



Red Hat Enterprise Linux 9

Using external Red Hat utilities with Identity Management

Integrating services and Red Hat products in IdM

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

Integrating services and Red Hat products in IdM

Legal Notice

Copyright © 2025 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

Administrators can integrate services and Red Hat products in a Red Hat Identity Management (IdM) domain. This includes services, such as Samba, Ansible, and automount, and also products, such as OpenShift Container Platform, OpenStack, and Satellite. IdM users can then access these services and products.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	3
CHAPTER 1. IDM INTEGRATION WITH RED HAT PRODUCTS	4
CHAPTER 2. USING EXTERNAL IDENTITY PROVIDERS TO AUTHENTICATE TO IDM	5
2.1. THE BENEFITS OF CONNECTING IDM TO AN EXTERNAL IDP	5
2.2. HOW IDM INCORPORATES LOGINS VIA EXTERNAL IDPS	5
2.3. CREATING A REFERENCE TO AN EXTERNAL IDENTITY PROVIDER	6
2.4. EXAMPLE REFERENCES TO DIFFERENT EXTERNAL IDPS IN IDM	7
2.5. OPTIONS FOR THE IPA IDP-* COMMANDS TO MANAGE EXTERNAL IDENTITY PROVIDERS IN IDM	8
2.6. MANAGING REFERENCES TO EXTERNAL IDPS	9
2.7. ENABLING AN IDM USER TO AUTHENTICATE VIA AN EXTERNAL IDP	10
2.8. RETRIEVING AN IDM TICKET-GRANTING TICKET AS AN EXTERNAL IDP USER	11
2.9. LOGGING IN TO AN IDM CLIENT VIA SSH AS AN EXTERNAL IDP USER	12
2.10. THE --PROVIDER OPTION IN THE IPA IDP-* COMMANDS	13
CHAPTER 3. SETTING UP SAMBA ON AN IDM DOMAIN MEMBER	17
3.1. PREPARING THE IDM DOMAIN FOR INSTALLING SAMBA ON DOMAIN MEMBERS	17
3.2. INSTALLING AND CONFIGURING A SAMBA SERVER ON AN IDM CLIENT	19
3.3. MANUALLY ADDING AN ID MAPPING CONFIGURATION IF IDM TRUSTS A NEW DOMAIN	20
3.4. ADDITIONAL RESOURCES	22
CHAPTER 4. MIGRATING FROM NIS TO IDENTITY MANAGEMENT	23
4.1. ENABLING NIS IN IDM	23
4.2. MIGRATING USER ENTRIES FROM NIS TO IDM	24
4.3. MIGRATING USER GROUP FROM NIS TO IDM	25
4.4. MIGRATING HOST ENTRIES FROM NIS TO IDM	26
4.5. MIGRATING NETGROUP ENTRIES FROM NIS TO IDM	27
4.6. MIGRATING AUTOMOUNT MAPS FROM NIS TO IDM	28
CHAPTER 5. USING AUTOMOUNT IN IDM	30
5.1. AUTOFS AND AUTOMOUNT IN IDM	30
5.2. SETTING UP AN NFS SERVER WITH KERBEROS IN A RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT DOMAIN	31
5.3. CONFIGURING AUTOMOUNT LOCATIONS AND MAPS IN IDM USING THE IDM CLI	32
5.4. CONFIGURING AUTOMOUNT ON AN IDM CLIENT	33
5.5. VERIFYING THAT AN IDM USER CAN ACCESS NFS SHARES ON AN IDM CLIENT	34
CHAPTER 6. USING ANSIBLE TO AUTOMOUNT NFS SHARES FOR IDM USERS	36
6.1. AUTOFS AND AUTOMOUNT IN IDM	36
6.2. SETTING UP AN NFS SERVER WITH KERBEROS IN A RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT DOMAIN	37
6.3. CONFIGURING AUTOMOUNT LOCATIONS, MAPS, AND KEYS IN IDM BY USING ANSIBLE	38
6.4. USING ANSIBLE TO ADD IDM USERS TO A GROUP THAT OWNS NFS SHARES	40
6.5. CONFIGURING AUTOMOUNT ON AN IDM CLIENT	42
6.6. VERIFYING THAT AN IDM USER CAN ACCESS NFS SHARES ON AN IDM CLIENT	42

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

Submitting feedback through Jira (account required)

1. Log in to the [Jira](#) website.
2. Click **Create** in the top navigation bar
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 **Create** at the bottom of the dialogue.

