Red Hat Enterprise Linux 9

Using external Red Hat utilities with Identity Management

A guide for using other Red Hat utilities, such as Satellite, Open Shift, and Samba, with Identity Management.

Last Updated: 2022-08-30

Red Hat Enterprise Linux 9 Using external Red Hat utilities with Identity Management

A guide for using other Red Hat utilities, such as Satellite, Open Shift, and Samba, with Identity Management.

Legal Notice

Abstract

This documentation collection provides instructions on how to integrate your Identity Management deployment with Ansible Automation Platform, NFS, OpenShift Container Platform, OpenStack Platform, Samba, Satellite, Single Sign-On, and Virtualization.

Table of Contents

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 feedback on our documentation. Let us know how we can improve it.

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the Bugzilla website.
2. Select the correct version from the **Version** menu.
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Submit Bug**.

CHAPTER 1. IDM INTEGRATION WITH OTHER RED HAT PRODUCTS

This section provides links to documentation for other Red Hat products that integrate with IdM. You can configure these products to allow your IdM users to access their services.

Ansible Automation Platform

Setting up LDAP authentication

OpenShift Container Platform

Configuring an LDAP identity provider

OpenStack Platform

Integrating OpenStack Identity (keystone) with Red Hat Identity Manager (IdM)

Satellite

Using Red Hat Identity Management

Single Sign-On

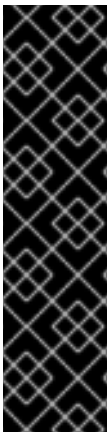
SSSD and FreeIPA Identity Management integration

Virtualization

Configuring an external LDAP provider

CHAPTER 2. 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, IdM trust controllers do not support the Active Directory Global Catalog service, and they 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 Samba shares and printers hosted on IdM clients when logged in to other IdM clients; AD users logged into a Windows machine can not access Samba shares hosted on an IdM domain member.

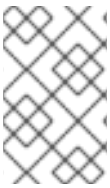
Customers deploying Samba on IdM domain members are encouraged to provide feedback to Red Hat.

Prerequisites

- The host is joined as a client to the IdM domain.

2.1. 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:

```
[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:

```
[root@ipaserver ~]# ipactl restart
```

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 -U user_name --use-kerberos=required
lp_load_ex: changing to config backend registry
   Sharename       Type      Comment
   ---------       ----      -------
   IPC$            IPC       IPC Service (Samba 4.15.2)
...
```

2.2. 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.
3. Navigate to **Computer Configuration → Policies → Windows Settings → Security Settings → Local Policies → Security Options**.
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
```

2.3. 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

- Both the IdM servers and the client must run on RHEL 9.0 or later.
- The IdM domain is prepared as described in Preparing the IdM domain for installing Samba on domain members.
- 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 .
- The IdM domain is prepared as described in Preparing the IdM domain for installing Samba on domain members.
- 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 .

Procedure

1. Install the **ipa-client-samba** package:

```
[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:

```
[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:
   - 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
[root@idm_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 step on a different IdM domain member that has the **samba-client** package installed:

- List the shares on the Samba server using Kerberos authentication:

```
$ smbclient -L idm_client.idm.example.com -U user_name --use-kerberos=required
lp_load_ex: changing to config backend registry

   Sharename       Type      Comment
   ---------       ----      -------
   example         Disk
   IPC$            IPC       IPC Service (Samba 4.15.2)
...
```

Additional resources

- **ipa-client-samba(1)** man page.

2.4. 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:

```
[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:

```
[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
  ipanttrusteddomainsid: 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:

```
[root@idm_client]# systemctl restart smb winbind
```

Verification steps

- List the shares on the Samba server using Kerberos authentication:

```
$ smbclient -L idm_client.idm.example.com -U user_name --use-kerberos=required
```

```
lp_load_ex: changing to config backend registry

    Sharename       Type      Comment
    ---------       ----      -------
    example         Disk
    IPC$            IPC       IPC Service (Samba 4.15.2)
...
```

2.5. ADDITIONAL RESOURCES

- See Installing an Identity Management client

CHAPTER 3. 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.

3.1. 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
```

3.2. 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:

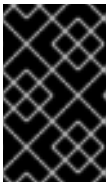
```
#!/bin/sh
# $1 is the NIS domain, $2 is the NIS master server
ypcat -d $1 -h $2 passwd > /dev/shm/nis-map.passwd 2>&1

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.

3.3. 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:

