure the absence of a global forwarder in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53
    ipadnsconfig:
      forwarders:
      - ip_address: 8.8.6.6
      - ip_address: 2001:4860:4860::8800
      port: 53
      action: member
      state: absent
```

6. Save the file.

7. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file ensure-absence-of-a-global-forwarder.yml
   ```

Additional resources

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsconfig module in the README-dnsconfig.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of ipadnsconfig variables.

### 79.10. ENSURING DNS GLOBAL FORWARDERS ARE DISABLED IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure DNS Global Forwarders are disabled in IdM. In the example procedure below, the IdM administrator ensures that the forwarding policy for the global forwarder is set to **none**, which
effectively disables the global forwarder.

**Prerequisites**

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.

- You know the IdM administrator password.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:
   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:
   ```
   [ipaserver]
   server.idm.example.com
   ```

3. Verify the contents of the `disable-global-forwarders.yml` Ansible playbook file which is already configured to disable all DNS global forwarders. For example:
   ```bash
   $ cat disable-global-forwarders.yml
   ---
   - name: Playbook to disable global DNS forwarders
     hosts: ipaserver
     become: true

     tasks:
     - name: Disable global forwarders.
       ipadnsconfig:
         forward_policy: none
   ```

4. Run the playbook:
   ```bash
   $ ansible-playbook -v -i inventory.file disable-global-forwarders.yml
   ```

**Additional resources**

- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsconfig` module in the `README-dnsconfig.md` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of `ipadnsconfig` variables.

---

**79.11. ENSURING THE PRESENCE OF A DNS FORWARD ZONE IN IDM USING ANSIBLE**

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the presence of a DNS Forward Zone in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS forward zone for `example.com` to a DNS server with an Internet Protocol (IP) address of `8.8.8.8`. 
Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:
   
   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:
   
   ```
   [ipaserver]
   server.idm.example.com
   ```

3. Make a copy of the forwarders-absent.yml Ansible playbook file. For example:
   
   ```
   $ cp forwarders-absent.yml ensure-presence-forwardzone.yml
   ```

4. Open the ensure-presence-forwardzone.yml file for editing.

5. Adapt the file by setting the following variables:
   
   a. Change the name variable for the playbook to Playbook to ensure the presence of a dnsforwardzone in IdM DNS.
   
   b. In the tasks section, change the name of the task to Ensure presence of a dnsforwardzone for example.com to 8.8.8.8.
   
   c. In the tasks section, change the ipadnsconfig heading to ipadnsforwardzone.
   
   d. In the ipadnsforwardzone section:
      
      i. Add the ipaadmin_password variable and set it to your IdM administrator password.
      
      ii. Add the name variable and set it to example.com.
      
      iii. In the forwarders section:
           
           A. Remove the ip_address and port lines.
           
           B. Add the IP address of the DNS server to receive forwarded requests by specifying it after a dash:
           
           ```
           - 8.8.8.8
           ```
      
      iv. Add the forwardpolicy variable and set it to first.
      
      v. Add the skip_overlap_check variable and set it to true.
      
      vi. Change the state variable to present.
This is the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to ensure the presence of a dnsforwardzone in IdM DNS
  hosts: ipaserver
  become: true
  tasks:
    - name: Ensure the presence of a dnsforwardzone for example.com to 8.8.8.8
      ipadnsforwardzone:
        ipaadmin_password: password01
        name: example.com
        forwarders:
          - 8.8.8.8
          - 4.4.4.4
        forwardpolicy: first
        skip_overlap_check: true
        state: present

6. Save the file.

7. Run the playbook:

   $ ansible-playbook -v -i inventory.file ensure-presence-forwardzone.yml

Additional resources

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsforwardzone module in the README-dnsforwardzone.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of ipadnsforwardzone variables.

79.12. ENSURING A DNS FORWARD ZONE HAS MULTIPLE FORWARDERS IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure a DNS Forward Zone in IdM has multiple forwarders. In the example procedure below, the IdM administrator ensures the DNS forward zone for example.com is forwarding to 8.8.8.8 and 4.4.4.4.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.

- You know the IdM administrator password.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsconfig directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `ipaserver` section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the `forwarders-absent.yml` Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-multiple-forwarders.yml
```

4. Open the `ensure-presence-multiple-forwarders.yml` file for editing.

5. Adapt the file by setting the following variables:

   a. Change the `name` variable for the playbook to **Playbook to ensure the presence of multiple forwarders in a dnsforwardzone in IdM DNS**.

   b. In the `tasks` section, change the `name` of the task to **Ensure presence of 8.8.8.8 and 4.4.4.4 forwarders in dnsforwardzone for example.com**.

   c. In the `tasks` section, change the `ipadnsconfig` heading to `ipadnsforwardzone`.

   d. In the `ipadnsforwardzone` section:

      i. Add the `ipaadmin_password` variable and set it to your IdM administrator password.

      ii. Add the `name` variable and set it to **example.com**.

      iii. In the `forwarders` section:

         A. Remove the `ip_address` and `port` lines.

         B. Add the IP address of the DNS servers you want to ensure are present, preceded by a dash:

         ```
         - 8.8.8.8
         - 4.4.4.4
         ```

   iv. Change the state variable to present.

   This the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to ensure the presence of multiple forwarders in a dnsforwardzone in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure presence of 8.8.8.8 and 4.4.4.4 forwarders in dnsforwardzone for example.com
    ipadnsforwardzone:
      ipaadmin_password: password01
      name: example.com
      forwarders:
```
6. Save the file.

7. Run the playbook:

   ```
   $ ansible-playbook -v -i inventory.file ensure-presence-multiple-forwarders.yml
   ```

Additional resources

- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsforwardzone` module in the `README-dnsforwardzone.md` Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of `ipadnsforwardzone` variables.

### 79.13. ENSURING A DNS FORWARD ZONE IS DISABLED IN IDM USING ANSIBLE

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure a DNS Forward Zone is disabled in IdM. In the example procedure below, the IdM administrator ensures the DNS forward zone for `example.com` is disabled.

**Prerequisites**

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `forwarders-absent.yml` Ansible playbook file. For example:

   ```
   $ cp forwarders-absent.yml ensure-disabled-forwardzone.yml
   ```

4. Open the `ensure-disabled-forwardzone.yml` file for editing.

5. Adapt the file by setting the following variables:
a. Change the name variable for the playbook to **Playbook to ensure a dnsforwardzone is disabled in IdM DNS**.

b. In the tasks section, change the name of the task to **Ensure a dnsforwardzone for example.com is disabled**.

c. In the tasks section, change the ipadnsconfig heading to **ipadnsforwardzone**.

d. In the ipadnsforwardzone section:
   
   i. Add the ipaadmin_password variable and set it to your IdM administrator password.
   
   ii. Add the name variable and set it to **example.com**.
   
   iii. Remove the entire forwarders section.
   
   iv. Change the state variable to **disabled**.

   This the modified Ansible playbook file for the current example:

```yaml
---
- name: Playbook to ensure a dnsforwardzone is disabled in IdM DNS
  hosts: ipaserver
  become: true

  tasks:
  - name: Ensure a dnsforwardzone for example.com is disabled
    ipadnsforwardzone:
      ipaadmin_password: password01
      name: example.com
      state: disabled
```

6. Save the file.

7. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file ensure-disabled-forwardzone.yml
   ```

**Additional resources**

- You can see more sample Ansible playbooks for the **ansible-freeipa ipadnsforwardzone** module in the **README-dnsforwardzone.md** Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of **ipadnsforwardzone** variables.

### 79.14. Ensuring the Absence of a DNS Forward Zone in IDM Using Ansible

This section describes how an Identity Management (IdM) administrator can use an Ansible playbook to ensure the absence of a DNS Forward Zone in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS forward zone for **example.com**.

**Prerequisites**

...
You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.

You know the IdM administrator password.

**Procedure**

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

   ```bash
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
   ```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

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

3. Make a copy of the `forwarders-absent.yml` Ansible playbook file. For example:

   ```bash
   $ cp forwarders-absent.yml ensure-absence-forwardzone.yml
   ```

4. Open the `ensure-absence-forwardzone.yml` file for editing.

5. Adapt the file by setting the following variables:
   a. Change the `name` variable for the playbook to **Playbook to ensure the absence of a dnsforwardzone in IdM DNS**.
   b. In the `tasks` section, change the `name` of the task to **Ensure the absence of a dnsforwardzone for example.com**.
   c. In the `tasks` section, change the `ipadnsconfig` heading to `ipadnsforwardzone`.
   d. In the `ipadnsforwardzone` section:
      i. Add the `ipaadmin_password` variable and set it to your IdM administrator password.
      ii. Add the `name` variable and set it to `example.com`.
      iii. Remove the entire `forwarders` section.
      iv. Leave the `state` variable as `absent`.

   This the modified Ansible playbook file for the current example:

   ```yaml
   ---
   - name: Playbook to ensure the absence of a dnsforwardzone in IdM DNS
     hosts: ipaserver
     become: true

     tasks:
     - name: Ensure the absence of a dnsforwardzone for example.com
       ipadnsforwardzone:
   ```
ipaadmin_password: password01
ame: example.com
state: absent

6. Save the file.

7. Run the playbook:

   $ ansible-playbook -v -i inventory.file ensure-absence-forwardzone.yml

Additional resources

- You can see more sample Ansible playbooks for the **ansible-freeipa ipadnsforwardzone** module in the **README-dnsforwardzone.md** Markdown file available in the **/usr/share/doc/ansible-freeipa/** directory. The file also contains the definitions of **ipadnsforwardzone** variables.
This chapter describes how to manage DNS records in Identity Management (IdM). As an IdM administrator, you can add, modify and delete DNS records in IdM. The chapter contains the following sections:

- DNS records in IdM
- Adding DNS resource records from the IdM Web UI
- Adding DNS resource records from the IdM CLI
- Common ipa dnsrecord-add options
- Deleting DNS records in the IdM Web UI
- Deleting an entire DNS record in the IdM Web UI
- Deleting DNS records in the IdM CLI

Prerequisites

- Your IdM deployment contains an integrated DNS server. For information how to install IdM with integrated DNS, see one of the following links:
  - Installing an IdM server: With integrated DNS, with an integrated CA as the root CA
  - Installing an IdM server: With integrated DNS, with an external CA as the root CA

80.1. DNS RECORDS IN IDM

Identity Management (IdM) supports many different DNS record types. The following four are used most frequently:

A

This is a basic map for a host name and an IPv4 address. The record name of an A record is a host name, such as www. The **IP Address** value of an A record is an IPv4 address, such as 192.0.2.1. For more information about A records, see RFC 1035.

AAAA

This is a basic map for a host name and an IPv6 address. The record name of an AAAA record is a host name, such as www. The **IP Address** value is an IPv6 address, such as 2001:DB8::1111. For more information about AAAA records, see RFC 3596.

SRV

**Service (SRV) resource records** map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the server which manages it.

The record name of an SRV record has the format **_service_** _protocol_, such as _ldap_.tcp. The configuration options for SRV records include priority, weight, port number, and host name for the target service.

For more information about SRV records, see RFC 2782.
**PTR**

A pointer record (PTR) adds a reverse DNS record, which maps an IP address to a domain name.

**NOTE**

All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the `in-addr.arpa` domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the `in-addr.arpa` domain appended to it. For example, for the network address `192.0.2.0/24`, the reverse zone is `2.0.192.in-addr.arpa`.

The record name of a PTR must be in the standard format specified in RFC 1035, extended in RFC 2317, and RFC 3596. The host name value must be a canonical host name of the host for which you want to create the record.

**NOTE**

Reverse zones can also be configured for IPv6 addresses, with zones in the `.ip6.arpa` domain. For more information about IPv6 reverse zones, see RFC 3596.

When adding DNS resource records, note that many of the records require different data. For example, a CNAME record requires a host name, while an A record requires an IP address. In the IdM Web UI, the fields in the form for adding a new record are updated automatically to reflect what data is required for the currently selected type of record.

### 80.2. ADDING DNS RESOURCE RECORDS IN THE IDM WEB UI

This section describes how to add DNS resource records in the Identity Management (IdM) Web UI.

**Prerequisites**

- The DNS zone to which you want to add a DNS record exists and is managed by IdM. For more information about creating a DNS zone in IdM DNS, see Managing DNS zones in IdM.
- You are logged in as IdM administrator.

**Procedure**

1. In the IdM Web UI, click Network Services → DNS → DNS Zones.
2. Click the DNS zone to which you want to add a DNS record.
3. In the DNS Resource Records section, click Add to add a new record.
4. Select the type of record to create and fill out the other fields as required.

5. Click **Add** to confirm the new record.

### 80.3. Adding DNS Resource Records from the IDM CLI

This section describes how to add a DNS resource record of any type from the command line interface (CLI).

**Prerequisites**

- The DNS zone to which you want to add a DNS records exists. For more information about creating a DNS zone in IdM DNS, see [Managing DNS zones in IdM](#).

---

**Figure 80.1. Adding a New DNS Resource Record**

![Figure 80.1. Adding a New DNS Resource Record](image1)

**Figure 80.2. Defining a New DNS Resource Record**

![Figure 80.2. Defining a New DNS Resource Record](image2)

---

**Figure 80.3. Adding DNS Resource Records from the IDM CLI**

![Figure 80.3. Adding DNS Resource Records from the IDM CLI](image3)
You are logged in as IdM administrator.

Procedure

1. To add a DNS resource record, use the `ipa dnsrecord-add` command. The command follows this syntax:

   ```
   $ ipa dnsrecord-add zone_name record_name --record_type_option=data
   ```

   In the command above:
   - The `zone_name` is the name of the DNS zone to which the record is being added.
   - The `record_name` is an identifier for the new DNS resource record.

   For example, to add an A type DNS record of `host1` to the `idm.example.com` zone, enter:

   ```
   $ ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123
   ```

80.4. COMMON IPA DNSRECORD-* OPTIONS

This section describes the options you can use when adding, modifying and deleting the most common DNS resource record types to Identity Management (IdM):

- A (IPv4)
- AAAA (IPv6)
- SRV
- PTR

In **Bash**, you can define multiple entries by listing the values in a comma-separated list inside curly braces, such as `--option={val1,val2,val3}`.

### Table 80.1. General Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--ttl=number</code></td>
<td>Sets the time to live for the record.</td>
</tr>
<tr>
<td><code>--structured</code></td>
<td>Parses the raw DNS records and returns them in a structured format.</td>
</tr>
</tbody>
</table>

### Table 80.2. "A" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--a-rec=ARECORD</code></td>
<td>Passes a single A record or a list of A records.</td>
<td><code>ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123</code></td>
</tr>
</tbody>
</table>
Can create a wildcard A record with a given IP address.