CHAPTER 1. IDM INTEGRATION WITH RED HAT PRODUCTS

Find 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](#)
- [OpenShift Container Platform](#)
- [Red Hat OpenStack Platform](#)
- [Red Hat Satellite](#)
- [Red Hat Single Sign-On](#)
- [Red Hat Virtualization](#)

CHAPTER 2. USING EXTERNAL IDENTITY PROVIDERS TO AUTHENTICATE TO IDM

You can associate users with external identity providers (IdP) that support the OAuth 2.0 device authorization flow. When these users authenticate with the System Security Services Daemon (SSSD) version available in RHEL 9.1 or later, they receive RHEL Identity Management (IdM) single sign-on capabilities with Kerberos tickets after performing authentication and authorization at the external IdP.

Notable features include:

- Adding, modifying, and deleting references to external IdPs with **ipa idp-*** commands.
- Enabling IdP authentication for users with the **ipa user-mod --user-auth-type=idp** command.

2.1. THE BENEFITS OF CONNECTING IDM TO AN EXTERNAL IDP

As an administrator, you might want to allow users stored in an external identity source, such as a cloud services provider, to access RHEL systems joined to your Identity Management (IdM) environment. To achieve this, you can delegate the authentication and authorization process of issuing Kerberos tickets for these users to that external entity.

You can use this feature to expand IdM's capabilities and allow users stored in external identity providers (IdPs) to access Linux systems managed by IdM.

2.2. HOW IDM INCORPORATES LOGINS VIA EXTERNAL IDPS

SSSD 2.7.0 contains the **sssd-idp** package, which implements the **idp** Kerberos pre-authentication method. This authentication method follows the OAuth 2.0 Device Authorization Grant flow to delegate authorization decisions to external IdPs:

1. An IdM client user initiates OAuth 2.0 Device Authorization Grant flow, for example, by attempting to retrieve a Kerberos Ticket Granting Ticket (TGT) with the **kinit** utility at the command line.
2. A special code and website link are sent from the Authorization Server to the IdM Key Distribution Center (KDC) backend.
3. The IdM client displays the link and the code to the user. In this example, the IdM client outputs the link and code on the command line.
4. The user opens the website link in a browser, which can be on another host, a mobile phone, and so on:
 - a. The user enters the special code.
 - b. If necessary, the user logs in to the OAuth 2.0-based IdP.
 - c. The user is prompted to authorize the client to access information.
5. The user confirms access at the original device prompt. In this example, the user hits the **Enter** key at the command line.
6. The IdM KDC backend polls the OAuth 2.0 Authorization Server for access to user information.

What is supported:

- Logging in remotely via Secure Shell (SSH) with the **keyboard-interactive** authentication method enabled, which allows calling Pluggable Authentication Module (PAM) libraries.
- Logging in locally with the console via the **logind** service.
- Retrieving a Kerberos TGT with the **kinit** utility.

What is currently not supported:

- Logging in to the IdM WebUI directly. To log in to the IdM WebUI, you must first acquire a Kerberos ticket.
- Logging in to Cockpit WebUI directly. To log in to the Cockpit WebUI, you must first acquire a Kerberos ticket.

Additional resources

- [Authentication against external Identity Providers](#)
- [RFC 8628: OAuth 2.0 Device Authorization Grant](#)

2.3. CREATING A REFERENCE TO AN EXTERNAL IDENTITY PROVIDER

To connect external identity providers (IdPs) to your Identity Management (IdM) environment, create IdP references in IdM. Complete this procedure to create a reference called **my-keycloak-idp** to an IdP based on the Keycloak template. For more reference templates, see [Example references to different external IdPs in IdM](#).

Prerequisites

- You have registered IdM as an OAuth application to your external IdP, and obtained a client ID.
- You can authenticate as the IdM admin account.
- Your IdM servers are using RHEL 9.1 or later.
- Your IdM servers are using SSSD 2.7.0 or later.