```
#!/bin/sh
# $1 is the NIS domain, $2 is the NIS master server
ypcat -d $1 -h $2 group > /dev/shm/nis-map.group 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.group); do
 IFS=' '
 groupname=$(echo $line | cut -f1 -d:)
 # Not collecting encrypted password because we need cleartext password
 # to create kerberos key
 gid=$(echo $line | cut -f3 -d:)
 members=$(echo $line | cut -f4 -d:)

 # Now create this entry
 ipa group-add $groupname --desc=NIS_GROUP_$groupname --gid=$gid
 if [ -n "$members" ]; then
  ipa group-add-member $groupname --users={$members}
 fi
 ipa group-show $groupname
done
```

3. Authenticate as the IdM **admin** user:

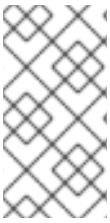
```
[root@nis-server ~]# kinit admin
```

4. Run the script. For example:

```
[root@nis-server ~]# sh /root/nis-groups.sh nisdomain nis-server.example.com
```

3.4. 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:

```
[root@nis-server ~]# dnf install yp-tools -y
```

2. Create the **/root/nis-hosts.sh** script with the following content on the NIS server:

```
#!/bin/sh
# $1 is the NIS domain, $2 is the NIS master server
ypcat -d $1 -h $2 hosts | egrep -v "localhost|127.0.0.1" > /dev/shm/nis-map.hosts 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.hosts); do
 IFS=' '
 ipaddress=$(echo $line | awk '{print $1}')
 hostname=$(echo $line | awk '{print $2}')
 master=$(ipa env xmlrpc_uri | tr -d '[:space:]' | cut -f3 -d: | cut -f3 -d/)
 domain=$(ipa env domain | tr -d '[:space:]' | cut -f2 -d:)
 if [ $(echo $hostname | grep "\." |wc -l) -eq 0 ] ; then
  hostname=$(echo $hostname.$domain)
 fi
 zone=$(echo $hostname | cut -f2- -d.)
 if [ $(ipa dnszone-show $zone 2>/dev/null | wc -l) -eq 0 ] ; then
  ipa dnszone-add --name-server=$master --admin-email=root.$master
 fi
 ptrzone=$(echo $ipaddress | awk -F. '{print $3 "." $2 "." $1 ".in-addr.arpa."}')
 if [ $(ipa dnszone-show $ptrzone 2>/dev/null | wc -l) -eq 0 ] ; then
  ipa dnszone-add  $ptrzone --name-server=$master --admin-email=root.$master
 fi
 # Now create this entry
```

```
 ipa host-add $hostname --ip-address=$ipaddress
 ipa host-show $hostname
done
```

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.

```
 :_content-type: PROCEDURE
// Module included in the following assemblies:
//
// assembly_migrating-from-nis-to-identity-management.adoc
```

3.5. 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:

```
#!/bin/sh
# $1 is the NIS domain, $2 is the NIS master server
ypcat -k -d $1 -h $2 netgroup > /dev/shm/nis-map.netgroup 2>&1

IFS=$'\n'
for line in $(cat /dev/shm/nis-map.netgroup); do
 IFS=' '
 netgroupname=$(echo $line | awk '{print $1}')
 triples=$(echo $line | sed "s/^$netgroupname //")
 echo "ipa netgroup-add $netgroupname --desc=NIS_NG_$netgroupname"
```

```
 if [ $(echo $line | grep "(," | wc -l) -gt 0 ]; then
  echo "ipa netgroup-mod $netgroupname --hostcat=all"
 fi
 if [ $(echo $line | grep ",," | wc -l) -gt 0 ]; then
  echo "ipa netgroup-mod $netgroupname --usercat=all"
 fi

 for triple in $triples; do
  triple=$(echo $triple | sed -e 's/-//g' -e 's/(//' -e 's/)//')
  if [ $(echo $triple | grep ",.*," | wc -l) -gt 0 ]; then
   hostname=$(echo $triple | cut -f1 -d,)
   username=$(echo $triple | cut -f2 -d,)
   domain=$(echo $triple | cut -f3 -d,)
   hosts=""; users=""; doms="";
   [ -n "$hostname" ] && hosts="--hosts=$hostname"
   [ -n "$username" ] && users="--users=$username"
   [ -n "$domain"   ] && doms="--nisdomain=$domain"
   echo "ipa netgroup-add-member $netgroup $hosts $users $doms"
  else
   netgroup=$triple
   echo "ipa netgroup-add $netgroup --desc=<NIS_NG>_$netgroup"
  fi
 done
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
```

3.6. 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
```

CHAPTER 4. USING AUTOMOUNT IN IDM

Automount is a way to manage, organize, and access directories across multiple systems. Automount automatically mounts a directory whenever access to it is requested. This works well within an Identity Management (IdM) domain as it allows you to share directories on clients within the domain easily.