```plaintext
ipa dnsrecord-add
idm.example.com "*" --a-rec=192.168.122.123 [a]
```

---

**Option** | **Description** | **Examples**
---|---|---
--a-ip-address=string | Gives the IP address for the record. When creating a record, the option to specify the A record value is --a-rec. However, when modifying an A record, the --a-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option. | ipa dnsrecord-mod
idm.example.com --a-rec 192.168.122.123 --a-ip-address 192.168.122.124

---

[a] The example creates a wildcard A record with the IP address of 192.0.2.123.

### Table 80.3. "AAAA" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| --aaaa-rec=AAAA_REC | Passes a single AAAA (IPv6) record or a list of AAAA records. | ipa dnsrecord-add
idm.example.com www --aaaa-rec 2001:db8::1231:5675 |

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| --aaaa-ip-address=string | Gives the IPv6 address for the record. When creating a record, the option to specify the A record value is --aaaa-rec. However, when modifying an A record, the --aaaa-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option. | ipa dnsrecord-mod
idm.example.com --aaaa-rec 2001:db8::1231:5675 --aaaa-ip-address 2001:db8::1231:5676 |

### Table 80.4. "PTR" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| --ptr-rec=PTRREC | Passes a single PTR record or a list of PTR records. When adding the reverse DNS record, the zone name used with the `ipa dnsrecord-add` command is reversed, compared to the usage for adding other DNS records. Typically, the host IP address is the last octet of the IP address in a given network. The first example on the right adds a PTR record for `server4.idm.example.com` with IPv4 address 192.168.122.4. The second example adds a reverse DNS entry to the 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. IPv6 reverse zone for the host `server2.example.com` with the IP address 2001:DB8::1111. | ipa dnsrecord-add
122.168.192.in-addr.arpa 4 --ptr-rec
server4.idm.example.com. |

$ ipa dnsrecord-add
0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. 1.1.1.0.0.0.0.0.0.0.0.0.0 --ptr-rec
server2.example.com.
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ptr-hostname=string</td>
<td>Gives the host name for the record.</td>
<td></td>
</tr>
</tbody>
</table>

Table 80.5. "SRV" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--srv-rec=SRVRECORD</td>
<td>Passes a single SRV record or a list of SRV records. In the examples on the right, _ldap._tcp defines the service type and the connection protocol for the SRV record. The <code>--srv-rec</code> option defines the priority, weight, port, and target values. The weight values of 51 and 49 in the examples add up to 100 and represent the probability, in percentages, that a particular record is used.</td>
<td># ipa dnsrecord-add idm.example.com _ldap._tcp --srv-rec=&quot;0 51 389 server1.idm.example.com.&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--srv-priority=number</td>
<td>Sets the priority of the record. There can be multiple SRV records for a service type. The priority (0 - 65535) sets the rank of the record; the lower the number, the higher the priority. A service has to use the record with the highest priority first.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;1 49 389 server2.idm.example.com.&quot; --srv-priority=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--srv-weight=number</td>
<td>Sets the weight of the record. This helps determine the order of SRV records with the same priority. The set weights should add up to 100, representing the probability (in percentages) that a particular record is used.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;0 49 389 server2.idm.example.com.&quot; --srv-weight=60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--srv-port=number</td>
<td>Gives the port for the service on the target host.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;0 60 389 server2.idm.example.com.&quot; --srv-port=636</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--srv-target=string</td>
<td>Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.</td>
<td></td>
</tr>
</tbody>
</table>

**Additional resources**

- For more information on how to use `ipa dnsrecord-add` and which DNS record types are supported by IdM, run the `ipa dnsrecord-add --help` command.
80.5. DELETING DNS RECORDS IN THE IDM WEB UI

This section describes how to delete DNS records in Identity Management (IdM) using the IdM Web UI.

Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.
2. Click the zone from which you want to delete a DNS record, for example *example.com*.
3. In the **DNS Resource Records** section, click the name of the resource record.

   ![Figure 80.3. Selecting a DNS Resource Record]

4. Select the check box by the name of the record type to delete.
5. Click **Delete**.
The selected record type is now deleted. The other configuration of the resource record is left intact.

Additional resources

- For more information on deleting an entire DNS record, see Deleting an entire DNS record in the IdM Web UI.

80.6. DELETING AN ENTIRE DNS RECORD IN THE IDM WEB UI

This section describes how to delete all the records for a particular resource in a zone using the Identity Management (IdM) Web UI.

Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.

2. Click the zone from which you want to delete a DNS record, for example `zone.example.com`.

3. In the **DNS Resource Records** section, select the check box of the resource record to delete.

4. Click **Delete**.
The entire resource record is now deleted.

### 80.7. DELETING DNS RECORDS IN THE IDM CLI

This section describes how to remove DNS records from a zone managed by the Identity Management (IdM) DNS.

**Prerequisites**

- You are logged in as IdM administrator.

**Procedure**

- To remove records from a zone, use the `ipa dnsrecord-del` command and add the `-recordType-rec` option together with the record value. For example, to remove an A type record:

  ```
  $ ipa dnsrecord-del example.com www --a-rec 192.0.2.1
  ```

  If you run `ipa dnsrecord-del` without any options, the command prompts for information about the record to delete. Note that passing the `--del-all` option with the command removes all associated records for the zone.

**Additional resources**

- For detailed information on how to use `ipa dnsrecord-del` and a complete list of options accepted by the command, run the `ipa dnsrecord-del --help` command.

### 80.8. ADDITIONAL RESOURCES

- You can use the `ansible-freeipa ipadnsrecord` module to add, modify and delete records in IdM DNS. For more information, see [Using Ansible to manage DNS records in IdM](#).
CHAPTER 81. USING ANSIBLE TO MANAGE DNS RECORDS IN IDM

This chapter describes how to manage DNS records in Identity Management (IdM) using an Ansible playbook. As an IdM administrator, you can add, modify, and delete DNS records in IdM. The chapter contains the following sections:

- Ensuring the presence of A and AAAA DNS records in IdM using Ansible
- Ensuring the presence of A and PTR DNS records in IdM using Ansible
- Ensuring the presence of multiple DNS records in IdM using Ansible
- Ensuring the presence of multiple CNAME records in IdM using Ansible
- Ensuring the presence of an SRV record in IdM using Ansible

81.1. DNS RECORDS IN IDM

Identity Management (IdM) supports many different DNS record types. The following four are used most frequently:

A
This is a basic map for a host name and an IPv4 address. The record name of an A record is a host name, such as www. The IP Address value of an A record is an IPv4 address, such as 192.0.2.1.
For more information about A records, see RFC 1035.

AAAA
This is a basic map for a host name and an IPv6 address. The record name of an AAAA record is a host name, such as www. The IP Address value is an IPv6 address, such as 2001:DB8::1111.
For more information about AAAA records, see RFC 3596.

SRV
Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the server which manages it.
The record name of an SRV record has the format _service._protocol, such as _ldap._tcp. The configuration options for SRV records include priority, weight, port number, and host name for the target service.
For more information about SRV records, see RFC 2782.

PTR
A pointer record (PTR) adds a reverse DNS record, which maps an IP address to a domain name.

NOTE
All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the in-addr.arpa domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the in-addr.arpa domain appended to it. For example, for the network address 192.0.2.0/24, the reverse zone is 2.0.192.in-addr.arpa.
The record name of a PTR must be in the standard format specified in RFC 1035, extended in RFC 2317, and RFC 3596. The host name value must be a canonical host name of the host for which you want to create the record.

**NOTE**

Reverse zones can also be configured for IPv6 addresses, with zones in the .ip6.arpa domain. For more information about IPv6 reverse zones, see RFC 3596.

When adding DNS resource records, note that many of the records require different data. For example, a CNAME record requires a host name, while an A record requires an IP address. In the IdM Web UI, the fields in the form for adding a new record are updated automatically to reflect what data is required for the currently selected type of record.

### 81.2. COMMON IPA DNSRECORD-* OPTIONS

This section describes the options you can use when adding, modifying and deleting the most common DNS resource record types to Identity Management (IdM):

- A (IPv4)
- AAAA (IPv6)
- SRV
- PTR

In Bash, you can define multiple entries by listing the values in a comma-separated list inside curly braces, such as `--option={val1,val2,val3}`.

#### Table 81.1. General Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ttl number</td>
<td>Sets the time to live for the record.</td>
</tr>
<tr>
<td>--structured</td>
<td>Parses the raw DNS records and returns them in a structured format.</td>
</tr>
</tbody>
</table>

#### Table 81.2. "A" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>--a-rec ARECORD</td>
<td>Passes a single A record or a list of A records.</td>
<td>ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123</td>
</tr>
<tr>
<td></td>
<td>Can create a wildcard A record with a given IP address.</td>
<td>ipa dnsrecord-add idm.example.com &quot;*&quot; --a-rec=192.168.122.123 [a]</td>
</tr>
</tbody>
</table>
---

**Option** | **Description** | **Examples**
--- | --- | ---
--a-ip-address=string | Gives the IP address for the record. When creating a record, the option to specify the A record value is `--a-rec`. However, when modifying an A record, the `--a-rec` option is used to specify the current value for the A record. The new value is set with the `--a-ip-address` option. | `ipa dnsrecord-mod
idm.example.com --a-rec
192.168.122.123 --a-ip-address 192.168.122.124`

[a] The example creates a wildcard A record with the IP address of 192.0.2.123.

### Table 81.3. "AAAA" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| --aaaa-rec=AAAAARECORD | Passes a single AAAA (IPv6) record or a list of AAAA records.               | `ipa dnsrecord-add
idm.example.com www --aaaa-rec
2001:db8::1231:5675`          |
| --aaaa-ip-address=string | Gives the IPv6 address for the record. When creating a record, the option to specify the A record value is `--aaaa-rec`. However, when modifying an A record, the `--aaaa-rec` option is used to specify the current value for the A record. The new value is set with the `--a-ip-address` option. | `ipa dnsrecord-mod
idm.example.com --aaaa-rec
2001:db8::1231:5675 --aaaa-ip-address
2001:db8::1231:5676`          |

### Table 81.4. "PTR" record options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| --ptr-rec=PTRRECORD | Passes a single PTR record or a list of PTR records.  When adding the reverse DNS record, the zone name used with the `ipa dnsrecord-add` command is reversed, compared to the usage for adding other DNS records. Typically, the host IP address is the last octet of the IP address in a given network. The first example on the right adds a PTR record for `server4.idm.example.com` with IPv4 address `192.168.122.4`. The second example adds a reverse DNS entry to the `0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa`. IPv6 reverse zone for the host `server2.example.com` with the IP address `2001:DB8::1111`. | `ipa dnsrecord-add
122.168.192.in-addr.arpa 4 --ptr-rec
server4.idm.example.com.`          |
| --ptr-hostname=string | Gives the host name for the record.                                          | `$ ipa dnsrecord-add
0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
1.1.1.0.0.0.0.0.0.0.0.0 --ptr-rec
server2.idm.example.com.`          |
### Table 81.5. "SRV" Record Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>--srv-rec=SRVRECORD</td>
<td>Passes a single SRV record or a list of SRV records. In the examples on the right, _ldap._tcp defines the service type and the connection protocol for the SRV record. The <code>--srv-rec</code> option defines the priority, weight, port, and target values. The weight values of 51 and 49 in the examples add up to 100 and represent the probability, in percentages, that a particular record is used.</td>
<td># ipa dnsrecord-add idm.example.com _ldap._tcp --srv-rec=&quot;0 51 389 server1.idm.example.com.&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td># ipa dnsrecord-add server.idm.example.com _ldap._tcp --srv-rec=&quot;1 49 389 server2.idm.example.com.&quot;</td>
</tr>
<tr>
<td>--srv-priority=number</td>
<td>Sets the priority of the record. There can be multiple SRV records for a service type. The priority (0 - 65535) sets the rank of the record; the lower the number, the higher the priority. A service has to use the record with the highest priority first.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;1 49 389 server2.idm.example.com.&quot; --srv-priority=0</td>
</tr>
<tr>
<td>--srv-weight=number</td>
<td>Sets the weight of the record. This helps determine the order of SRV records with the same priority. The set weights should add up to 100, representing the probability (in percentages) that a particular record is used.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;0 49 389 server2.idm.example.com.&quot; --srv-weight=60</td>
</tr>
<tr>
<td>--srv-port=number</td>
<td>Gives the port for the service on the target host.</td>
<td># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec=&quot;0 60 389 server2.idm.example.com.&quot; --srv-port=636</td>
</tr>
<tr>
<td>--srv-target=string</td>
<td>Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.</td>
<td></td>
</tr>
</tbody>
</table>

### Additional resources

- For more information on how to use `ipa dnsrecord-add` and which DNS record types are supported by IdM, run the `ipa dnsrecord-add --help` command.

## 81.3. ENSURING THE PRESENCE OF A AND AAAA DNS RECORDS IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that A and AAAA records for a particular IdM host are present. In the example used in the procedure below, an IdM administrator ensures the presence of A and AAAA records for **host1** in the `idm.example.com` DNS zone.
Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
- The idm.example.com zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see Using Ansible playbooks to manage IdM DNS zones.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory:
   
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

   [ipaserver]
   server.idm.example.com

3. Make a copy of the ensure-A-and-AAAA-records-are-present.yml Ansible playbook file. For example:


4. Open the ensure-A-and-AAAA-records-are-present-copy.yml file for editing.

5. Adapt the file by setting the following variables in the ipadnsrecord task section:

   - Set the ipaadmin_password variable to your IdM administrator password.
   - Set the zone_name variable to idm.example.com.
   - In the records variable, set the name variable to host1, and the a_ip_address variable to 192.168.122.123.
   - In the records variable, set the name variable to host1, and the aaaa_ip_address variable to ::1.

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

   ---
   - name: Ensure A and AAAA records are present
     hosts: ipaserver
     become: true
     gather_facts: false

     tasks:
     # Ensure A and AAAA records are present
     - name: Ensure that 'host1' has A and AAAA records.
       ipadnsrecord:
         ipaadmin_password: Secret123
zone_name: idm.example.com
records:
- name: host1
  a_ip_address: 192.168.122.123
- name: host1
  aaaa_ip_address: ::1

6. Save the file.

7. Run the playbook:

   $ ansible-playbook -v -i inventory.file ensure-A-and-AAAA-records-are-present-copy.yml

Additional resources

- For more information on A and AAAA records, see DNS records in IdM.
- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsrecord module in the README-dnsrecord.md Markdown file available in the /usr/share/doc/ansible-freeipa/directory. The file also contains the definitions of the ipadnsrecord variables.
- You can see sample Ansible playbooks for the ipadnsrecord module in the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory.

81.4. ENSURING THE PRESENCE OF A AND PTR DNS RECORDS IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that an A record for a particular IdM host is present, with a corresponding PTR record. In the example used in the procedure below, an IdM administrator ensures the presence of A and PTR records for host1 with an IP address of 192.168.122.45 in the idm.example.com zone.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
- The idm.example.com DNS zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see Using Ansible playbooks to manage IdM DNS zones.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:
3. Make a copy of the `ensure-dnsrecord-with-reverse-is-present.yml` Ansible playbook file. For example:

$ cp ensure-dnsrecord-with-reverse-is-present.yml ensure-dnsrecord-with-reverse-is-present-copy.yml

4. Open the `ensure-dnsrecord-with-reverse-is-present-copy.yml` file for editing.

5. Adapt the file by setting the following variables in the `ipadnsrecord` task section:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - Set the `name` variable to `host1`.
   - Set the `zone_name` variable to `idm.example.com`.
   - Set the `ip_address` variable to `192.168.122.45`.
   - Set the `create_reverse` variable to `yes`.
     This is the modified Ansible playbook file for the current example:

```yaml
---
- name: Ensure DNS Record is present.
  hosts: ipaserver
  become: true
  gatherfacts: false
  tasks:
    - ipadnsrecord:
        ipaadmin_password: Secret123
        name: host1
        zone_name: idm.example.com
        ip_address: 192.168.122.45
        create_reverse: yes
        state: present

6. Save the file.

7. Run the playbook:

```bash
$ ansible-playbook -v -i inventory.file ensure-dnsrecord-with-reverse-is-present-copy.yml
```

Additional resources

- For more information on A and PTR DNS records, see [DNS records in IdM](#).
- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsrecord` module in the [README-dnsrecord.md](#) Markdown file available in the `/usr/share/doc/ansible-freeipa/` directory. The file also contains the definitions of the `ipadnsrecord` variables.
You can see sample Ansible playbooks for the `ipadnsrecord` module in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

### 81.5. ENSURING THE PRESENCE OF MULTIPLE DNS RECORDS IN IDM USING ANSIBLE

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that multiple values are associated with a particular IdM DNS record. In the example used in the procedure below, an IdM administrator ensures the presence of multiple A records for `host1` in the `idm.example.com` DNS zone.

#### Prerequisites

- You have installed the `ansible-freeipa` package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
- The `idm.example.com` zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see Using Ansible playbooks to manage IdM DNS zones.

#### Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory:
   ```sh
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
   ```
2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:
   ```
   [ipaserver]
   server.idm.example.com
   ```
3. Make a copy of the `ensure-presence-multiple-records.yml` Ansible playbook file. For example:
   ```sh
   $ cp ensure-presence-multiple-records.yml ensure-presence-multiple-records-copy.yml
   ```
4. Open the `ensure-presence-multiple-records-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipadnsrecord` task section:
   - Set the `ipaadmin_password` variable to your IdM administrator password.
   - In the `records` section, set the `name` variable to `host1`.
   - In the `records` section, set the `zone_name` variable to `idm.example.com`.
   - In the `records` section, set the `a_rec` variable to `192.168.122.112` and to `192.168.122.122`.
   - Define a second record in the `records` section:
     - Set the `name` variable to `host1`.
Set the `zone_name` variable to `idm.example.com`.

Set the `aaaa_rec` variable to `::1`.