Procedure

1. Authenticate as the IdM admin on an IdM server.

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

2. Create a reference called **my-keycloak-idp** to an IdP based on the Keycloak template, where the **--base-url** option specifies the URL to the Keycloak server in the format **server-name.\$DOMAIN:\$PORT/prefix**.

```
[root@server ~]# ipa idp-add my-keycloak-idp \
--provider keycloak --organization main \
--base-url keycloak.idm.example.com:8443/auth \
--client-id id13778
```

```
-----
Added Identity Provider reference "my-keycloak-idp"
-----
```

```
Identity Provider reference name: my-keycloak-idp
Authorization URI:
```

```

https://keycloak.idm.example.com:8443/auth/realms/main/protocol/openid-connect/auth
Device authorization URI:
https://keycloak.idm.example.com:8443/auth/realms/main/protocol/openid-
connect/auth/device
Token URI: https://keycloak.idm.example.com:8443/auth/realms/main/protocol/openid-
connect/token
User info URI: https://keycloak.idm.example.com:8443/auth/realms/main/protocol/openid-
connect/userinfo
Client identifier: ipa_oidc_client
Scope: openid email
External IdP user identifier attribute: email

```

Verification

- Verify that the output of the **ipa idp-show** command shows the IdP reference you have created.

```
[root@server ~]# ipa idp-show my-keycloak-idp
```


Additional resources

- [Example references to different external IdPs in IdM](#)
- [Options for the ipa idp-* commands to manage external identity providers in IdM](#)
- [The --provider option in the ipa idp-* commands](#)
- **ipa help idp-add**

2.4. EXAMPLE REFERENCES TO DIFFERENT EXTERNAL IDPS IN IDM

The following table lists examples of the **ipa idp-add** command for creating references to different IdPs in IdM.

Identity Provider	Important options	Command example
Microsoft Identity Platform, Azure AD	--provider microsoft --organization	<pre># ipa idp-add my-azure-idp \ --provider microsoft \ --organization main \ --client-id <azure_client_id></pre>
Google	--provider google	<pre># ipa idp-add my-google-idp \ --provider google \ --client-id <google_client_id></pre>
GitHub	--provider github	<pre># ipa idp-add my-github-idp \ --provider github \ --client-id <github_client_id></pre>

Identity Provider	Important options	Command example
Keycloak, Red Hat Single Sign-On	--provider keycloak --organization --base-url	<pre># ipa idp-add my-keycloak-idp \ --provider keycloak \ --organization main \ --base-url keycloak.idm.example.com:8443/auth \ -- client-id <keycloak_client_id></pre> <div>  <p>NOTE</p> <p>The Quarkus version of Keycloak 17 and later have removed the /auth/ portion of the URI. If you use the non-Quarkus distribution of Keycloak in your deployment, include /auth/ in the --base-url option.</p> </div>
Okta	--provider okta	<pre># ipa idp-add my-okta-idp \ --provider okta --base-url dev-12345.okta.com \ --client-id <okta_client_id></pre>

Additional resources

- [Creating a reference to an external identity provider](#)
- [Options for the ipa idp-* commands to manage external identity providers in IdM](#)
- [The --provider option in the ipa idp-* commands](#)
- [Configure IdM to use Google as external IdP](#) (Red Hat Knowledgebase)
- [Configure IdM to use Entra ID \(Azure AD\) as external IdP](#) (Red Hat Knowledgebase)
- [Configure IdM to use GitHub as external IdP](#) (Red Hat Knowledgebase)

2.5. OPTIONS FOR THE IPA IDP-* COMMANDS TO MANAGE EXTERNAL IDENTITY PROVIDERS IN IDM

The following examples show how to configure references to external IdPs based on the different IdP templates. Use the following options to specify your settings:

--provider

The predefined template for one of the known identity providers.

--client-id

The OAuth 2.0 client identifier issued by the IdP during application registration. As the application registration procedure is specific to each IdP, refer to their documentation for details. If the external IdP is Red Hat Single Sign-On (SSO), see [Creating an OpenID Connect Client](#).

--base-url

Base URL for IdP templates, required by Keycloak and Okta.

--organization

Domain or Organization ID from the IdP, required by Microsoft Azure.

--secret

Optional: Use this option if you have configured your external IdP to require a secret from confidential OAuth 2.0 clients. If you use this option when creating an IdP reference, you are prompted for the secret interactively. Protect the client secret as a password.

**NOTE**

SSSD in RHEL 9.1 only supports non-confidential OAuth 2.0 clients that do not use a client secret. If you want to use external IdPs that require a client secret from confidential clients, you must use SSSD in RHEL 9.2 and later.

Additional resources

- [Creating a reference to an external identity provider](#)
- [Example references to different external IdPs in IdM](#)
- [The --provider option in the ipa idp-* commands](#)

2.6. MANAGING REFERENCES TO EXTERNAL IDPS

After you have created a reference to an external identity provider (IdP), you can find, show, modify, and delete that reference. This example shows you how to manage a reference to an external IdP named **keycloak-server1**.