The example uses the following scenario:

- **nfs-server.idm.example.com** is the fully-qualified domain name (FQDN) of a Network File System (NFS) server.
- For the sake of simplicity, **nfs-server.idm.example.com** is an IdM client that provides the maps for the **raleigh** automount location.

NOTE

An automount location is a unique set of NFS maps. Ideally, these maps are all located in the same geographical region so that, for example, the clients can benefit from fast connections, but this is not mandatory.

- The NFS server exports the **/exports/project** directory as read-write.
- Any IdM user belonging to the **developers** group can access the contents of the exported directory as **/devel/project/** on any IdM client that uses the **raleigh** automount location.
- **idm-client.idm.example.com** is an IdM client that uses the **raleigh** automount location.

IMPORTANT

If you want to use a Samba server instead of an NFS server to provide the shares for IdM clients, see the How do I configure kerberized CIFS mounts with Autofs in an IPA environment? KCS solution.

4.1. AUTOFS AND AUTOMOUNT IN IDM

The **autofs** service automates the mounting of directories, as needed, by directing the **automount** daemon to mount directories when they are accessed. In addition, after a period of inactivity, **autofs** directs **automount** to unmount auto-mounted directories. Unlike static mounting, on-demand mounting saves system resources.

Automount maps

On a system that utilizes **autofs**, the **automount** configuration is stored in several different files. The primary **automount** configuration file is **/etc/auto.master**, which contains the master mapping of **automount** mount points, and their associated resources, on a system. This mapping is known as *automount maps*.

The **/etc/auto.master** configuration file contains the *master map*. It can contain references to other maps. These maps can either be direct or indirect. Direct maps use absolute path names for their mount points, while indirect maps use relative path names.

Automount configuration in IdM

While **automount** typically retrieves its map data from the local **/etc/auto.master** and associated files, it can also retrieve map data from other sources. One common source is an LDAP server. In the context of Identity Management (IdM), this is a 389 Directory Server.
If a system that uses **autofs** is a client in an IdM domain, the **automount** configuration is not stored in local configuration files. Instead, the **autofs** configuration, such as maps, locations, and keys, is stored as LDAP entries in the IdM directory. For example, for the **idm.example.com** IdM domain, the default *master map* is stored as follows:

```
dn:
automountmapname=auto.master,cn=default,cn=automount,dc=idm,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master
```

Additional resources

- Mounting file systems on demand

4.2. CONFIGURING AN IDM KEYTAB FOR AN NFS SERVER

Configure a Kerberos-aware NFS server so that users logged in to other Identity Management (IdM) clients can access directories and files on this NFS server.

The example describes how to configure NFS service running on **nfs-server.idm.example.com**.

Prerequisites

- You are logged in as an IdM administrator.
- You have **root** access to the NFS server.
- You have installed the required packages to export NFS shares .
- [Optional] You have configured the NFS server to run behind a firewall .

Procedure

1. On any IdM-enrolled host, add the NFS service to IdM:

```
$ ipa service-add nfs/nfs-server.idm.example.com
---------------------------------------------------------
Added service "nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM"
---------------------------------------------------------
  Principal name: nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM
  Principal alias: nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM
  Managed by: nfs-server.idm.example.com
```

2. On the NFS server, obtain the keytab for the NFS service:

```
# ipa-getkeytab -p nfs/nfs-server.idm.example.com -k /etc/krb5.keytab
Keytab successfully retrieved and stored in: /etc/krb5.keytab
```

3. On the NFS server, restart the NFS service:

```
# systemctl restart nfs-server
```

4. On the NFS server, enable the NFS service:

```
# systemctl enable nfs-server
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service →
/usr/lib/systemd/system/nfs-server.service.
```

4.3. EXPORTING NFS SHARES IN IDM

As an Identity Management (IdM) system administrator, you can use an NFS server to share a directory with IdM users over the network.

Prerequisites

- You are logged in as an IdM administrator.
- You have **root** access to the NFS server.
- You have installed the required packages to export NFS shares .
- [Optional] You have configured the NFS server to run behind a firewall .

