



Red Hat Enterprise Linux 8

Planning Identity Management

Planning the infrastructure and service integration of an IdM environment

Red Hat Enterprise Linux 8 Planning Identity Management

Planning the infrastructure and service integration of an IdM environment

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

Identity Management (IdM) provides a centralized and unified way to manage identity stores, authentication, and authorization policies. For a successful integration of IdM in your environment, learn about the components of IdM and plan the installation. For example, plan a replication topology for failover and load-balancing, the integration into Active Directory (AD), the structure of DNS zones and Certificate Authority (CA), as well as backup and recovery scenarios.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	4
CHAPTER 1. OVERVIEW OF IDM AND ACCESS CONTROL IN RHEL	5
1.1. INTRODUCTION TO IDM	5
1.2. COMMON IDM CUSTOMER SCENARIOS AND THEIR SOLUTIONS	7
1.3. INTRODUCTION TO IDM SERVERS AND CLIENTS	9
1.4. SUPPORTED VERSIONS OF RHEL FOR INSTALLING IDM CLIENTS	10
1.5. IDM AND ACCESS CONTROL IN RHEL: CENTRAL VS. LOCAL	11
1.6. IDM TERMINOLOGY	11
CHAPTER 2. FAILOVER, LOAD-BALANCING, AND HIGH-AVAILABILITY IN IDM	20
Client-side failover capability	20
Primary and backup server configuration	20
Failover behavior for IdM servers and services	20
Server-side load-balancing and service availability	21
CHAPTER 3. PLANNING THE REPLICA TOPOLOGY	22
3.1. MULTIPLE REPLICA SERVERS AS A SOLUTION FOR HIGH PERFORMANCE AND DISASTER RECOVERY	22
3.2. INTRODUCTION TO IDM SERVERS AND CLIENTS	22
3.3. REPLICATION AGREEMENTS BETWEEN IDM REPLICAS	23
3.4. GUIDELINES FOR DETERMINING THE APPROPRIATE NUMBER OF IDM REPLICAS IN A TOPOLOGY	24
3.5. GUIDELINES FOR CONNECTING IDM REPLICAS IN A TOPOLOGY	24
3.6. REPLICA TOPOLOGY EXAMPLES	25
3.7. THE HIDDEN REPLICA MODE	26
CHAPTER 4. PLANNING YOUR DNS SERVICES AND HOST NAMES	28
4.1. DNS SERVICES AVAILABLE IN AN IDM SERVER	28
4.2. GUIDELINES FOR PLANNING THE DNS DOMAIN NAME AND KERBEROS REALM NAME	29
Additional notes on planning the DNS domain name and Kerberos realm name	30
Planning DNS forwarding	30
CHAPTER 5. PLANNING YOUR CA SERVICES	31
5.1. CA SERVICES AVAILABLE IN AN IDM SERVER	31
5.2. GUIDELINES FOR DISTRIBUTION OF CA SERVICES	32
CHAPTER 6. PLANNING INTEGRATION WITH AD	34
6.1. DIRECT INTEGRATION OF LINUX SYSTEMS INTO ACTIVE DIRECTORY	34
Recommendations	34
6.2. INDIRECT INTEGRATION OF LINUX SYSTEMS INTO ACTIVE DIRECTORY BY USING IDENTITY MANAGEMENT	34
6.3. GUIDELINES FOR DECIDING BETWEEN DIRECT AND INDIRECT INTEGRATION	35
Number of systems to be connected to Active Directory	35
Frequency of deploying new systems and their type	36
Active Directory is the required authentication provider	36
CHAPTER 7. PLANNING A CROSS-FOREST TRUST BETWEEN IDM AND AD	37
7.1. CROSS-FOREST AND EXTERNAL TRUSTS BETWEEN IDM AND AD	37
A cross-forest trust between IdM and AD	37
An external trust to an AD domain	37
7.2. TRUST CONTROLLERS AND TRUST AGENTS	37
7.3. ONE-WAY TRUSTS AND TWO-WAY TRUSTS	38
7.4. KERBEROS FAST FOR TRUSTED DOMAINS	39