Prerequisites

- You can authenticate as the IdM admin account.
- Your IdM servers are using RHEL 9.1 or later.
- Your IdM servers are using SSSD 2.7.0 or later.
- You have created a reference to an external IdP in IdM. See [Creating a reference to an external identity provider](#).

Procedure

1. Authenticate as the IdM admin on an IdM server.

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

2. Manage the IdP reference.

- To find an IdP reference whose entry includes the string **keycloak**:

```
[root@server ~]# ipa idp-find keycloak
```

- To display an IdP reference named **my-keycloak-idp**:

```
[root@server ~]# ipa idp-show my-keycloak-idp
```

- To modify an IdP reference, use the **ipa idp-mod** command. For example, to change the secret for an IdP reference named **my-keycloak-idp**, specify the **--secret** option to be prompted for the secret:

```
[root@server ~]# ipa idp-mod my-keycloak-idp --secret
```

- To delete an IdP reference named **my-keycloak-idp**:

```
[root@server ~]# ipa idp-del my-keycloak-idp
```

2.7. ENABLING AN IDM USER TO AUTHENTICATE VIA AN EXTERNAL IDP

To enable an IdM user to authenticate via an external identity provider (IdP), associate the external IdP reference you have previously created with the user account. This example associates the external IdP reference **keycloak-server1** with the user **idm-user-with-external-idp**.

Prerequisites

- Your IdM client and IdM servers are using RHEL 9.1 or later.
- Your IdM client and IdM servers are using SSSD 2.7.0 or later.
- You have created a reference to an external IdP in IdM. See [Creating a reference to an external identity provider](#).

Procedure

- Modify the IdM user entry to associate an IdP reference with the user account:

```
[root@server ~]# ipa user-mod idm-user-with-external-idp \
    --idp my-keycloak-idp \
    --idp-user-id idm-user-with-external-idp@idm.example.com \
    --user-auth-type=idp
-----
Modified user "idm-user-with-external-idp"
-----
User login: idm-user-with-external-idp
First name: Test
Last name: User1
Home directory: /home/idm-user-with-external-idp
Login shell: /bin/sh
Principal name: idm-user-with-external-idp@idm.example.com
Principal alias: idm-user-with-external-idp@idm.example.com
Email address: idm-user-with-external-idp@idm.example.com
UID: 35000003
GID: 35000003
User authentication types: idp
External IdP configuration: keycloak
```

External IdP user identifier: `idm-user-with-external-idp@idm.example.com`

Account disabled: False

Password: False

Member of groups: `ipausers`

Kerberos keys available: False

Verification

- Verify that the output of the **`ipa user-show`** command for that user displays references to the IdP:

```
[root@server ~]# ipa user-show idm-user-with-external-idp
User login: idm-user-with-external-idp
First name: Test
Last name: User1
Home directory: /home/idm-user-with-external-idp
Login shell: /bin/sh
Principal name: idm-user-with-external-idp@idm.example.com
Principal alias: idm-user-with-external-idp@idm.example.com
Email address: idm-user-with-external-idp@idm.example.com
ID: 35000003
GID: 35000003
User authentication types: idp
External IdP configuration: keycloak
External IdP user identifier: idm-user-with-external-idp@idm.example.com
Account disabled: False
Password: False
Member of groups: ipausers
Kerberos keys available: False
```

2.8. RETRIEVING AN IDM TICKET-GRANTING TICKET AS AN EXTERNAL IDP USER

If you have delegated authentication for an Identity Management (IdM) user to an external identity provider (IdP), the IdM user can request a Kerberos ticket-granting ticket (TGT) by authenticating to the external IdP.

Complete this procedure to:

1. Retrieve and store an anonymous Kerberos ticket locally.
2. Request the TGT for the **`idm-user-with-external-idp`** user by using **`kinit`** with the **`-T`** option to enable Flexible Authentication via Secure Tunneling (FAST) channel to provide a secure connection between the Kerberos client and Kerberos Distribution Center (KDC).

Prerequisites

- Your IdM client and IdM servers use RHEL 9.1 or later.
- Your IdM client and IdM servers use SSSD 2.7.0 or later.
- You have created a reference to an external IdP in IdM. See [Creating a reference to an external identity provider](#).

- You have associated an external IdP reference with the user account. See [Enabling an IdM user to authenticate via an external IdP](#).
- The user that you are initially logged in as has write permissions on a directory in the local filesystem.