Procedure

1. Create the directory you want to export:

```
# mkdir -p /exports/project
```

2. Give the owner and group the rights to read, write and execute the directory:

```
# chmod 770 /exports/project
```

3. Add the **GSID** sticky bit so that any files created in the directory will have their group ownership set to that of the directory owner:

```
# chmod g+s /exports_ro/documentation /exports/project
```

4. Create an IdM group whose members will be able to access the directory. The example IdM group is **developers**:

```
# ipa group-add developers
```

5. Change the group ownership of the **/exports/project** directory to **developers** so that every IdM user in the group can access it:

```
# chgrp developers /exports/project
```

6. Add an IdM user to the group. The example user is **idm_user**:

```
# ipa group-add-member developers --users=idm_user
```

7. Create a file in the directory with some content:

```
# echo "this is a read-write file" > /exports/project/rw_file
```

8. To a file in the **/etc/exports.d/** directory, add the following information:

   - Which directory you want to export
   - How you want users to authenticate to access the files in the directory
   - What permissions you want users to have on the files in the directory

```
# echo "/exports/project *(sec=krb5p,rw)" > /etc/exports.d/project.exports
```

   sec=krb5 uses the Kerberos V5 protocol instead of local UNIX UIDs and GIDs to authenticate users.

   Alternatively, use **sec=krb5i** or **sec=krb5p**:

   sec=krb5i

   uses Kerberos V5 for user authentication and performs integrity checking of NFS operations using secure checksums to prevent data tampering.

   sec=krb5p

   uses Kerberos V5 for user authentication, integrity checking, and encrypts NFS traffic to prevent traffic sniffing. This is the most secure setting, but it also involves the most performance overhead.

9. Reexport all directories, synchronizing the master export table kept in **/var/lib/nfs/etab** with **/etc/exports** and files under **/etc/exports.d**:

```
# exportfs -r
```

10. Display the current export list suitable for **/etc/exports**:

```
# exportfs -s
/exports/project  *
(sync,wdelay,hide,no_subtree_check,sec=krb5p,rw,secure,root_squash,no_all_squash)
```

Additional resources

- For information on **krb5** methods, see the **nfs** man page.

4.4. CONFIGURING AUTOMOUNT LOCATIONS AND MAPS IN IDM USING THE IDM CLI

A location is a set of maps, which are all stored in **auto.master**. A location can store multiple maps. The location entry only works as a container for map entries; it is not an automount configuration in and of itself.

As a system administrator in Identity Management (IdM), you can configure automount locations and maps in IdM so that IdM users in the specified locations can access shares exported by an NFS server by navigating to specific mount points on their hosts. Both the exported NFS server directory and the

mount points are specified in the maps. The example describes how to configure the **raleigh** location and a map that mounts the **nfs-server.idm.example.com:/exports/project** share on the **/devel/** mount point on the IdM client as a read-write directory.

Prerequisites

- You are logged in as an IdM administrator on any IdM-enrolled host.

Procedure

1. Create the **raleigh** automount location:

```
$ ipa automountlocation-add raleigh
---------------------------------
Added automount location "raleigh"
---------------------------------
  Location: raleigh
```

2. Create an **auto.devel** automount map in the **raleigh** location:

```
$ ipa automountmap-add raleigh auto.devel
-------------------------------
Added automount map "auto.devel"
-------------------------------
  Map: auto.devel
```

3. Add the keys and mount information for the **exports/** share:

   a. Add the key and mount information for the **auto.devel** map:

```
$ ipa automountkey-add raleigh auto.devel --key='*' --info='-sec=krb5p,vers=4 nfs-server.idm.example.com:/exports/&'
-----------------------
Added automount key "*"
-----------------------
  Key: *
  Mount information: -sec=krb5p,vers=4 nfs-server.idm.example.com:/exports/&
```

   b. Add the key and mount information for the **auto.master** map:

```
$ ipa automountkey-add raleigh auto.master --key=/devel --info=auto.devel
---------------------------
Added automount key "/devel"
---------------------------
  Key: /devel
  Mount information: auto.devel
```

4.5. CONFIGURING AUTOMOUNT ON AN IDM CLIENT

As an Identity Management (IdM) system administrator, you can configure automount services on an IdM client so that NFS shares configured for a location to which the client has been added are accessible to an IdM user automatically when the user logs in to the client. The example describes how to configure an IdM client to use automount services that are available in the **raleigh** location.

Prerequisites

- You have **root** access to the IdM client.
- You are logged in as IdM administrator.
- The automount location exists. The example location is **raleigh**.

Procedure

1. On the IdM client, enter the **ipa-client-automount** command and specify the location. Use the **-U** option to run the script unattended:

```
# ipa-client-automount --location raleigh -U
```

2. Stop the autofs service, clear the SSSD cache, and start the autofs service to load the new configuration settings:

```
# systemctl stop autofs ; sss_cache -E ; systemctl start autofs
```

4.6. VERIFYING THAT AN IDM USER CAN ACCESS NFS SHARES ON AN IDM CLIENT

As an Identity Management (IdM) system administrator, you can test if an IdM user that is a member of a specific group can access NFS shares when logged in to a specific IdM client.