7.5. POSIX AND ID MAPPING ID RANGE TYPES FOR AD USERS	40
7.6. OPTIONS FOR AUTOMATICALLY MAPPING PRIVATE GROUPS FOR AD USERS: POSIX TRUSTS	40
7.7. OPTIONS FOR AUTOMATICALLY MAPPING PRIVATE GROUPS FOR AD USERS: ID MAPPING TRUSTS	43
7.8. NON-POSIX EXTERNAL GROUPS AND SID MAPPING	44
7.9. GUIDELINES FOR SETTING UP DNS FOR AN IDM-AD TRUST	44
7.10. GUIDELINES FOR CONFIGURING NETBIOS NAMES	45
7.11. SUPPORTED VERSIONS OF WINDOWS SERVER	46
7.12. AD SERVER DISCOVERY AND AFFINITY	46
Options for configuring LDAP and Kerberos on the IdM client for communication with local IdM servers	46
Options for configuring Kerberos on the IdM client for communication with local AD servers	47
Options for configuring embedded clients on IdM servers for communication with local AD servers over Kerberos and LDAP	47
7.13. OPERATIONS PERFORMED DURING INDIRECT INTEGRATION OF IDM TO AD	48
CHAPTER 8. BACKING UP AND RESTORING IDM	50
8.1. IDM BACKUP TYPES	50
8.2. NAMING CONVENTIONS FOR IDM BACKUP FILES	50
8.3. CONSIDERATIONS WHEN CREATING A BACKUP	51
8.4. CREATING AN IDM BACKUP	51
8.5. CREATING A GPG2-ENCRYPTED IDM BACKUP	52
8.6. CREATING A GPG2 KEY	53
8.7. WHEN TO RESTORE FROM AN IDM BACKUP	55
8.8. CONSIDERATIONS WHEN RESTORING FROM AN IDM BACKUP	55
8.9. RESTORING AN IDM SERVER FROM A BACKUP	56
8.10. RESTORING FROM AN ENCRYPTED BACKUP	59
CHAPTER 9. BACKING UP AND RESTORING IDM SERVERS USING ANSIBLE PLAYBOOKS	61
9.1. USING ANSIBLE TO CREATE A BACKUP OF AN IDM SERVER	61
9.2. USING ANSIBLE TO CREATE A BACKUP OF AN IDM SERVER ON YOUR ANSIBLE CONTROLLER	62
9.3. USING ANSIBLE TO COPY A BACKUP OF AN IDM SERVER TO YOUR ANSIBLE CONTROLLER	64
9.4. USING ANSIBLE TO COPY A BACKUP OF AN IDM SERVER FROM YOUR ANSIBLE CONTROLLER TO THE IDM SERVER	65
9.5. USING ANSIBLE TO REMOVE A BACKUP FROM AN IDM SERVER	67
9.6. USING ANSIBLE TO RESTORE AN IDM SERVER FROM A BACKUP STORED ON THE SERVER	68
9.7. USING ANSIBLE TO RESTORE AN IDM SERVER FROM A BACKUP STORED ON YOUR ANSIBLE CONTROLLER	70
CHAPTER 10. IDM INTEGRATION WITH RED HAT PRODUCTS	72
CHAPTER 11. IDM DIRECTORY SERVER RFC SUPPORT	73

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. OVERVIEW OF IDM AND ACCESS CONTROL IN RHEL

Learn how you can use Identity Management (IdM) to centralize identity management, enforce security controls, and comply with best practices and security policies. Explore common customer scenarios and solutions for IdM implementation in both Linux and Windows environments.

1.1. INTRODUCTION TO IDM

Identity Management (IdM) provides a centralized and unified way to manage identity stores, authentication, policies, and authorization policies in a Linux-based domain.

The goal of IdM in Red Hat Enterprise Linux

IdM significantly reduces the administrative overhead of managing different services individually and using different tools on different machines.

IdM is one of the few centralized identity, policy, and authorization software solutions that support:

- Advanced features of Linux operating system environments
- Unifying large groups of Linux machines
- Native integration with Active Directory

IdM creates a Linux-based and Linux-controlled domain:

- IdM builds on existing, native Linux tools and protocols. It has its own processes and configuration, but its underlying technologies are well-established on Linux systems and trusted by Linux administrators.
- IdM servers and clients are Red Hat Enterprise Linux machines. IdM clients can also be other Linux and UNIX distributions if they support standard protocols. A Windows client cannot be a member of the IdM domain but users logged into Windows systems managed by Active Directory (AD) can connect to Linux clients or access services managed by IdM. This is accomplished by establishing cross forest trust between AD and IdM domains.

Managing identities and policies on multiple Linux servers

Without IdM: Each server is administered separately. All passwords are saved on the local machines. The IT administrator manages users on every machine, sets authentication and authorization policies separately, and maintains local passwords. However, more often the users rely on other centralized solution, for example direct integration with AD. Systems can be directly integrated with AD using several different solutions:

- Legacy Linux tools (not recommended to use)
- Solution based on Samba winbind (recommended for specific use cases)
- Solution based on a third-party software (usually require a license from another vendor)
- Solution based on SSSD (native Linux and recommended for the majority of use cases)

With IdM: The IT administrator can:

- Maintain the identities in one central place: the IdM server

- Apply policies uniformly to multiples of machines at the same time
- Set different access levels for users by using host-based access control, delegation, and other rules
- Centrally manage privilege escalation rules
- Define how home directories are mounted

Enterprise SSO

In case of IdM Enterprise, single sign-on (SSO) is implemented leveraging the Kerberos protocol. This protocol is popular in the infrastructure level and enables SSO with services such as SSH, LDAP, NFS, CUPS, or DNS. Web services using different web stacks (Apache, EAP, Django, and others) can also be enabled to use Kerberos for SSO. However, practice shows that using OpenID Connect or SAML based on SSO is more convenient for web applications. To bridge the two layers, it is recommended to deploy an Identity Provider (IdP) solution that would be able to convert Kerberos authentication into a OpenID Connect ticket or SAML assertion. Red Hat SSO technology based on the Keycloak open source project is an example of such an IdP.