Procedure

1. Use Anonymous PKINIT to obtain a Kerberos ticket and store it in a file named **./fast.ccache**.

```
$ kinit -n -c ./fast.ccache
```

2. Optional: View the retrieved ticket:

```
$ klist -c fast.ccache
Ticket cache: FILE:fast.ccache
Default principal: WELLKNOWN/ANONYMOUS@WELLKNOWN:ANONYMOUS

Valid starting    Expires          Service principal
03/03/2024 13:36:37 03/04/2024 13:14:28
krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
```

3. Begin authenticating as the IdM user, using the **-T** option to enable the FAST communication channel.

```
[root@client ~]# kinit -T ./fast.ccache idm-user-with-external-idp
Authenticate at https://oauth2.idp.com:8443/auth/realms/master/device?user_code=YHMQ-
XKTL and press ENTER.:
```

4. In a browser, authenticate as the user at the website provided in the command output.
5. At the command line, press the **Enter** key to finish the authentication process.

Verification

- Display your Kerberos ticket information and confirm that the line **config: pa_type** shows **152** for pre-authentication with an external IdP.

```
[root@client ~]# klist -C
Ticket cache: KCM:0:58420
Default principal: idm-user-with-external-idp@IDM.EXAMPLE.COM

Valid starting    Expires          Service principal
05/09/22 07:48:23 05/10/22 07:03:07 krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
config: fast_avail(krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM) = yes
08/17/2022 20:22:45 08/18/2022 20:22:43
krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
config: pa_type(krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM) = 152
```

The **pa_type = 152** indicates external IdP authentication.

2.9. LOGGING IN TO AN IDM CLIENT VIA SSH AS AN EXTERNAL IDP USER

To log in to an IdM client via SSH as an external identity provider (IdP) user, begin the login process on the command line. When prompted, perform the authentication process at the website associated with the IdP, and finish the process at the Identity Management (IdM) client.

Prerequisites

- Your IdM client and IdM servers are using RHEL 9.1 or later.
- Your IdM client and IdM servers are using SSSD 2.7.0 or later.
- You have created a reference to an external IdP in IdM. See [Creating a reference to an external identity provider](#).
- You have associated an external IdP reference with the user account. See [Enabling an IdM user to authenticate via an external IdP](#).

Procedure

1. Attempt to log in to the IdM client via SSH.

```
[user@client ~]$ ssh idm-user-with-external-idp@client.idm.example.com
(idm-user-with-external-idp@client.idm.example.com) Authenticate at
https://oauth2.idp.com:8443/auth/realms/main/device?user_code=XYFL-ROYR and press
ENTER.
```

2. In a browser, authenticate as the user at the website provided in the command output.
3. At the command line, press the **Enter** key to finish the authentication process.

Verification

- Display your Kerberos ticket information and confirm that the line **config: pa_type** shows **152** for pre-authentication with an external IdP.

```
[idm-user-with-external-idp@client ~]$ klist -C
Ticket cache: KCM:0:58420
Default principal: idm-user-with-external-idp@IDM.EXAMPLE.COM

Valid starting    Expires          Service principal
05/09/22 07:48:23 05/10/22 07:03:07 krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
config: fast_avail(krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM) = yes
08/17/2022 20:22:45 08/18/2022 20:22:43
krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM
config: pa_type(krbtgt/IDM.EXAMPLE.COM@IDM.EXAMPLE.COM) = 152
```

2.10. THE --PROVIDER OPTION IN THE IPA IDP-* COMMANDS

The following identity providers (IdPs) support OAuth 2.0 device authorization grant flow:

- Microsoft Identity Platform, including Azure AD
- Google
- GitHub

- Keycloak, including Red Hat Single Sign-On (SSO)
- Okta

When using the **ipa idp-add** command to create a reference to one of these external IdPs, you can specify the IdP type with the **--provider** option, which expands into additional options as described below:

--provider=microsoft

Microsoft Azure IdPs allow parametrization based on the Azure tenant ID, which you can specify with the **--organization** option to the **ipa idp-add** command. If you need support for the live.com IdP, specify the option **--organization common**.

Choosing **--provider=microsoft** expands to use the following options. The value of the **--organization** option replaces the string **\${ipaidporg}** in the table.