In the example, the following scenario is tested:

- An IdM user named **idm_user** belonging to the **developers** group can read and write the contents of the files in the **/devel/project** directory automounted on **idm-client.idm.example.com**, an IdM client located in the **raleigh** automount location.

Prerequisites

- You have configured an IdM keytab for an NFS server and exported an NFS share .
- You have configured automount locations, maps, and mount points in IdM in which you configured how IdM users can access the NFS share.
- You have configured automount on the IdM client.

Procedure

1. Verify that the IdM user can access the **read-write** directory:
   a. Connect to the IdM client as the IdM user:

```
$ ssh idm_user@idm-client.idm.example.com
Password:
```

   b. Obtain the ticket-granting ticket (TGT) for the IdM user:

```
$ kinit idm_user
```

c. [Optional] View the group membership of the IdM user:

```
$ ipa user-show idm_user
  User login: idm_user
  [...]
  Member of groups: developers, ipausers
```

d. Navigate to the **/devel/project** directory:

```
$ cd /devel/project
```

e. List the directory contents:

```
$ ls
rw_file
```

f. Add a line to the file in the directory to test the **write** permission:

```
$ echo "idm_user can write into the file" > rw_file
```

g. [Optional] View the updated contents of the file:

```
$ cat rw_file
this is a read-write file
idm_user can write into the file
```

The output confirms that **idm_user** can write into the file.

CHAPTER 5. USING ANSIBLE TO AUTOMOUNT NFS SHARES FOR IDM USERS

Automount is a way to manage, organize, and access directories across multiple systems. Automount automatically mounts a directory whenever access to it is requested. This works well within an Identity Management (IdM) domain as it allows you to share directories on clients within the domain easily.

You can use Ansible to configure NFS shares to be mounted automatically for IdM users logged in to IdM clients in an IdM location.

The example in this chapter uses the following scenario:

- **nfs-server.idm.example.com** is the fully-qualified domain name (FQDN) of a Network File System (NFS) server.
- **nfs-server.idm.example.com** is an IdM client located in the **raleigh** automount location.
- The NFS server exports the **/exports/project** directory as read-write.
- Any IdM user belonging to the **developers** group can access the contents of the exported directory as **/devel/project/** on any IdM client that is located in the same **raleigh** automount location as the NFS server.
- **idm-client.idm.example.com** is an IdM client located in the **raleigh** automount location.

IMPORTANT

If you want to use a Samba server instead of an NFS server to provide the shares for IdM clients, see the How do I configure kerberized CIFS mounts with Autofs in an IPA environment? KCS solution.

The chapter contains the following sections:

1. Autofs and automount in IdM
2. Configuring an IdM keytab for an NFS server
3. Exporting NFS shares in IdM
4. Preparing your Ansible control node for managing IdM
5. Configuring automount locations, maps, and keys in IdM by using Ansible
6. Using Ansible to add IdM users to a group that owns NFS shares
7. Using Ansible to add an IdM client to an automount location
8. Configuring automount on an IdM client
9. Verifying that an IdM user can access NFS shares on an IdM client

5.1. AUTOFS AND AUTOMOUNT IN IDM

The **autofs** service automates the mounting of directories, as needed, by directing the **automount** daemon to mount directories when they are accessed. In addition, after a period of inactivity, **autofs** directs **automount** to unmount auto-mounted directories. Unlike static mounting, on-demand mounting saves system resources.

Automount maps

On a system that utilizes **autofs**, the **automount** configuration is stored in several different files. The primary **automount** configuration file is **/etc/auto.master**, which contains the master mapping of **automount** mount points, and their associated resources, on a system. This mapping is known as *automount maps*.

The **/etc/auto.master** configuration file contains the *master map*. It can contain references to other maps. These maps can either be direct or indirect. Direct maps use absolute path names for their mount points, while indirect maps use relative path names.

Automount configuration in IdM

While **automount** typically retrieves its map data from the local **/etc/auto.master** and associated files, it can also retrieve map data from other sources. One common source is an LDAP server. In the context of Identity Management (IdM), this is a 389 Directory Server.

If a system that uses **autofs** is a client in an IdM domain, the **automount** configuration is not stored in local configuration files. Instead, the **autofs** configuration, such as maps, locations, and keys, is stored as LDAP entries in the IdM directory. For example, for the **idm.example.com** IdM domain, the default *master map* is stored as follows:

```
dn:
automountmapname=auto.master,cn=default,cn=automount,dc=idm,dc=example,dc=com
objectClass: automountMap
objectClass: top
automountMapName: auto.master
```

Additional resources

- Mounting file systems on demand

5.2. CONFIGURING AN IDM KEYTAB FOR AN NFS SERVER

Configure a Kerberos-aware NFS server so that users logged in to other Identity Management (IdM) clients can access directories and files on this NFS server.