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

```yaml
---
- name: Test multiple DNS Records are present.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
    # Ensure that multiple dns records are present
    - ipadnsrecord:
        ipaadmin_password: Secret123
        records:
          - name: host1
            zone_name: idm.example.com
            a_rec: 192.168.122.112
            a_rec: 192.168.122.122
          - name: host1
            zone_name: idm.example.com
            aaaa_rec: ::1

 6. Save the file.

7. Run the playbook:

   ```bash
   $ ansible-playbook -v -i inventory.file ensure-presence-multiple-records-copy.yml
   ```

Additional resources

- For more information on A records in DNS, see DNS records in IdM.

- You can see more sample Ansible playbooks for the `ansible-freeipa ipadnsrecord` module in the README-dnsrecord.md Markdown file available in the `/usr/share/doc/ansible-freeipa` directory. The file also contains the definitions of the `ipadnsrecord` variables.

- You can see sample Ansible playbooks for the `ipadnsrecord` module in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

81.6. ENSURING THE PRESENCE OF MULTIPLE CNAME RECORDS IN IDM USING ANSIBLE

A Canonical Name record (CNAME record) is a type of resource record in the Domain Name System (DNS) that maps one domain name, an alias, to another name, the canonical name.

You may find CNAME records useful when running multiple services from a single IP address: for example, an FTP service and a web service, each running on a different port.

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that multiple CNAME records are present in IdM DNS. In the example used in the procedure below, `host03` is both an HTTP server and an FTP server. The IdM administrator ensures the presence of CNAME records for each service.
of the www and ftp CNAME records for the host03 A record in the idm.example.com zone.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
- The idm.example.com zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see Using Ansible playbooks to manage IdM DNS zones.
- The host03 A record exists in the idm.example.com zone.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory:

   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

   [ipaserver]
   server.idm.example.com

3. Make a copy of the ensure-CNAME-record-is-present.yml Ansible playbook file. For example:

   $ cp ensure-CNAME-record-is-present.yml ensure-CNAME-record-is-present-copy.yml

4. Open the ensure-CNAME-record-is-present-copy.yml file for editing.

5. Adapt the file by setting the following variables in the ipadnsrecord task section:
   - (Optional) Adapt the description provided by the name of the play.
   - Set the ipaadmin_password variable to your IdM administrator password.
   - Set the zone_name variable to idm.example.com.
   - In the records variable section, set the following variables and values:
     - Set the name variable to www.
     - Set the cname_hostname variable to host03.
     - Set the name variable to ftp.
     - Set the cname_hostname variable to host03.

   This is the modified Ansible playbook file for the current example:
point to 'host03.idm.example.com'.
hosts: ipaserver
become: true
gather_facts: false
tasks:
  - ipadnsrecord:
      ipaadmin_password: Secret123
      zone_name: idm.example.com
      records:
        - name: www
cname_hostname: host03
        - name: ftp
cname_hostname: host03

6. Save the file.
7. Run the playbook:

$ ansible-playbook -v -i inventory.file ensure-CNAME-record-is-present.yml

Additional resources

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsrecord module in the README-dnsrecord.md Markdown file available in the /usr/share/doc/ansible-freeipa/ directory. The file also contains the definitions of the ipadnsrecord variables.

- You can see sample Ansible playbooks for the ipadnsrecord module in the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory.

81.7. ENSURING THE PRESENCE OF AN SRV RECORD IN IDM USING ANSIBLE

A DNS service (SRV) record defines the hostname, port number, transport protocol, priority and weight of a service available in a domain. In Identity Management (IdM), you can use SRV records to locate IdM servers and replicas.

This section shows how an Identity Management (IdM) administrator can use an Ansible playbook to ensure that an SRV record is present in IdM DNS. In the example used in the procedure below, an IdM administrator ensures the presence of the _kerberos._udp.idm.example.com SRV record with the value of 10 50 88 idm.example.com. This sets the following values:

- It sets the priority of the service to 10.
- It sets the weight of the service to 50.
- It sets the port to be used by the service to 88.

Prerequisites

- You have installed the ansible-freeipa package on the Ansible controller. This is the host on which you execute the steps in the procedure.
- You know the IdM administrator password.
The idm.example.com zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see Using Ansible playbooks to manage IdM DNS zones.

Procedure

1. Navigate to the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory:

   ```
   $ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
   ```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the [ipaserver] section. For example, to instruct Ansible to configure server.idm.example.com, enter:

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

3. Make a copy of the ensure-SRV-record-is-present.yml Ansible playbook file. For example:

   ```
   $ cp ensure-SRV-record-is-present.yml ensure-SRV-record-is-present-copy.yml
   ```

4. Open the ensure-SRV-record-is-present-copy.yml file for editing.

5. Adapt the file by setting the following variables in the ipadnsrecord task section:

   - Set the ipaadmin_password variable to your IdM administrator password.
   - Set the name variable to _kerberos._udp.idm.example.com.
   - Set the srv_rec variable to '10 50 88 idm.example.com'.
   - Set the zone_name variable to idm.example.com.

   This the modified Ansible playbook file for the current example:

   ```
   ---
   - name: Test multiple DNS Records are present.
     hosts: ipaserver
     become: true
     gather_facts: false
     tasks:
     # Ensure a SRV record is present
     - ipadnsrecord:
         ipaadmin_password: Secret123
         name: _kerberos._udp.idm.example.com
         srv_rec: '10 50 88 idm.example.com'
         zone_name: idm.example.com
         state: present
   ```

6. Save the file.

7. Run the playbook:

   ```
   $ ansible-playbook -v -i inventory_file ensure-SRV-record-is-present.yml
   ```
Additional resources

- For more information on SRV records, see DNS records in IdM.

- You can see more sample Ansible playbooks for the ansible-freeipa ipadnsrecord module in the README-dnsrecord.md Markdown file available in the /usr/share/doc/ansible-freeipa directory. The file also contains the definitions of the ipadnsrecord variables.

- You can see sample Ansible playbooks for the ipadnsrecord module in the /usr/share/doc/ansible-freeipa/playbooks/dnsrecord directory.
CHAPTER 82. COLLECTING IDM HEALTHCHECK INFORMATION

Healthcheck has been designed as a manual command line tool which should help you to identify possible problems in Identity Management (IdM).

This chapter describes how you can create a collection of logs based on the Healthcheck output with 30-day rotation.

82.1. HEALTHCHECK IN IDM

The Healthcheck tool in Identity Management (IdM) helps find issues that may impact the health of your IdM environment.

NOTE

The Healthcheck tool is a command line tool that can be used without Kerberos authentication.

Modules are independent

Healthcheck consists of independent modules which test for:

- Replication issues
- Certificate validity
- Certificate Authority infrastructure issues
- IdM and Active Directory trust issues
- Correct file permissions and ownership settings

Two output formats

Healthcheck generates the following outputs, which you can set using the output-type option:

- json: Machine-readable output in JSON format (default)
- human: Human-readable output

You can specify a different file destination with the --output-file option.

Results

Each Healthcheck module returns one of the following results:

SUCCESS
  configured as expected

WARNING
  not an error, but worth keeping an eye on or evaluating

ERROR
  not configured as expected

CRITICAL
82.2. LOG ROTATION

Log rotation creates a new log file every day, and the files are organized by date. Since log files are saved in the same directory, you can select a particular log file according to the date.

Rotation means that there is configured a number for max number of log files and if the number is exceeded, the newest file rewrites and renames the oldest one. For example, if the rotation number is 30, the thirty-first log file replaces the first (oldest) one.

Log rotation reduces voluminous log files and organizes them, which can help with analysis of the logs.

82.3. CONFIGURING LOG ROTATION USING THE IDM HEALTHCHECK

This section describes how to configure a log rotation with:

- the systemctl timer
- the crond service

The systemctl timer runs the Healthcheck tool periodically and generates the logs. The default value is set to 4 a.m. every day.

The crond service is used for log rotation.

The default log name is healthcheck.log and the rotated logs use the healthcheck.log-YYYYMMDD format.

Prerequisites

- You must execute commands as root.

Procedure

1. Enable a systemctl timer:

   ```bash
   # systemctl enable ipa-healthcheck.timer
   Created symlink /etc/systemd/system/multi-user.target.wants/ipa-healthcheck.timer ->
   /usr/lib/systemd/system/ipa-healthcheck.timer.
   ```

2. Start the systemctl timer:

   ```bash
   # systemctl start ipa-healthcheck.timer
   ```

3. Open the /etc/logrotate.d/ipahealthcheck file to configure the number of logs which should be saved.
   By default, log rotation is set up for 30 days.

4. In the /etc/logrotate.d/ipahealthcheck file, configure the path to the logs.
   By default, logs are saved in the /var/log/ipa/healthcheck/ directory.

5. In the /etc/logrotate.d/ipahealthcheck file, configure the time for log generation.
   By default, a log is created daily at 4 a.m.
6. To use log rotation, ensure that the `crond` service is enabled and running:

```
# systemctl enable crond
# systemctl start crond
```

To start with generating logs, start the IPA healthcheck service:

```
# systemctl start ipa-healthcheck
```

To verify the result, go to `/var/log/ipa/healthcheck/` and check if logs are created correctly.
CHAPTER 83. CHECKING SERVICES USING IDM
HEALTHCHECK

This section describes monitoring services used by the Identity Management (IdM) server using the Healthcheck tool.

For details, see Healthcheck in IdM.

83.1. SERVICES HEALTHCHECK TEST

The Healthcheck tool includes a test to check if any IdM services is not running. This test is important because services which are not running can cause failures in other tests. Therefore, check that all services are running first. You can then check all other test results.

To see all services tests, run ipa-healthcheck with the --list-sources option:

```
# ipa-healthcheck --list-sources
```

You can find all services tested with Healthcheck under the ipahealthcheck.meta.services source:

- certmonger
- dirsrv
- gssproxy
- httpd
- ipa_custodia
- ipa_dnskeysyncd
- ipa_otpd
- kadmin
- krb5kdc
- named
- pki_tomcatd
- sssd

**NOTE**

Run these tests on all IdM servers when trying to discover issues.

83.2. SCREENING SERVICES USING HEALTHCHECK

This section describes a standalone manual test of services running on the Identity Management (IdM) server using the Healthcheck tool.

The Healthcheck tool includes many tests, whose results can be shortened with:
- excluding all successful test: **--failures-only**
- including only services tests: **--source=ipahealthcheck.meta.services**

**Procedure**

- To run Healthcheck with warnings, errors and critical issues regarding services, enter:

  ```
  # ipa-healthcheck --source=ipahealthcheck.meta.services --failures-only
  ```

A successful test displays empty brackets:

```
[]
```

If one of the services fails, the result can look similarly to this example:

```
{
    "source": "ipahealthcheck.meta.services",
    "check": "httpd",
    "result": "ERROR",
    "kw": {
        "status": false,
        "msg": "httpd: not running"
    }
}
```

**Additional resources**

- For reviewing detailed reference, enter `man ipa-healthcheck` in the command line.
CHAPTER 84. VERIFYING YOUR IDM AND AD TRUST CONFIGURATION USING IDM HEALTHCHECK

This section helps you understand and use the Healthcheck tool in Identity management (IdM) to identify issues with IdM and an Active Directory trust.

84.1. IDM AND AD TRUST HEALTHCHECK TESTS

The Healthcheck tool includes several tests for testing the status of your Identity Management (IdM) and Active Directory (AD) trust.

To see all trust tests, run `ipa-healthcheck` with the `--list-sources` option:

```
# ipa-healthcheck --list-sources
```

You can find all tests under the `ipahealthcheck.ipa.trust` source:

**IPATrustAgentCheck**
This test checks the SSSD configuration when the machine is configured as a trust agent. For each domain in `/etc/sssd/sssd.conf` where `id_provider=ipa` ensure that `ipa_server_mode` is `True`.

**IPATrustDomainsCheck**
This test checks if the trust domains match SSSD domains by comparing the list of domains in `sssctl domain-list` with the list of domains from `ipa trust-find` excluding the IPA domain.

**IPATrustCatalogCheck**
This test resolves resolves an AD user, `Administrator@REALM`. This populates the AD Global catalog and AD Domain Controller values in `sssctl domain-status` output. For each trust domain look up the user with the id of the SID + 500 (the administrator) and then check the output of `sssctl domain-status <domain> --active-server` to ensure that the domain is active.

**IPAsidgenpluginCheck**
This test verifies that the `sidgen` plugin is enabled in the IPA 389-ds instance. The test also verifies that the `IPA SIDGEN` and `ipa-sidgen-task` plugins in `cn=plugins,cn=config` include the `nsslapd-pluginEnabled` option.

**IPATrustAgentMemberCheck**
This test verifies that the current host is a member of `cn=adtrust agents,cn=sysaccounts,cn=etc,SUFFIX`.

**IPATrustControllerPrincipalCheck**
This test verifies that the current host is a member of `cn=adtrust agents,cn=sysaccounts,cn=etc,SUFFIX`.

**IPATrustControllerServiceCheck**
This test verifies that the current host starts the ADTRUST service in ipactl.

**IPATrustControllerConfCheck**
This test verifies that `ldapi` is enabled for the passdb backend in the output of `net conf` list.

**IPATrustControllerGroupSIDCheck**
This test verifies that the admins group’s SID ends with 512 (Domain Admins RID).

**IPATrustPackageCheck**
This test verifies that the trust-ad package is installed if the trust controller and AD trust are not enabled.

NOTE
Run these tests on all IdM servers when trying to find an issue.

84.2. SCREENING THE TRUST WITH THE HEALTHCHECK TOOL

This section describes a standalone manual test of an Identity Management (IdM) and Active Directory (AD) trust health check using the Healthcheck tool.

The Healthcheck tool includes many tests, therefore, you can shorten the results by:

- excluding all successful test: --failures-only
- including only trust tests: --source=ipahealthcheck.ipa.trust

Procedure

- To run Healthcheck with warnings, errors and critical issues in the trust, enter:

  # ipa-healthcheck --source=ipahealthcheck.ipa.trust --failures-only

Successful test displays empty brackets:

  # ipa-healthcheck --source=ipahealthcheck.ipa.trust --failures-only
[]

Additional resources

- For reviewing detailed reference, enter man ipa-healthcheck in the command line.
CHAPTER 85. VERIFYING CERTIFICATES USING IDM HEALTHCHECK

This section helps in understanding and using the Healthcheck tool in Identity management (IdM) to identify issues with IPA certificates maintained by certmonger.

For details, see Healthcheck in IdM.

85.1. IDM CERTIFICATES HEALTHCHECK TESTS

The Healthcheck tool includes several tests for verifying the status of certificates maintained by certmonger in Identity Management (IdM). For details about certmonger, see Obtaining an IdM certificate for a service using certmonger.

This suite of tests checks expiration, validation, trust and other issues. Multiple errors may be thrown for the same underlying issue.

To see all certificate tests, run the ipa-healthcheck with the --list-sources option:

```
# ipa-healthcheck --list-sources
```

You can find all tests under the ipahealthcheck.ipa.certs source:

IPACertmongerExpirationCheck

This test checks expirations in certmonger.
If an error is reported, the certificate has expired.

If a warning appears, the certificate will expire soon. By default, this test applies within 28 days or fewer days before certificate expiration.

You can configure the number of days in the /etc/ipahealthcheck/ipahealthcheck.conf file. After opening the file, change the cert_expiration_days option located in the default section.

NOTE

Certmonger loads and maintains its own view of the certificate expiration. This check does not validate the on-disk certificate.

IPACertfileExpirationCheck

This test checks if the certificate file or NSS database cannot be opened. This test also checks expiration. Therefore, carefully read the msg attribute in the error or warning output. The message specifies the problem.

NOTE

This test checks the on-disk certificate. If a certificate is missing, unreadable, etc a separate error can also be raised.

IPACertNSSTrust
This test compares the trust for certificates stored in NSS databases. For the expected tracked certificates in NSS databases the trust is compared to an expected value and an error raised on a non-match.

**IPANSSChainValidation**

This test validates the certificate chain of the NSS certificates. The test executes: `certutil -V -u V -e -d [dbdir] -n [nickname]`

**IPAOOpenSSLChainValidation**

This test validates the certificate chain of the OpenSSL certificates. To be comparable to the NSSChain validation here is the OpenSSL command we execute:

```
openssl verify -verbose -show_chain -CAfile /etc/ipa/ca.crt [cert file]
```

**IPARAAgent**

This test compares the certificate on disk with the equivalent record in LDAP in `uid=ipara,ou=People,o=ipaca`.

**IPACertRevocation**

This test uses certmonger to verify that certificates have not been revoked. Therefore, the test can find issues connected with certificates maintained by certmonger only.

**IPACertmongerCA**

This test verifies the certmonger Certificate Authority (CA) configuration. IdM cannot issue certificates without CA.

Certmonger maintains a set of CA helpers. In IdM, there is a CA named IPA which issues certificates through IdM, authenticating as a host or user principal, for host or service certs.

There are also `dogtag-ipa-ca-renew-agent` and `dogtag-ipa-ca-renew-agent-reuse` which renew the CA subsystem certificates.

**NOTE**

Run these tests on all IdM servers when trying to check for issues.

### 85.2. SCREENING CERTIFICATES USING THE HEALTHCHECK TOOL

This section describes a standalone manual test of an Identity Management (IdM) certificate health check using the Healthcheck tool.

The Healthcheck tool includes many tests, therefore, you can shorten the results with:

- excluding all successful test: `--failures-only`
- including only certificate tests: `--source=ipahealthcheck.ipa.certs`

**Prerequisites**

- Healthcheck tests must be performed as the root user.

**Procedure**

- To run Healthcheck with warnings, errors and critical issues regarding certificates, enter:
# ipa-healthcheck --source=ipahealthcheck.ipa.certs --failures-only

Successful test displays empty brackets:

[]

Failed test shows you the following output:

```
{
  "source": "ipahealthcheck.ipa.certs",
  "check": "IPACertfileExpirationCheck",
  "result": "ERROR",
  "kw": {
    "key": 1234,
    "dbdir": "/path/to/nssdb",
    "error": [error],
    "msg": "Unable to open NSS database '/path/to/nssdb': [error]"
  }
}
```

This **IPACertfileExpirationCheck** test failed on opening the NSS database.

**Additional resources**

- For reviewing detailed reference, enter `man ipa-healthcheck` in the command line.
CHAPTER 86. VERIFYING SYSTEM CERTIFICATES USING IDM HEALTHCHECK

This section describes a Healthcheck tool in Identity Management (IdM) to identify issues with system certificates.

For details, see Healthcheck in IdM.

86.1. SYSTEM CERTIFICATES HEALTHCHECK TESTS

The Healthcheck tool includes several tests for verifying system (DogTag) certificates.

To see all tests, run the `ipa-healthcheck` with the `--list-sources` option:

```bash
# ipa-healthcheck --list-sources
```

You can find all tests under the `ipahealthcheck.dogtag.ca` source:

**DogtagCertsConfigCheck**

This test compares the CA (Certificate Authority) certificates in its NSS database to the same values stored in `CS.cfg`. If they don’t match, CA fails to start. Specifically, it checks:

- `auditSigningCert cert-pki-ca` against `ca.audit_signing.cert`
- `ocspSigningCert cert-pki-ca` against `ca.ocsp_signing.cert`
- `caSigningCert cert-pki-ca` against `ca.signing.cert`
- `subsystemCert cert-pki-ca` against `ca.subsystem.cert`
- `Server-Cert cert-pki-ca` against `ca.sslserver.cert`

If Key Recovery Authority (KRA) is installed:

- `transportCert cert-pki-kra` against `ca.connector.KRA.transportCert`

**DogtagCertsConnectivityCheck**

This test verifies connectivity. This test is equivalent to the `ipa cert-show 1` command which checks:

- The PKI proxy configuration in Apache
- IdM being able to find a CA
- The RA agent client certificate
- Correctness of CA replies to requests

Note that the test checks a certificate with serial #1 because you want to verify that a `cert-show` can be executed and get back an expected result from CA (either the certificate or a not found).
NOTE

Run these tests on all IdM servers when trying to find an issue.

86.2. SCREENING SYSTEM CERTIFICATES USING HEALTHCHECK

This section describes a standalone manual test of Identity Management (IdM) certificates using the Healthcheck tool.

Since, the Healthcheck tool includes many tests, you can narrow the results by including only DogTag tests: `--source=ipahealthcheck.dogtag.ca`

Procedure

- To run Healthcheck restricted to DogTag certificates, enter:

```
# ipa-healthcheck --source=ipahealthcheck.dogtag.ca
```

An example of a successful test:

```
{
    "source": "ipahealthcheck.dogtag.ca",
    "check": "DogtagCertsConfigCheck",
    "result": "SUCCESS",
    "uuid": "9b366200-9ec8-4bd9-bb5e-9a280c803a9c",
    "when": "20191008135826Z",
    "duration": "0.252280",
    "kw": {
        "key": "Server-Cert cert-pki-ca",
        "configfile": "/var/lib/pki/pki-tomcat/conf/ca/CS.cfg"
    }
}
```

An example of a failed test:

```
{
    "source": "ipahealthcheck.dogtag.ca",
    "check": "DogtagCertsConfigCheck",
    "result": "CRITICAL",
    "uuid": "59d66200-1447-4b3b-be01-89810c803a98",
    "when": "20191008135912Z",
    "duration": "0.002022",
    "kw": {
        "exception": "NSDB /etc/pki/pki-tomcat/alias not initialized",
    }
}
```

Additional resources

- For reviewing detailed reference, enter `man ipa-healthcheck` in the command line.
CHAPTER 87. CHECKING DISK SPACE USING IDM HEALTHCHECK

This section describes how to monitor the Identity Management server’s free disk space using the Healthcheck tool.

For details, see Healthcheck in IdM.

87.1. DISK SPACE HEALTHCHECK TEST

The Healthcheck tool includes a test for checking available disk space. Insufficient free disk space can cause issues with:

- Logging
- Execution
- Backups

The test checks the following paths:

<table>
<thead>
<tr>
<th>Paths checked by the test</th>
<th>Minimal disk space in MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>/var/lib/dirsrv/</td>
<td>1024</td>
</tr>
<tr>
<td>/var/lib/ipa/backup/</td>
<td>512</td>
</tr>
<tr>
<td>/var/log/</td>
<td>1024</td>
</tr>
<tr>
<td>var/log/audit/</td>
<td>512</td>
</tr>
<tr>
<td>/var/tmp/</td>
<td>512</td>
</tr>
<tr>
<td>/tmp/</td>
<td>512</td>
</tr>
</tbody>
</table>

To list all tests, run the `ipa-healthcheck` with the `--list-sources` option:

```
# ipa-healthcheck --list-sources
```

The file system space check test is placed under the `ipahealthcheck.system.filesystemspace` source:

**FileSystemSpaceCheck**

This test checks available disk space in the following ways:

- The minimum raw free bytes needed.
- The percentage – the minimum free disk space is hardcoded to 20%.
87.2. SCREENING DISK SPACE USING THE HEALTHCHECK TOOL

This section describes a standalone manual test of available disk space on an Identity Management (IdM) server using the Healthcheck tool.

Since Healthcheck includes many tests, you can narrow the results by:

- excluding all successful test: `--failures-only`
- including only space check tests: `--source=ipahealthcheck.system.filesystemspace`

Procedure

- To run Healthcheck with warnings, errors and critical issues regarding available disk space, enter:

  ```
  # ipa-healthcheck --source=ipahealthcheck.system.filesystemspace --failures-only
  ```

A successful test displays empty brackets:

```
[]
```

As an example, a failed test can display:

```
{
  "source": "ipahealthcheck.system.filesystemspace",
  "check": "FileSystemSpaceCheck",
  "result": "ERROR",
  "kw": {
    "msg": "/var/lib/dirsrv: free space under threshold: 0 MiB < 1024 MiB",
    "store": "/var/lib/dirsrv",
    "free_space": 0,
    "threshold": 1024
  }
}
```

The failed test informs you that the `/var/lib/dirsrv` directory has run out of space.

Additional resources

- For reviewing detailed reference, enter `man ipa-healthcheck` in the command line.
CHAPTER 88. VERIFYING PERMISSIONS OF IDM CONFIGURATION FILES USING HEALTHCHECK

This section describes how to test Identity Management (IdM) configuration files using the Healthcheck tool.

For details, see Healthcheck in IdM.

88.1. FILE PERMISSIONS HEALTHCHECK TESTS

The Healthcheck tool tests ownership and permissions of some important files installed or configured by Identity Management (IdM).

If you change the ownership or permissions of any tested file, the test returns a warning in the result section. While it does not necessarily mean that the configuration will not work, it means that the file differs from the default configuration.

To see all tests, run the ipa-healthcheck with the --list-sources option:

```
# ipa-healthcheck --list-sources
```

The file permissions test is placed under the ipahealthcheck.ipa.files source:

IPAFileNSSDBCheck

This test checks the 389-ds NSS database and the Certificate Authority (CA) database. The 389-ds database is located in `/etc/dirsrv/slapd-<dashed-REALM>` and the CA database is located in `/etc/pki/pki-tomcat/alias/`.

IPAFileCheck

This test checks the following files:

- `/var/lib/ipa/ra-agent.{key|pem}
- `/var/lib/ipa/certs/httpd.pem`
- `/var/lib/ipa/private/httpd.key`
- `/etc/httpd/alias/ipasession.key`
- `/etc/dirsrv/ds.keytab`
- `/etc/ipa/ca.crt`
- `/etc/ipa/custodia/server.keys`
  
  If PKINIT is enabled:

- `/var/lib/ipa/certs/kdc.pem`
- `/var/lib/ipa/private/kdc.key`
  
  If DNS is configured:

- `/etc/named.keytab`
- `/etc/ipa/dnssec/ipa-dnskeysyncd.keytab`
TomcatFileCheck

This test checks some tomcat-specific files if a CA is configured:

- /etc/pki/pki-tomcat/password.conf
- /var/lib/pki/pki-tomcat/conf/ca/CS.cfg
- /etc/pki/pki-tomcat/server.xml

NOTE

Run these tests on all IdM servers when trying to find issues.

88.2. SCREENING CONFIGURATION FILES USING HEALTHCHECK

This section describes a standalone manual test of an Identity Management (IdM) server’s configuration files using the Healthcheck tool.

The Healthcheck tool includes many tests. Results can be narrowed down by:

- excluding all successful test: --failures-only
- including only ownership and permissions tests: --source=ipahealthcheck.ipa.files

Procedure

1. To run Healthcheck tests on IdM configuration file ownership and permissions, while displaying only warnings, errors and critical issues, enter:

   # ipa-healthcheck --source=ipahealthcheck.ipa.files --failures-only

A successful test displays empty brackets:

```
# ipa-healthcheck --source=ipahealthcheck.ipa.files --failures-only
[]
```

Failed tests display results similar to the following WARNING:

```
{
   "source": "ipahealthcheck.ipa.files",
   "check": "IPAFileNSSDBCheck",
   "result": "WARNING",
   "kw": {
      "key": ".etc_dirsrv_slapd-EXAMPLE-TEST_pkcs11.txt_mode",
      "path": "/etc/dirsrv/slapd-EXAMPLE-TEST/pkcs11.txt",
      "type": "mode",
      "expected": "0640",
      "got": "0666",
      "msg": "Permissions of /etc/dirsrv/slapd-EXAMPLE-TEST/pkcs11.txt are 0666 and should be 0640"
   }
}
```

Additional resources
• For reviewing detailed reference material, open `man ipa-healthcheck` in the command line.
CHAPTER 89. CHECKING IDM REPLICATION USING HEALTHCHECK

This section describes how to test Identity Management (IdM) replication using the Healthcheck tool.

For details, see Healthcheck in IdM.

89.1. REPLICATION HEALTHCHECK TESTS

The Healthcheck tool tests the Identity Management (IdM) topology configuration and searches for replication conflict issues.

To list all tests, run the `ipa-healthcheck` with the `--list-sources` option:

```
# ipa-healthcheck --list-sources
```

The topology tests are placed under the `ipahealthcheck.ipa.topology` and `ipahealthcheck.ds.replication` sources:

**IPATopologyDomainCheck**

This test verifies:

- whether topology is not disconnected and there are replication paths between all servers.
- if servers don’t have more than the recommended number of replication agreements.

If the test fails, the test returns errors, such as connection errors or too many replication agreements.

If the test succeeds, the test returns the configured domains.

**NOTE**

The test runs the `ipa topologysuffix-verify` command for both the domain and ca suffixes (assuming the Certificate Authority is configured on this server).

**ReplicationConflictCheck**

The test searches for entries in LDAP matching `(&(!(objectclass=nstombstone)) (nsds5ReplConflict=*))`.

**NOTE**

Run these tests on all IdM servers when trying to check for issues.

89.2. SCREENING REPLICATION USING HEALTHCHECK

This section describes a standalone manual test of an Identity Management (IdM) replication topology and configuration using the Healthcheck tool.

The Healthcheck tool includes many tests, therefore, you can shorten the results with:

- Replication conflict test: `--source=ipahealthcheck.ds.replication`
Correct topology test: 
\[\text{--source=ipahealthcheck.ipa.topology}\]

Prerequisites

- Healthcheck tests must be performed as the root user.

Procedure

- To run Healthcheck replication conflict and topology checks, enter:
  
  ```bash
  # ipa-healthcheck --source=ipahealthcheck.ds.replication --
  source=ipahealthcheck.ipa.topology
  ```

Four different results are possible:

- **SUCCESS** — the test passed successfully.
  
  ```json
  {
    "source": "ipahealthcheck.ipa.topology",
    "check": "IPATopologyDomainCheck",
    "result": "SUCCESS",
    "kw": {}
  }
  ```

- **WARNING** — the test passed but there might be a problem.
  
  ```json
  {
    "source": "ipahealthcheck.ipa.topology",
    "check": "IPATopologyDomainCheck",
    "result": "ERROR",
    "uuid": d6ce3332-92da-423d-9818-e79f49ed321f,
    "when": 20191007115449Z,
    "duration": 0.005943,
    "kw": {
      "msg": "topologysuffix-verify domain failed, server2 is not connected
(server2_139664377356472 in MainThread)"
    }
  }
  ```

- **ERROR** — the test failed.

- **CRITICAL** — the test failed and it affects the IdM server functionality.

Additional resources

- For reviewing detailed reference, open `man ipa-healthcheck` in the command line.
CHAPTER 90. CHECKING DNS RECORDS USING IDM
HEALTHCHECK

This section describes a Healthcheck tool in Identity Management (IdM) to identify issues with DNS records.

90.1. DNS RECORDS HEALTHCHECK TEST

The Healthcheck tool includes a test for checking that the expected DNS records required for autodiscovery are resolvable.

To list all tests, run the `ipa-healthcheck` with the `--list-sources` option:

```
# ipa-healthcheck --list-sources
```

The DNS records check test is placed under the `ipahealthcheck.ipa.idns` source.

**IPADNSSystemRecordsCheck**

This test checks the DNS records from the `ipa dns-update-system-records --dry-run` command using the first resolver specified in the `/etc/resolv.conf` file. The records are tested on the IPA server.

90.2. SCREENING DNS RECORDS USING THE HEALTHCHECK TOOL

This section describes a standalone manual test of DNS records on an Identity Management (IdM) server using the Healthcheck tool.

The Healthcheck tool includes many tests. Results can be narrowed down by including only the DNS records tests by adding the `--source ipahealthcheck.ipa.idns` option.

**Prerequisites**

- Healthcheck tests must be performed as the root user.

**Procedure**

- To run the DNS records check, enter:

```
# ipa-healthcheck --source ipahealthcheck.ipa.idns
```

If the record is resolvable, the test returns **SUCCESS** as a result:

```
{
  "source": "ipahealthcheck.ipa.idns",
  "check": "IPADNSSystemRecordsCheck",
  "result": "SUCCESS",
  "uuid": "eb7a3b68-f6b2-4631-af01-798cac0eb018",
  "when": "20200415143339Z",
  "duration": "0.210471",
  "kw": {
    "key": ":_ldap._tcp.idm.example.com.:server1.idm.example.com."
  }
}
```
The test returns a **WARNING** when, for example, the number of records does not match the expected number:

```json
{
    "source": "ipahealthcheck.ipa.idns",
    "check": "IPADNSSystemRecordsCheck",
    "result": "WARNING",
    "uuid": "972b7782-1616-48e0-bd5c-49a80c257895",
    "when": "20200409100614Z",
    "duration": "0.203049",
    "kw": {
        "msg": "Got \{count\} ipa-ca A records, expected \{expected\}",
        "count": 2,
        "expected": 1
    }
}
```

**Additional resources**

- For reviewing detailed reference, enter `man ipa-healthcheck` in the command line.
CHAPTER 91. DEMOTING OR PROMOTING HIDDEN REPLICAS

After a replica has been installed, you can configure whether the replica is hidden or visible.

For details about hidden replicas, see The hidden replica mode.

If the replica is a CA renewal server, move the service to another replica before making this replica hidden. For details, see Changing and resetting IdM CA renewal server.

Procedure

- To hide the replica, enter:
  
  ```
  # ipa server-state replica.idm.example.com --state=hidden
  ```

  Alternatively, you can make the replica visible with the following command:

  ```
  # ipa server-state replica.idm.example.com --state=enabled
  ```
CHAPTER 92. IDENTITY MANAGEMENT SECURITY SETTINGS

This section describes security-related features of Identity Management.

92.1. HOW IDENTITY MANAGEMENT APPLIES DEFAULT SECURITY SETTINGS

By default, Identity Management (IdM) uses the system-wide crypto policy. The benefit of this policy is that you do not need to harden individual IdM components manually.

IMPORTANT

Red Hat recommends that you use the system-wide crypto policy. Changing individual security settings can break components of IdM.

Additional resources

- For further details about the system-wide crypto policies, see the `crypto-policies(7)` man page.

92.2. ANONYMOUS LDAP BINDS IN IDENTITY MANAGEMENT

By default, anonymous binds to the Identity Management (IdM) LDAP server are enabled. Anonymous binds can expose certain configuration settings or directory values. However, some utilities, such as `realmd`, or older RHEL clients require anonymous binds enabled to discover domain settings when enrolling a client.

Additional resources

- For details about disabling anonymous binds in the IdM LDAP server, see the Disabling Anonymous Binds section in the Red Hat Directory Server 11 Administration Guide.
CHAPTER 93. SETTING UP SAMBA ON AN IDM DOMAIN MEMBER

This section describes how to set up Samba on a host that is joined to a Red Hat Identity Management (IdM) domain. Users from IdM and also, if available, from trusted Active Directory (AD) domains, can access shares and printer services provided by Samba.

IMPORTANT

Using Samba on an IdM domain member is an unsupported Technology Preview feature and contains certain limitations. For example, due to IdM trust controllers not supporting the Global Catalog service, AD-enrolled Windows hosts cannot find IdM users and groups in Windows. Additionally, IdM Trust Controllers do not support resolving IdM groups using the Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) protocols. As a consequence, AD users can only access the Samba shares and printers from IdM clients.

Customers deploying Samba on IdM domain members are encouraged to provide feedback to Red Hat.

Prerequisites

- The host is joined as a client to the IdM domain.

93.1. PREPARING THE IDM DOMAIN FOR INSTALLING SAMBA ON DOMAIN MEMBERS

Before you can set up Samba on an IdM client, you must prepare the IdM domain using the `ipa-adtrust-install` utility on an IdM server.

NOTE

Any system where you run the `ipa-adtrust-install` command automatically becomes an AD trust controller. However, you must run `ipa-adtrust-install` only once on an IdM server.

Prerequisites

- IdM server is installed.
- You need root privileges to install packages and restart IdM services.

Procedure

1. Install the required packages:

   `[root@ipaserver ~]# dnf install ipa-server-trust-ad samba-client`

2. Authenticate as the IdM administrative user:

   `[root@ipaserver ~]# kinit admin`
3. Run the **ipa-adtrust-install** utility:

   ```bash
   [root@ipaserver ~]# ipa-adtrust-install
   ```

   The DNS service records are created automatically if IdM was installed with an integrated DNS server.

   If you installed IdM without an integrated DNS server, **ipa-adtrust-install** prints a list of service records that you must manually add to DNS before you can continue.

4. The script prompts you that the **/etc/samba/smb.conf** already exists and will be rewritten:

   ```
   WARNING: The smb.conf already exists. Running ipa-adtrust-install will break your existing Samba configuration.
   Do you wish to continue? [no]: yes
   ```

5. The script prompts you to configure the **slapi-nis** plug-in, a compatibility plug-in that allows older Linux clients to work with trusted users:

   ```
   Do you want to enable support for trusted domains in Schema Compatibility plugin? This will allow clients older than SSSD 1.9 and non-Linux clients to work with trusted users.
   Enable trusted domains support in slapi-nis? [no]: yes
   ```

6. When prompted, enter the NetBIOS name for the IdM domain or press **Enter** to accept the name suggested:

   ```
   Trust is configured but no NetBIOS domain name found, setting it now.
   Enter the NetBIOS name for the IPA domain.
   Only up to 15 uppercase ASCII letters, digits and dashes are allowed.
   Example: EXAMPLE.
   ```

   ```
   NetBIOS domain name [IDM]:
   ```

7. You are prompted to run the SID generation task to create a SID for any existing users:

   ```
   Do you want to run the ipa-sidgen task? [no]: yes
   ```

   This is a resource-intensive task, so if you have a high number of users, you can run this at another time.

8. **(Optional)** By default, the Dynamic RPC port range is defined as **49152-65535** for Windows Server 2008 and later. If you need to define a different Dynamic RPC port range for your environment, configure Samba to use different ports and open those ports in your firewall settings. The following example sets the port range to **55000-65000**.

   ```
   [root@ipaserver ~]# net conf setparm global 'rpc server dynamic port range' 55000-65000
   [root@ipaserver ~]# firewall-cmd --add-port=55000-65000/tcp
   [root@ipaserver ~]# firewall-cmd --runtime-to-permanent
   ```

9. Restart the **ipa** service:
10. Use the `smbclient` utility to verify that Samba responds to Kerberos authentication from the IdM side:

```
[root@ipaserver ~]# smbclient -L server.idm.example.com -k
lp_load_ex: changing to config backend registry
Sharename       Type      Comment
---------       ----      -------
IPC$            IPC       IPC Service (Samba 4.12.3)
...```

### 93.2. Enabling the AES Encryption Type in Active Directory Using a GPO

This section describes how to enable the AES encryption type in Active Directory (AD) using a group policy object (GPO). Certain features on RHEL, such as running a Samba server on an IdM client, require this encryption type.

Note that RHEL no longer supports the weak DES and RC4 encryption types.

**Prerequisites**

- You are logged into AD as a user who can edit group policies.
- The **Group Policy Management Console** is installed on the computer.

**Procedure**

1. Open the **Group Policy Management Console**.

2. Right-click **Default Domain Policy**, and select **Edit**. The **Group Policy Management Editor** opens.


4. Double-click the **Network security: Configure encryption types allowed for Kerberos** policy.

5. Select **AES256_HMAC_SHA1** and, optionally, **Future encryption types**.

6. Click **OK**.

7. Close the **Group Policy Management Editor**.

8. Repeat the steps for the **Default Domain Controller Policy**.

9. Wait until the Windows domain controllers (DC) applied the group policy automatically. Alternatively, to apply the GPO manually on a DC, enter the following command using an account that has administrator permissions:

```
C:\> gpupdate /force /target:computer
```
93.3. INSTALLING AND CONFIGURING A SAMBA SERVER ON AN IDM CLIENT

This section describes how to install and configure Samba on a client enrolled in an IdM domain.