Option	Value
--auth-uri=URI	https://login.microsoftonline.com/\${ipaidporg}/oauth2/v2.0/authorize
--dev-auth-uri=URI	https://login.microsoftonline.com/\${ipaidporg}/oauth2/v2.0/devicecode
--token-uri=URI	https://login.microsoftonline.com/\${ipaidporg}/oauth2/v2.0/token
--userinfo-uri=URI	https://graph.microsoft.com/oidc/userinfo
--keys-uri=URI	https://login.microsoftonline.com/common/discovery/v2.0/keys
--scope=STR	openid email
--idp-user-id=STR	email

--provider=google

Choosing **--provider=google** expands to use the following options:

Option	Value
--auth-uri=URI	https://accounts.google.com/o/oauth2/auth
--dev-auth-uri=URI	https://oauth2.googleapis.com/device/code
--token-uri=URI	https://oauth2.googleapis.com/token
--userinfo-uri=URI	https://openidconnect.googleapis.com/v1/userinfo
--keys-uri=URI	https://www.googleapis.com/oauth2/v3/certs

Option	Value
--scope=STR	openid email
--idp-user-id=STR	email

--provider=github

Choosing **--provider=github** expands to use the following options:

Option	Value
--auth-uri=URI	https://github.com/login/oauth/authorize
--dev-auth-uri=URI	https://github.com/login/device/code
--token-uri=URI	https://github.com/login/oauth/access_token
--userinfo-uri=URI	https://openidconnect.googleapis.com/v1/userinfo
--keys-uri=URI	https://api.github.com/user
--scope=STR	user
--idp-user-id=STR	login

--provider=keycloak

With Keycloak, you can define multiple realms or organizations. Since it is often a part of a custom deployment, both base URL and realm ID are required, and you can specify them with the **--base-url** and **--organization** options to the **ipa idp-add** command:

```
[root@client ~]# ipa idp-add MySSO --provider keycloak \ --org main --base-url
keycloak.domain.com:8443/auth \ --client-id <your-client-id>
```

Choosing **--provider=keycloak** expands to use the following options. The value you specify in the **--base-url** option replaces the string **\${ipaidpbaseurl}** in the table, and the value you specify for the **--organization** option replaces the string **\${ipaidporg}**.

Option	Value
--auth-uri=URI	https://\${ipaidpbaseurl}/realms/\${ipaidporg}/protocol/openid-connect/auth
--dev-auth-uri=URI	https://\${ipaidpbaseurl}/realms/\${ipaidporg}/protocol/openid-connect/auth/device
--token-uri=URI	https://\${ipaidpbaseurl}/realms/\${ipaidporg}/protocol/openid-connect/token

Option	Value
--userinfo-uri=URI	https://<code>\${ipaidpbaseurl}</code>/realms/<code>\${ipaidporg}</code>/protocol/openid-connect/userinfo
--scope=STR	openid email
--idp-user-id=STR	email

--provider=okta

After registering a new organization in Okta, a new base URL is associated with it. You can specify this base URL with the **--base-url** option to the **ipa idp-add** command:

```
[root@client ~]# ipa idp-add MyOkta --provider okta --base-url dev-12345.okta.com --client-id <your-client-id>
```

Choosing **--provider=okta** expands to use the following options. The value you specify for the **--base-url** option replaces the string **`${ipaidpbaseurl}`** in the table.

Option	Value
--auth-uri=URI	https://<code>\${ipaidpbaseurl}</code>/oauth2/v1/authorize
--dev-auth-uri=URI	https://<code>\${ipaidpbaseurl}</code>/oauth2/v1/device/authorize
--token-uri=URI	https://<code>\${ipaidpbaseurl}</code>/oauth2/v1/token
--userinfo-uri=URI	https://<code>\${ipaidpbaseurl}</code>/oauth2/v1/userinfo
--scope=STR	openid email
--idp-user-id=STR	email

Additional resources

- [Pre-populated IdP templates](#)

CHAPTER 3. SETTING UP SAMBA ON AN IDM DOMAIN MEMBER

You can 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.

If users from AD domains need to access shares and printer services provided by Samba, ensure the AES encryption type is enabled in AD. For more information, see [Enabling the AES encryption type in Active Directory using a GPO](#).

Prerequisites

- The host is joined as a client to the IdM domain.
- Both the IdM servers and the client must run on RHEL 9.0 or later.

3.1. 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 have 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. 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.

- 7. 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
```

- 8. Restart the **ipa** service:

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

- 9. Use the **smbclient** utility to verify that Samba responds to Kerberos authentication from the IdM side:
-

```
[root@ipaserver ~]# smbclient -L ipaserver.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)
  ...
```

3.2. INSTALLING AND CONFIGURING A SAMBA SERVER ON AN IDM CLIENT

You can 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](#).

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
```

- 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
```

- 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](#)

- 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
```

- Enable and start the **smb** and **winbind** services:

```
[root@idm_client]# systemctl enable --now smb winbind
```

Verification

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 on your system

3.3. 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
ipairangesize: 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 **ipairangesize** attributes in the next steps.