The example describes how to configure NFS service running on **nfs-server.idm.example.com**.

Prerequisites

- You are logged in as an IdM administrator.
- You have **root** access to the NFS server.
- You have installed the required packages to export NFS shares .
- [Optional] You have configured the NFS server to run behind a firewall .

Procedure

1. On any IdM-enrolled host, add the NFS service to IdM:

```
$ ipa service-add nfs/nfs-server.idm.example.com
--------------------------------------------------------------
Added service "nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM"
--------------------------------------------------------------
  Principal name: nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM
  Principal alias: nfs/nfs-server.idm.example.com@IDM.EXAMPLE.COM
  Managed by: nfs-server.idm.example.com
```

2. On the NFS server, obtain the keytab for the NFS service:

```
# ipa-getkeytab -p nfs/nfs-server.idm.example.com -k /etc/krb5.keytab
Keytab successfully retrieved and stored in: /etc/krb5.keytab
```

3. On the NFS server, restart the NFS service:

```
# systemctl restart nfs-server
```

4. On the NFS server, enable the NFS service:

```
# systemctl enable nfs-server
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service →
/usr/lib/systemd/system/nfs-server.service.
```

5.3. EXPORTING NFS SHARES IN IDM

As an Identity Management (IdM) system administrator, you can use an NFS server to share a directory with IdM users over the network.

Prerequisites

- You are logged in as an IdM administrator.
- You have **root** access to the NFS server.
- You have installed the required packages to export NFS shares .
- [Optional] You have configured the NFS server to run behind a firewall .

Procedure

1. Create the directory you want to export:

```
# mkdir -p /exports/project
```

2. Give the owner and group the rights to read, write and execute the directory:

```
# chmod 770 /exports/project
```

3. Add the **GSID** sticky bit so that any files created in the directory will have their group ownership set to that of the directory owner:

```
# chmod g+s /exports_ro/documentation /exports/project
```

4. Create a file in the directory with some content:

```
# echo "this is a read-write file" > /exports/project/rw_file
```

5. To a file in the **/etc/exports.d/** directory, add the following information:

   - Which directory you want to export
   - How you want users to authenticate to access the files in the directory
   - What permissions you want users to have on the files in the directory

```
# echo "/exports/project *(sec=krb5p,rw)" > /etc/exports.d/project.exports
```

sec=krb5 uses the Kerberos V5 protocol instead of local UNIX UIDs and GIDs to authenticate users.

Alternatively, use **sec=krb5i** or **sec=krb5p**:

sec=krb5i

uses Kerberos V5 for user authentication and performs integrity checking of NFS operations using secure checksums to prevent data tampering.

sec=krb5p

uses Kerberos V5 for user authentication, integrity checking, and encrypts NFS traffic to prevent traffic sniffing. This is the most secure setting, but it also involves the most performance overhead.

6. Reexport all directories, synchronizing the master export table kept in **/var/lib/nfs/etab** with **/etc/exports** and files under **/etc/exports.d**:

```
# exportfs -r
```

7. Display the current export list suitable for **/etc/exports**:

```
# exportfs -s
/exports/project  *
(sync,wdelay,hide,no_subtree_check,sec=krb5p,rw,secure,root_squash,no_all_squash)
```

Additional resources

- For information on **krb5** methods, see the **nfs** man page.

5.4. PREPARING YOUR ANSIBLE CONTROL NODE FOR MANAGING IDM

As a system administrator managing Identity Management (IdM), when working with Red Hat Ansible Engine, it is good practice to do the following:

- Create a subdirectory dedicated to Ansible playbooks in your home directory, for example **~/MyPlaybooks**.

- Copy and adapt sample Ansible playbooks from the **/usr/share/doc/ansible-freeipa/*** and **/usr/share/doc/rhel-system-roles/*** directories and subdirectories into your **~/MyPlaybooks** directory.
- Include your inventory file in your **~/MyPlaybooks** directory.

By following this practice, you can find all your playbooks in one place and you can run your playbooks without invoking root privileges.

NOTE

You only need **root** privileges on the managed nodes to execute the **ipaserver**, **ipareplica**, **ipaclient** and **ipabackup ansible-freeipa** roles. These roles require privileged access to directories and the **dnf** software package manager.

This section describes how to create the **~/MyPlaybooks** directory and configure it so that you can use it to store and run Ansible playbooks.