Prerequisites

- The IdM domain is prepared as described in Preparing the IdM domain for installing Samba on domain members in the Deploying different types of servers documentation.

- If IdM has a trust configured with AD, enable the AES encryption type for Kerberos. For example, use a group policy object (GPO) to enable the AES encryption type. For details, see Enabling AES encryption in Active Directory using a GPO in the Deploying different types of servers documentation.

Procedure

1. Install the `ipa-client-samba` package:

   ```shell
   [root@idm_client]# dnf install ipa-client-samba
   ```

2. Use the `ipa-client-samba` utility to prepare the client and create an initial Samba configuration:

   ```shell
   [root@idm_client]# ipa-client-samba
   Searching for IPA server...
   IPA server: DNS discovery
   Chosen IPA master: idm_server.idm.example.com
   SMB principal to be created: cifs/idm_client.idm.example.com@IDM.EXAMPLE.COM
   NetBIOS name to be used: IDM_CLIENT
   Discovered domains to use:
   
   Domain name: idm.example.com
   NetBIOS name: IDM
   SID: S-1-5-21-525930803-952335037-206501584
   ID range: 212000000 - 212199999

   Domain name: ad.example.com
   NetBIOS name: AD
   SID: None
   ID range: 1918400000 - 1918599999

   Continue to configure the system with these values? [no]: yes
   Samba domain member is configured. Please check configuration at /etc/samba/smb.conf and start smb and winbind services
   ```

3. By default, `ipa-client-samba` automatically adds the `[homes]` section to the `/etc/samba/smb.conf` file that dynamically shares a user’s home directory when the user connects. If users do not have home directories on this server, or if you do not want to share them, remove the following lines from `/etc/samba/smb.conf`:

   ```
   [homes]
   read only = no
   ```
4. Share directories and printers. For details, see the following sections in the Deploying different types of servers documentation:
   - Setting up a Samba file share that uses POSIX ACLs
   - Setting up a share that uses Windows ACLs
   - Setting up Samba as a print server

5. Open the ports required for a Samba client in the local firewall:

   ```
   [root@idm_client]
   firewall-cmd --permanent --add-service=samba-client
   firewall-cmd --reload
   ```

6. Enable and start the smb and winbind services:

   ```
   [root@idm_client]
   systemctl enable --now smb winbind
   ```

Verification steps

Run the following verification steps on a different IdM domain member that has the samba-client package installed:

1. Authenticate and obtain a Kerberos ticket-granting ticket:

   ```
   $ kinit example_user
   ```

2. List the shares on the Samba server using Kerberos authentication:

   ```
   $ smbclient -L idm_client.idm.example.com -k
   lp_load_ex: changing to config backend registry
   
<table>
<thead>
<tr>
<th>Sharename</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>example</td>
<td>Disk</td>
<td></td>
</tr>
<tr>
<td>IPC$</td>
<td>IPC</td>
<td>IPC Service (Samba 4.12.3)</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
   ```

Additional resources

- For details about which steps ipa-client-samba performs during the configuration, see the ipa-client-samba(1) man page.

93.4. MANUALLY ADDING AN ID MAPPING CONFIGURATION IF IDM TRUSTS A NEW DOMAIN

Samba requires an ID mapping configuration for each domain from which users access resources. On an existing Samba server running on an IdM client, you must manually add an ID mapping configuration after the administrator added a new trust to an Active Directory (AD) domain.

Prerequisites

- You configured Samba on an IdM client. Afterward, a new trust was added to IdM.
The DES and RC4 encryption types for Kerberos must be disabled in the trusted AD domain. For security reasons, RHEL 9 does not support these weak encryption types.

**Procedure**

1. Authenticate using the host’s keytab:
   
   ```bash
   [root@idm_client]# kinit -k
   ```

2. Use the `ipa idrange-find` command to display both the base ID and the ID range size of the new domain. For example, the following command displays the values for the ad.example.com domain:
   
   ```bash
   [root@idm_client]# ipa idrange-find --name="AD.EXAMPLE.COM_id_range" --raw
   ------------
   1 range matched
   ------------
   cn: AD.EXAMPLE.COM_id_range
   ipabaseid: 1918400000
   ipaidrangesize: 200000
   ipabaserid: 0
   ipantrusteddomainsid: S-1-5-21-968346183-862388825-1738313271
   iparangetype: ipa-ad-trust
   ----------------------------
   Number of entries returned 1
   ----------------------------
   
   You need the values from the `ipabaseid` and `ipaidrangesize` attributes in the next steps.

3. To calculate the highest usable ID, use the following formula:
   
   ```
   maximum_range = ipabaseid + ipaidrangesize - 1
   ```

   With the values from the previous step, the highest usable ID for the ad.example.com domain is 1918599999 (1918400000 + 200000 - 1).

4. Edit the `/etc/samba/smb.conf` file, and add the ID mapping configuration for the domain to the `[global]` section:
   
   ```
   idmap config AD : range = 1918400000 - 1918599999
   idmap config AD : backend = sss
   ```

   Specify the value from `ipabaseid` attribute as the lowest and the computed value from the previous step as the highest value of the range.

5. Restart the `smb` and `winbind` services:
   
   ```bash
   [root@idm_client]# systemctl restart smb winbind
   ```

**Verification steps**

1. Authenticate as a user from the new domain and obtain a Kerberos ticket-granting ticket:

   ```bash
   $ kinit example_user
   ```
2. List the shares on the Samba server using Kerberos authentication:

   $ smbclient -L idm_client.idm.example.com -k
   lp_load_ex: changing to config backend registry

   Sharename  Type         Comment
     ---------  ----          -------
      example   Disk          
      IPC$      IPC           IPC Service (Samba 4.12.3)

   ...

93.5. ADDITIONAL RESOURCES

- For details about joining a RHEL host to an IdM domain, see the Installing an Identity Management client section in the Installing Identity Management guide.
Automount is a way to manage, organize, and access directories across multiple systems. Automount program automatically mounts a directory whenever access to it is requested. This works well within an IdM domain since it allows directories on clients within the domain to be shared easily. This is especially important with user home directories.

In IdM, automount works with the internal LDAP directory and also with DNS services if configured.

**94.1. SETTING UP A KERBEROS-AWARE NFS SERVER**

This procedure describes how to set up a Kerberos-aware NFS server.

**Prerequisites**

- IdM domain set up. For more information, see [Installing Identity Management](#).
- IPA client installed. For more information, see [Installing ipa-client packages](#).

**Procedure**

1. If any of your NFS clients support only weak cryptography, such as Red Hat Enterprise Linux 5 clients:
   a. Update the IdM server Kerberos configuration to enable the weak `des-cbc-crc` encryption type:

```bash
$ ldapmodify -x -D "cn=directory manager" -w password -h ipaserver.example.com -p 389
   
   dn: cn=REALM_NAME, cn=kerberos, dc=example, dc=com
   changetype: modify
   add: krbSupportedEncSaltTypes
   krbSupportedEncSaltTypes: des-cbc-crc:normal
   
   add: krbSupportedEncSaltTypes
   krbSupportedEncSaltTypes: des-cbc-crc:special
```

   b. On the NFS server, add the following entry to the `/etc/krb5.conf` file of the NFS server enable weak cryptography support:

   ```ini
   allow_weak_crypto = true
   ```
WARNING

NFS Clients prior to RHEL6 require enabling the des-cbc-crc weak encryption type. DES handles only 56-bit keys by design which are not large enough for modern computing power. DES is therefore considered weak encryption now. Only execute the previous two steps if supporting older NFS clients is required; consider retiring or upgrading the clients instead.

2. Obtain a Kerberos ticket:

   [root@nfs-server ~]# kinit admin

3. If the NFS host machine has not been added as a client to the IdM domain, create the host entry. See Adding IdM host entries from IdM CLI.

4. Create the NFS service entry:

   [root@nfs-server ~]# ipa service-add nfs/nfs-server.example.com

5. Retrieve an NFS service keytab for the NFS server using the following `ipa-getkeytab` command that saves the keys in the `/etc/krb5.keytab` file:

   [root@nfs-server ~]# ipa-getkeytab -s ipaserver.example.com -p nfs/nfs-server.example.com -k /etc/krb5.keytab

   If any of your NFS clients support only weak cryptography, additionally pass the `-e des-cbc-crc` option to the command to request a DES-encrypted keytab.

6. Verify that the NFS service has been properly configured in IdM, with its keytab, by checking the service entry:

   [root@nfs-server ~]# ipa service-show nfs/nfs-server.example.com
   Principal name: nfs/nfs-server.example.com@IDM.EXAMPLE.COM
   Principal alias: nfs/nfs-server.example.com@IDM.EXAMPLE.COM
   Keytab: True
   Managed by: nfs-server.example.com

7. Install the `nfs-utils` package:

   [root@nfs-server ~]# dnf install nfs-utils

8. Run the `ipa-client-automount` utility to configure the NFS settings:

   [root@nfs-server ~] ipa-client-automount
   Searching for IPA server...
   IPA server: DNS discovery
   Location: default
   Continue to configure the system with these values? [no]: yes
Configured /etc/idmapd.conf
Restarting sssd, waiting for it to become available.
Started autofs

By default, this command enables secure NFS and sets the Domain parameter in the
/etc/idmapd.conf file to the IdM DNS domain. If you use a different domain, specify it using the
--idmap-domain domain_name parameter.

9. Edit the /etc/exports file and add shares with the krb5p Kerberos security setting:

```
/export *(rw,sec=krb5:krb5i:krb5p)
/home *(rw,sec=krb5:krb5i:krb5p)
```

This example shares the /export and /home directories in read-write mode with Kerberos
authentication enabled.

10. Restart and enable nfs-server:

```
[root@nfs-server ~]# systemctl restart nfs-server
[root@nfs-server ~]# systemctl enable nfs-server
```

11. Re-export the shared directories:

```
[root@nfs-server ~]# exportfs -rav
```

12. Optionally, configure the NFS server as an NFS client. See Section 94.2, "Setting up a
Kerberos-aware NFS client".

### 94.2. SETTING UP A KERBEROS-AWARE NFS CLIENT

This procedure describes how to set up a kerberos-aware NFS client.

#### Prerequisites

- IdM domain set up. For more information, see Installing Identity Management.
- IPA client installed. For more information, see Installing ipa-client packages.

#### Procedure

1. If the NFS clients supports only weak cryptography, such as a Red Hat Enterprise Linux 5 client,
set the following entry in the /etc/krb5.conf file of the server to allow weak cryptography:

```
allow_weak_crypto = true
```

2. If the NFS client is not enrolled as a client in the IdM domain, set up the required host entries, as
described in Adding IdM host entries from IdM CLI.

3. Install the nfs-utils package:

```
[root@nfs-client ~]# dnf install nfs-utils
```

4. Obtain a Kerberos ticket before running IdM tools.
5. Run the `ipa-client-automount` utility to configure the NFS settings:

```
[root@nfs-client ~]# ipa-client-automount
Searching for IPA server...
IPA server: DNS discovery
Location: default
Continue to configure the system with these values? [no]: yes
Configured /etc/idmapd.conf
Restarting sssd, waiting for it to become available.
Started autofs
```

By default, this enables secure NFS in the `/etc/sysconfig/nfs` file and sets the IdM DNS domain in the `Domain` parameter in the `/etc/idmapd.conf` file.

6. Add the following entries to the `/etc/fstab` file to mount the NFS shares from the `nfs-server.example.com` host when the system boots:

```
nfs-server.example.com:/export /mnt nfs4  sec=krb5p,rw
nfs-server.example.com:/home  /home  nfs4  sec=krb5p,rw
```

These settings configure Red Hat Enterprise Linux to mount the `/export` share to the `/mnt` and the `/home` share to the `/home` directory.

7. Create the mount points if they do not exist. In our case both should exist.

8. Mount the NFS shares:

```
[root@nfs-client ~]# mount /mnt/
[root@nfs-client ~]# mount /home
```

The command uses the information from the `/etc/fstab` entry.

9. Configure SSSD to renew Kerberos tickets:

   a. Set the following parameters in the IdM domain section of the `/etc/sssd/sssd.conf` file to configure SSSD to automatically renew tickets:

```
/domain/EXAMPLE.COM
...
krb5_renewable_lifetime = 50d
krb5_renew_interval = 3600
```

   b. Restart SSSD:

```
[root@nfs-client ~]# systemctl restart sssd
```

**IMPORTANT**

The `pam_oddjob_mkhome` module does not support automatic creation of home directories on an NFS share. Therefore, you must manually create the home directories on the server in the root of the share that contains the home directories.
CHAPTER 95. MIGRATING FROM NIS TO IDENTITY MANAGEMENT

A Network Information Service (NIS) server can contain information about users, groups, hosts, netgroups and automount maps. As a system administrator you can migrate these entry types, authentication, and authorization from NIS server to an Identity Management (IdM) server so that all user management operations are performed on the IdM server. Migrating from NIS to IdM will also allow you access to more secure protocols such as Kerberos.

95.1. ENABLING NIS IN IDM

To allow communication between NIS and Identity Management (IdM) server, you must enable NIS compatibility options on IdM server.

Prerequisites

- You have root access on IdM server.

Procedure

1. Enable the NIS listener and compatibility plug-ins on IdM server:

   [root@ipaserver ~]# ipa-nis-manage enable
   [root@ipaserver ~]# ipa-compat-manage enable

2. Optional: For a more strict firewall configuration, set a fixed port. For example, to set the port to unused port 514:

   [root@ipaserver ~]# ldapmodify -x -D 'cn=directory manager' -W
   dn: cn=NIS Server,cn=plugins,cn=config
   changetype: modify
   add: nsslapd-pluginarg0
   nsslapd-pluginarg0: 514

   **WARNING**
   To avoid conflict with other services do not use any port number above 1024.

3. Enable and start the port mapper service:

   [root@ipaserver ~]# systemctl enable rpcbind.service
   [root@ipaserver ~]# systemctl start rpcbind.service

4. Restart Directory Server:

   [root@ipaserver ~]# systemctl restart dirsrv.target
95.2. MIGRATING USER ENTRIES FROM NIS TO IDM

The NIS passwd map contains information about users, such as names, UIDs, primary group, GECOS, shell, and home directory. Use this data to migrate NIS user accounts to Identity Management (IdM):

Prerequisites

- You have root access on NIS server.
- NIS is enabled in IdM.
- The NIS server is enrolled into IdM.

Procedure

1. Install the yp-tools package:

   [root@nis-server ~]# dnf install yp-tools -y

2. On the NIS server create the /root/nis-users.sh script with the following content:

   ```
   #!/bin/sh
   # $1 is the NIS domain, $2 is the NIS master server
   ypcat -d $1 -h $2 passwd > /dev/shm/nis-map.passwd 2>&1
   IFS=$'\n'
   for line in $(cat /dev/shm/nis-map.passwd) ; do
     IFS=''
     username=$(echo $line | cut -f1 -d:)
     # Not collecting encrypted password because we need cleartext password
     # to create kerberos key
     uid=$(echo $line | cut -f3 -d:)
     gid=$(echo $line | cut -f4 -d:)
     gecos=$(echo $line | cut -f5 -d:)
     homedir=$(echo $line | cut -f6 -d:)
     shell=$(echo $line | cut -f7 -d:)
     # Now create this entry
     echo passw0rd1 | ipa user-add $username --first=NIS --last=USER
     --password --gidnumber=$gid --uid=$uid --gecos='$gecos' --homedir=$homedir
     --shell=$shell
     ipa user-show $username
   done
   ```

3. Authenticate as the IdM admin user:

   [root@nis-server ~]# kinit admin

4. Run the script. For example:

   [root@nis-server ~]# sh /root/nis-users.sh nisdomain nis-server.example.com
IMPORTANT

This script uses hard-coded values for first name, last name, and sets the password to **passw0rd1**. The user must change the temporary password at the next login.

### 95.3. MIGRATING USER GROUP FROM NIS TO IDM

The NIS group map contains information about groups, such as group names, GIDs, or group members. Use this data to migrate NIS groups to Identity Management (IdM):

**Prerequisites**

- You have root access on NIS server.
- NIS is enabled in IdM.
- The NIS server is enrolled into IdM.

**Procedure**

1. Install the **yp-tools** package:

   ```
   [root@nis-server ~]# dnf install yp-tools -y
   ```

2. Create the `/root/nis-groups.sh` script with the following content on the NIS server:

   ```bash
   #!/bin/sh
   # $1 is the NIS domain, $2 is the NIS master server
   ypcat -d $1 -h $2 group > /dev/shm/nis-map.group 2>&1
   
   IFS=$'n'
   for line in $(cat /dev/shm/nis-map.group); do
     IFS=' ' 
     groupname=$(echo $line | cut -f1 -d:)
     gid=$(echo $line | cut -f3 -d:)
     members=$(echo $line | cut -f4 -d:)
     
     # Not collecting encrypted password because we need cleartext password
     # to create kerberos key
     #ipa group-add $groupname --desc=NIS_GROUP_$groupname --gid=$gid
     ipa group-add-member $groupname --users={$members}
   done
   
3. Authenticate as the IdM **admin** user:

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

4. Run the script. For example:
95.4. MIGRATING HOST ENTRIES FROM NIS TO IDM

The NIS hosts map contains information about hosts, such as host names and IP addresses. Use this data to migrate NIS host entries to Identity Management (IdM):

**NOTE**

When you create a host group in IdM, a corresponding shadow NIS group is automatically created. Do not use the `ipa netgroup-*` commands on these shadow NIS groups. Use the `ipa netgroup-*` commands only to manage native netgroups created via the `netgroup-add` command.

**Prerequisites**

- You have root access on NIS server.
- NIS is enabled in IdM.
- The NIS server is enrolled into IdM.

**Procedure**

1. Install the `yp-tools` package:

   ```bash
   [root@nis-server ~]# dnf install yp-tools -y
   ```

2. Create the `/root/nis-hosts.sh` script with the following content on the NIS server:

   ```bash
   #!/bin/sh
   # $1 is the NIS domain, $2 is the NIS master server
   ypcat -d $1 -h $2 hosts | egrep -v "localhost|127.0.0.1" > /dev/shm/nis-map.hosts 2>&1
   IFS=$'\n'
   for line in $(cat /dev/shm/nis-map.hosts); do
     IFS=' '
     ipaddress=$(echo $line | awk '{print $1}')
     hostname=$(echo $line | awk '{print $2}')
     master=$(ipa env xmlrpc_uri | tr -d '[:space:]' | cut -f3 -d: | cut -f3 -d/)
     domain=$(ipa env domain | tr -d '[:space:]' | cut -f2 -d/)
     if [ $(echo $hostname | grep "." | wc -l) -eq 0 ]; then
       hostname=$(echo $hostname.$domain)
     fi
     zone=$(echo $hostname | cut -f2- -d.)
     if [ $(ipa dnszone-show $zone 2>/dev/null | wc -l) -eq 0 ]; then
       ipa dnszone-add --name-server=$master --admin-email=root.$master
     fi
     ptrzone=$(echo $ipaddress | awk -F. '{print $3 ". "$2 ". "$1 ".in-addr.arpa."}"
     if [ $(ipa dnszone-show $ptrzone 2>/dev/null | wc -l) -eq 0 ]; then
       ipa dnszone-add $ptrzone --name-server=$master --admin-email=root.$master
     fi
   done
   # Now create this entry
   ```
3. Authenticate as the IdM admin user:

   [root@nis-server ~]# kinit admin

4. Run the script. For example:

   [root@nis-server ~]# sh /root/nis-hosts.sh nisdomain nis-server.example.com

   **NOTE**
   
   This script does not migrate special host configurations, such as aliases.

### 95.5. MIGRATING NETGROUP ENTRIES FROM NIS TO IDM

The NIS netgroup map contains information about netgroups. Use this data to migrate NIS netgroups to Identity Management (IdM):

**Prerequisites**

- You have root access on NIS server.
- NIS is enabled in IdM.
- The NIS server is enrolled into IdM.

**Procedure**

1. Install the `yp-tools` package:

   [root@nis-server ~]# dnf install yp-tools -y

2. Create the `/root/nis-netgroups.sh` script with the following content on the NIS server:

```bash
#!/bin/sh
# $1 is the NIS domain, $2 is the NIS master server
ypcat -k -d $1 -h $2 netgroup > /dev/shm/nis-map.netgroup 2>&1
IFS="\n"
for line in $(cat /dev/shm/nis-map.netgroup); do
  IFS=""
  netgroupname=$(echo $line | awk '{print $1}')
  triples=$(echo $line | sed "s/\"$netgroupname \"/\"")
  echo "ipa netgroup-add $netgroupname --desc=NIS_NG_$netgroupname"
  if [ $(echo $line | grep "\," | wc -l) -gt 0 ]; then
    echo "ipa netgroup-mod $netgroupname --hostcat=all"
  fi
  if [ $(echo $line | grep ",,\," | wc -l) -gt 0 ]; then
    echo "ipa netgroup-mod $netgroupname --usercat=all"
  fi
```

---

CHAPTER 95. MIGRATING FROM NIS TO IDENTIITY MANAGEMENT

727
for triple in $triples; do
triple=$(echo $triple | sed -e 's/-//g' -e 's/(// -e 's/)//')
if [ $(echo $triple | grep ",.*," | wc -l) -gt 0 ]; then
    hostname=$(echo $triple | cut -f1 -d,)
    username=$(echo $triple | cut -f2 -d,)
    domain=$(echo $triple | cut -f3 -d,)
    hosts=""; users=""; doms="";
    [ -n "$hostname" ] && hosts="--hosts=$hostname"
    [ -n "$username" ] && users="--users=$username"
    [ -n "$domain" ] && doms="--nisdomain=$domain"
    echo "ipa netgroup-add-member $hosts $users $doms"
else
    netgroup=$triple
    echo "ipa netgroup-add $netgroup --desc=NIS_NG_$netgroup"
fi
done

3. Authenticate as the IdM admin user:

[root@nis-server ~]# kinit admin

4. Run the script. For example:

[root@nis-server ~]# sh /root/nis-netgroups.sh nisdomain nis-server.example.com

95.6. MIGRATING AUTOMOUNT MAPS FROM NIS TO IDM

Automount maps are a series of nested and interrelated entries that define the location (the parent entry), the associated keys, and maps. To migrate NIS automount maps to Identity Management (IdM):

Prerequisites

- You have root access on NIS server.
- NIS is enabled in IdM.
- The NIS server is enrolled into IdM.

Procedure

1. Install the yp-tools package:

   [root@nis-server ~]# dnf install yp-tools -y

2. Create the /root/nis-automounts.sh script with the following content on the NIS server:

   #!/bin/sh
   # $1 is for the automount entry in ipa
   ipa automountlocation-add $1

   # $2 is the NIS domain, $3 is the NIS master server, $4 is the map name
ypcat -k -d $2 -h $3 $4 > /dev/shm/nis-map.$4 2>&1

ipa automountmap-add $1 $4

basedn=$(ipa env basedn | tr -d '[:space:]' | cut -f2 -d:)
cat > /tmp/amap.ldif <<EOF
dn: nis-domain=$2+nis-map=$4,cn=NIS Server,cn=plugins,cn=config
objectClass: extensibleObject
nis-domain: $2
nis-map: $4
nis-base: automountmapname=$4, cn=$1, cn=automount, $basedn
nis-filter: (objectclass=*)
nis-key-format: %{automountKey}
nis-value-format: %{automountInformation}
EOF
ldapadd -x -h $3 -D "cn=Directory Manager" -W -f /tmp/amap.ldif

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.$4); do
  IFS=" ">
  key=$(echo "$line" | awk '{print $1}"
  info=$(echo "$line" | sed -e "s\$key\[ \t\]\""
  ipa automountkey-add nis $4 --key="$key" --info="$info"
done

NOTE

The script exports the NIS automount information, generates an LDAP Data Interchange Format (LDIF) for the automount location and associated map, and imports the LDIF file into the IdM Directory Server.

3. Authenticate as the IdM admin user:

   [root@nis-server ~]# kinit admin

4. Run the script. For example:

   [root@nis-server ~]# sh /root/nis-automounts.sh location nisdomain
                 nis-server.example.com map_name