3. To calculate the highest usable ID, use the following formula:

```
maximum_range = ipabaseid + ipairangesize - 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

- 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
-----	----	-----
<i>example</i>	Disk	
IPC\$	IPC	IPC Service (Samba 4.15.2)
...		

3.4. ADDITIONAL RESOURCES

- [Installing an Identity Management client](#)

CHAPTER 4. 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.

4.1. 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
```

4.2. 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.
- You have [ID ranges](#) that can store UIDs of importing users.

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 primary NIS 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.

4.3. 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 primary NIS 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
    useropts=$(eval echo --users={ $members })
    ipa group-add-member $groupname $useropts
  fi
  ipa group-show $groupname
done
```



NOTE

Make sure your usernames do not contain any special characters to ensure successful migration of the user group.

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
```

4.4. 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 primary NIS 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 $2}')
    primary=$(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 $zone 2>/dev/null | wc -l) -eq 0 ] ; then
        ipa dnszone-add --name-server=$primary --admin-email=root.$primary
```

```

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 $ptrzone --name-server=$primary --admin-email=root.$primary
fi
# Now create this entry
ipa host-add $hostname --ip-address=$ipaddress
ipa host-show $hostname
done

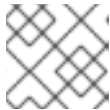
```

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.

4.5. 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 primary NIS 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"
done

```

```

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
```

4.6. 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 primary NIS 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

# $5 is the LDAP server

ldapadd -x -h $5 -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 5. 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 Red Hat Knowledgebase solution [How do I configure kerberized CIFS mounts with Autofs in an IPA environment?](#).

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. SETTING UP AN NFS SERVER WITH KERBEROS IN A RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT DOMAIN

If you use Red Hat Enterprise Linux Identity Management (IdM), you can join your NFS server to the IdM domain. This enables you to centrally manage users and groups and to use Kerberos for authentication, integrity protection, and traffic encryption.

Prerequisites

- The NFS server is [enrolled](#) in a Red Hat Enterprise Linux Identity Management (IdM) domain.
- The NFS server is running and configured.

Procedure

1. Obtain a kerberos ticket as an IdM administrator:

```
# kinit admin
```

2. Create a **nfs/<FQDN>** service principal:

```
# ipa service-add nfs/nfs_server.idm.example.com
```

3. Retrieve the **nfs** service principal from IdM, and store it in the **/etc/krb5.keytab** file:

```
# ipa-getkeytab -s idm_server.idm.example.com -p nfs/nfs_server.idm.example.com -k /etc/krb5.keytab
```

4. Optional: Display the principals in the **/etc/krb5.keytab** file:

```
# klist -k /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
-----
```

```
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
```

By default, the IdM client adds the host principal to the **/etc/krb5.keytab** file when you join the host to the IdM domain. If the host principal is missing, use the **ipa-getkeytab -s *idm_server.idm.example.com* -p *host/nfs_server.idm.example.com* -k /etc/krb5.keytab** command to add it.

5. Use the **ipa-client-automount** utility to configure mapping of IdM IDs:

```
# 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
```

6. Update your **/etc/exports** file, and add the Kerberos security method to the client options. For example:

```
/nfs/projects/ 192.0.2.0/24(rw,sec=krb5i)
```

If you want that your clients can select from multiple security methods, specify them separated by colons:

```
/nfs/projects/ 192.0.2.0/24(rw,sec=krb5:krb5i:krb5p)
```

7. Reload the exported file systems:

```
# exportfs -r
```

5.3. 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
```

5.4. 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
```

5.5. 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 [set up an NFS server with Kerberos on an IdM host](#) .
- 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.

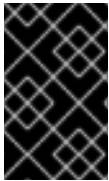
CHAPTER 6. USING ANSIBLE TO AUTOMOUNT NFS SHARES FOR IDM USERS

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 Red Hat Knowledgebase solution [How do I configure kerberized CIFS mounts with Autofs in an IPA environment?](#).

6.1. 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](#)

6.2. SETTING UP AN NFS SERVER WITH KERBEROS IN A RED HAT ENTERPRISE LINUX IDENTITY MANAGEMENT DOMAIN

If you use Red Hat Enterprise Linux Identity Management (IdM), you can join your NFS server to the IdM domain. This enables you to centrally manage users and groups and to use Kerberos for authentication, integrity protection, and traffic encryption.