Prerequisites

- You have installed an IdM server on your managed nodes, ***server.idm.example.com*** and ***replica.idm.example.com***.
- You have configured DNS and networking so you can log in to the managed nodes, ***server.idm.example.com*** and ***replica.idm.example.com***, directly from the control node.
- You know the IdM **admin** password.

Procedure

1. Create a directory for your Ansible configuration and playbooks in your home directory:

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

2. Change into the **~/MyPlaybooks/** directory:

```
$ cd ~/MyPlaybooks
```

3. Create the **~/MyPlaybooks/ansible.cfg** file with the following content:

```
[defaults]
inventory = /home/your_username/MyPlaybooks/inventory

[privilege_escalation]
become=True
```

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

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

[us]
replica.idm.example.com
```

```
[ipaserver:children]
eu
us
```

This configuration defines two host groups, **eu** and **us**, for hosts in these locations. Additionally, this configuration defines the **ipaserver** host group, which contains all hosts from the **eu** and **us** groups.

5. [Optional] Create an SSH public and private key. To simplify access in your test environment, do not set a password on the private key:

```
$ ssh-keygen
```

6. Copy the SSH public key to the IdM **admin** account on each managed node:

```
$ ssh-copy-id admin@server.idm.example.com
$ ssh-copy-id admin@replica.idm.example.com
```

You must enter the IdM **admin** password when you enter these commands.

Additional resources

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

5.5. CONFIGURING AUTOMOUNT LOCATIONS, MAPS, AND KEYS IN IDM BY USING ANSIBLE

As an Identity Management (IdM) system administrator, you can configure automount locations and maps in IdM so that IdM users in the specified locations can access shares exported by an NFS server by navigating to specific mount points on their hosts. Both the exported NFS server directory and the mount points are specified in the maps. In LDAP terms, a location is a container for such map entries.

The example describes how to use Ansible to configure the **raleigh** location and a map that mounts the **nfs-server.idm.example.com:/exports/project** share on the **/devel/project** mount point on the IdM client as a read-write directory.

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 an IdM server.

Procedure

1. On your Ansible control node, navigate to your ***~/MyPlaybooks/*** directory:

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

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

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/automount/automount-location-present.yml automount-location-map-and-key-present.yml
```

3. Open the **automount-location-map-and-key-present.yml** file for editing.

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

   - Set the **ipaadmin_password** variable to the password of the IdM **admin**.
   - Set the **name** variable to **raleigh**.
   - Ensure that the **state** variable is set to **present**.
     This is the modified Ansible playbook file for the current example:

```
---
- name: Automount location present example
  hosts: ipaserver
  become: true
  tasks:
  - name: Ensure automount location is present
    ipaautomountlocation:
      ipaadmin_password: Secret123
      name: raleigh
      state: present
```

5. Continue editing the **automount-location-map-and-key-present.yml** file:

   a. In the **tasks** section, add a task to ensure the presence of an automount map:

```
[...]
  tasks:
[...]
  - name: ensure map named auto.devel in location raleigh is created
    ipaautomountmap:
      ipaadmin_password: Secret123
      name: auto.devel
      location: raleigh
      state: present
```

   b. In the **tasks** section, add a task to add the mount point and NFS server information to the map:

```
[...]
  tasks:
[...]
  - name: ensure automount key /devel/project is present
    ipaautomountkey:
      ipaadmin_password: Secret123
      location: raleigh
      mapname: auto.devel
```

```
    key: /devel/project
    info: nfs-server.idm.example.com:/exports/project
    state: present
```

6. Save the file.
7. Run the Ansible playbook and specify the playbook and inventory files:

```
$ ansible-playbook -v -i inventory automount-location-map-and-key-present.yml
```

5.6. USING ANSIBLE TO ADD IDM USERS TO A GROUP THAT OWNS NFS SHARES

As an Identity Management (IdM) system administrator, you can use Ansible to create a group of users that is able to access NFS shares, and add IdM users to this group.

This example describes how to use an Ansible playbook to ensure that the **idm_user** account belongs to the **developers** group, so that **idm_user** can access the **/exports/project** NFS share.

Prerequisites

- You have **root** access to the **nfs-server.idm.example.com** NFS server, which is an IdM client located in the **raleigh** automount location.
- 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 an IdM server.
  - In ***~/MyPlaybooks/***, you have created the **automount-location-map-and-key-present.yml** file that already contains tasks from Configuring automount locations, maps, and keys in IdM by using Ansible.