Prerequisites

- The NFS server is [enrolled](#) in a Red Hat Enterprise Linux Identity Management (IdM) domain.
- The NFS server is running and configured.

Procedure

1. Obtain a kerberos ticket as an IdM administrator:

```
# kinit admin
```

2. Create a **nfs/<FQDN>** service principal:

```
# ipa service-add nfs/nfs_server.idm.example.com
```

3. Retrieve the **nfs** service principal from IdM, and store it in the **/etc/krb5.keytab** file:

```
# ipa-getkeytab -s idm_server.idm.example.com -p nfs/nfs_server.idm.example.com -k /etc/krb5.keytab
```

4. Optional: Display the principals in the **/etc/krb5.keytab** file:

```
# klist -k /etc/krb5.keytab
Keytab name: FILE:/etc/krb5.keytab
KVNO Principal
-----
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
1 nfs/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
```

```
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
7 host/nfs_server.idm.example.com@IDM.EXAMPLE.COM
```

By default, the IdM client adds the host principal to the **/etc/krb5.keytab** file when you join the host to the IdM domain. If the host principal is missing, use the **ipa-getkeytab -s idm_server.idm.example.com -p host/nfs_server.idm.example.com -k /etc/krb5.keytab** command to add it.

5. Use the **ipa-client-automount** utility to configure mapping of IdM IDs:

```
# 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/ldap.conf
Restarting sssd, waiting for it to become available.
Started autofs
```

6. Update your **/etc/exports** file, and add the Kerberos security method to the client options. For example:

```
/nfs/projects/ 192.0.2.0/24(rw,sec=krb5i)
```

If you want that your clients can select from multiple security methods, specify them separated by colons:

```
/nfs/projects/ 192.0.2.0/24(rw,sec=krb5:krb5i:krb5p)
```

7. Reload the exported file systems:

```
# exportfs -r
```

6.3. 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

- On the control node:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package.

- The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
- The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- The target node, that is the node on which the `ansible-freeipa` module is executed, is part of the IdM domain as an IdM client, server or replica.

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
  vars_files:
    - /home/user_name/MyPlaybooks/secret.yml
  tasks:
    - name: Ensure automount location is present
      ipaautomountlocation:
        ipaadmin_password: "{{ ipaadmin_password }}"
        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:

```
[...]
vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
tasks:
[...]
```

```
- name: ensure map named auto.devel in location raleigh is created
  ipaautomountmap:
    ipaadmin_password: "{{ ipaadmin_password }}"
```

```

name: auto.devel
location: raleigh
state: present

```

- b. Add another task to add the mount point and NFS server information to the map:

```

[...]  

vars_files:  

- /home/user_name/MyPlaybooks/secret.yml  

tasks:  

[...]  

- name: ensure automount key /devel/project is present  

  ipaautomountkey:  

    ipaadmin_password: "{{ ipaadmin_password }}"
    location: raleigh
    mapname: auto.devel
    key: /devel/project
    info: nfs-server.idm.example.com:/exports/project
    state: present

```

- c. Add another task to ensure **auto.devel** is connected to **auto.master**:

```

[...]  

vars_files:  

- /home/user_name/MyPlaybooks/secret.yml  

tasks:  

[...]  

- name: Ensure auto.devel is connected in auto.master:  

  ipaautomountkey:  

    ipaadmin_password: "{{ ipaadmin_password }}"
    location: raleigh
    mapname: auto.map
    key: /devel
    info: auto.devel
    state: present

```

6. Save the file.

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

```

$ ansible-playbook --vault-password-file=password_file -v -i inventory automount-
location-map-and-key-present.yml

```

6.4. 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.
- On the control node:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package.
 - The example assumes that in the **~/MyPlaybooks/** directory, you have created an **Ansible inventory file** with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipaadmin_password**.
- The target node, that is the node on which the **ansible-freeipa** module is executed, is part of the IdM domain as an IdM client, server or replica.
 - 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:

```
[...]
vars_files:
- /home/user_name/MyPlaybooks/secret.yml
tasks:
[...]
- ipagroup:
  ipaadmin_password: "{{ ipaadmin_password }}"
  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 --vault-password-file=password_file -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
```

6.5. 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
```

6.6. 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 [set up an NFS server with Kerberos on an IdM host](#) .
- 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 [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.