Procedure

1. On your Ansible control node, navigate to the ***~/MyPlaybooks/*** directory:

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

2. Open the **automount-location-map-and-key-present.yml** file for editing.
3. In the **tasks** section, add a task to ensure that the IdM **developers** group exists and **idm_user** is added to this group:

```
[...]
  tasks:
[...]
  - ipagroup:
```

```
    ipaadmin_password: Secret123
    name: developers
    user:
    - idm_user
    state: present
```

4. Save the file.

5. Run the Ansible playbook and specify the playbook and inventory files:

```
$ ansible-playbook -v -i inventory automount-location-map-and-key-present.yml
```

6. On the NFS server, change the group ownership of the **/exports/project** directory to **developers** so that every IdM user in the group can access the directory:

```
# chgrp developers /exports/project
```

5.7. USING ANSIBLE TO ADD AN IDM CLIENT TO AN AUTOMOUNT LOCATION

As an Identity Management (IdM) system administrator, you can use Ansible to add an IdM client to an automount location so that the NFS shares configured for the location are accessible to an IdM user provided the IdM user is logged in to this IdM client.

Prerequisites

- The host exists in IdM. The example host is **idm-client.idm.example.com**.
- The automount location exists. The example location is **raleigh**.
- 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 an IdM server.
  - In ***~/MyPlaybooks/***, you have created the **automount-location-map-and-key-present.yml** file that already contains tasks from Configuring automount locations, maps, and keys in IdM by using Ansible.

Procedure

1. On your Ansible control node, navigate to your ***~/MyPlaybooks/*** directory:

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

2. Open the **automount-location-map-and-key-present.yml** file for editing.

3. In the **tasks** section, add a task to ensure that the **idm-client.idm.example.com** IdM host is added to the **raleigh** automount location:

```
[...]
  tasks:
[...]
  - ipahost:
      ipaadmin_password: Secret123
      name: idm-client.idm.example.com
      location: raleigh
      state: present
```

4. Save the file.

5. Run the Ansible playbook and specify the playbook and inventory files:

```
$ ansible-playbook -v -i inventory automount-location-map-and-key-present.yml
```

5.8. CONFIGURING AUTOMOUNT ON AN IDM CLIENT

As an Identity Management (IdM) system administrator, you can configure automount services on an IdM client so that NFS shares configured for a location to which the client has been added are accessible to an IdM user automatically when the user logs in to the client. The example describes how to configure an IdM client to use automount services that are available in the **raleigh** location.

Prerequisites

- You have **root** access to the IdM client.
- You are logged in as IdM administrator.
- The automount location exists. The example location is **raleigh**.

Procedure

1. On the IdM client, enter the **ipa-client-automount** command and specify the location. Use the **-U** option to run the script unattended:

```
# ipa-client-automount --location raleigh -U
```

2. Stop the autofs service, clear the SSSD cache, and start the autofs service to load the new configuration settings:

```
# systemctl stop autofs ; sss_cache -E ; systemctl start autofs
```

5.9. VERIFYING THAT AN IDM USER CAN ACCESS NFS SHARES ON AN IDM CLIENT

As an Identity Management (IdM) system administrator, you can test if an IdM user that is a member of a specific group can access NFS shares when logged in to a specific IdM client.

In the example, the following scenario is tested:

- An IdM user named **idm_user** belonging to the **developers** group can read and write the contents of the files in the **/devel/project** directory automounted on **idm-client.idm.example.com**, an IdM client located in the **raleigh** automount location.

Prerequisites

- You have configured an IdM keytab for an NFS server and exported an NFS share.
- You have configured automount locations, maps, and mount points in IdM in which you configured how IdM users can access the NFS share.
- You have used Ansible to add IdM users to the developers group that owns the NFS shares.
- You have used Ansible to add the IdM client to the relevant automount location.
- You have configured automount on the IdM client.

Procedure

1. Verify that the IdM user can access the **read-write** directory:
   a. Connect to the IdM client as the IdM user:

```
$ ssh idm_user@idm-client.idm.example.com
Password:
```

   b. Obtain the ticket-granting ticket (TGT) for the IdM user:

```
$ kinit idm_user
```

   c. [Optional] View the group membership of the IdM user:

```
$ ipa user-show idm_user
  User login: idm_user
  [...]
  Member of groups: developers, ipausers
```

   d. Navigate to the **/devel/project** directory:

```
$ cd /devel/project
```

   e. List the directory contents:

```
$ ls
rw_file
```

   f. Add a line to the file in the directory to test the **write** permission:

```
$ echo "idm_user can write into the file" > rw_file
```

   g. [Optional] View the updated contents of the file:

```
$ cat rw_file
this is a read-write file
idm_user can write into the file
```

The output confirms that **idm_user** can write into the file.