Without IdM: Users log in to the system and are prompted for a password every single time they access a service or application. These passwords might be different, and the users have to remember which credential to use for which application.

With IdM: After users log in to the system, they can access multiple services and applications without being repeatedly asked for their credentials. This helps to:

- Improve usability
- Reduce the security risk of passwords being written down or stored insecurely
- Boost user productivity

Managing a mixed Linux and Windows environment

Without IdM: Windows systems are managed in an AD forest, but development, production, and other teams have many Linux systems. The Linux systems are excluded from the AD environment.

With IdM: The IT administrator can:

- Manage the Linux systems using native Linux tools
- Integrate the Linux systems into the environments centrally managed by Active Directory, therefore preserving a centralized user store.
- Easily deploy new Linux systems at scale or as needed.
- Quickly react to business needs and make decisions related to management of the Linux infrastructure without dependency on other teams avoiding delays.

Contrasting IdM with a standard LDAP directory

A standard LDAP directory, such as Red Hat Directory Server, is a general-purpose directory: it can be customized to fit a broad range of use cases.

- Schema: a flexible schema that can be customized for a vast array of entries, such as users, machines, network entities, physical equipment, or buildings.

- Typically used as: a back-end directory to store data for other applications, such as business applications that provide services on the Internet.

IdM has a specific purpose: managing internal, inside-the-enterprise identities as well as authentication and authorization policies that relate to these identities.

- Schema: a specific schema that defines a particular set of entries relevant to its purpose, such as entries for user or machine identities.
- Typically used as: the identity and authentication server to manage identities within the boundaries of an enterprise or a project.

The underlying directory server technology is the same for both Red Hat Directory Server and IdM. However, IdM is optimized to manage identities inside the enterprise. This limits its general extensibility, but also brings certain benefits: simpler configuration, better automation of resource management, and increased efficiency in managing enterprise identities.

Additional resources

- [Identity Management or Red Hat Directory Server – Which One Should I Use?](#) on the Red Hat Enterprise Linux Blog
- [Standard protocols](#) (Red Hat Knowledgebase)

1.2. COMMON IDM CUSTOMER SCENARIOS AND THEIR SOLUTIONS

Explore examples of common identity management and access control use cases both in Linux and Windows environments, and their solutions.

Scenario 1

Situation

You are a Windows administrator in your company.
Apart from Windows systems, you also have several Linux systems to administer.

As you cannot delegate control of any part of your environment to a Linux administrator, you must handle all security controls in Active Directory (AD).

Solution

[Integrate your Linux hosts to AD directly](#)

If you want **sudo** rules to be defined centrally in an LDAP server, you must implement a schema extension in the AD domain controller (DC). If you do not have permissions to implement this extension, consider installing Identity Management (IdM) – see Scenario 3 below. As IdM already contains the schema extension, you can [manage **sudo** rules directly in IdM](#).

Further advice if you are expecting to need more Linux skills in the future

Connect with the Linux community to see how others manage identities: users, hosts, and services. Research best practices.

Make yourself more familiar with Linux:

- Use the [RHEL web console](#) when at all possible.
- Use easy commands on the command-line whenever possible.

- Attend a Red Hat System Administration course.

Scenario 2

Situation

You are a Linux administrator in your company.

Your Linux users require different levels of access to the company resources.

You need tight, centralized access control of your Linux machines.

Solution

[Install IdM](#) and migrate your users to it.

Further advice if you are expecting your company to scale up in the future

After installing IdM, configure [host-based access control](#) and [sudo rules](#). These are necessary to maintain security best practices of limited access and least privilege.

To meet your security targets, develop a cohesive identity and access management (IAM) strategy that uses protocols to secure both infrastructure and application layers.

Scenario 3

Situation

You are a Linux administrator in your company and you must integrate your Linux systems with the company Windows servers. You want to remain the sole maintainer of access control to your Linux systems.

Different users require different levels of access to the Linux systems but they all reside in AD.

Solution

As AD controls are not robust enough, you must configure access control to the Linux systems on the Linux side. [Install IdM](#) and [establish an IdM-AD trust](#).

Further advice to enhance the security of your environment

After installing IdM, configure [host-based access control](#) and [sudo rules](#). These are necessary to maintain security best practices of limited access and least privilege.

To meet your security targets, develop a cohesive Identity and Access Management (IAM) strategy that uses protocols to secure both infrastructure and application layers.

Scenario 4

Situation

As a security administrator, you must manage identities and access across all of your environments, including all of your Red Hat products. You must manage all of your identities in one place, and maintain access controls across all of your platforms, clouds and products.

Solution

Integrate IdM, [Red Hat Single Sign-On](#), [Red Hat Satellite](#), [Red Hat Ansible Automation Platform](#) and other Red Hat products.

Scenario 5

Situation

As a security and system administrator in a Department of Defense (DoD) or Intelligence Community (IC) environment, you are required to use smart card or RSA authentication. You are required to use PIV certificates or RSA tokens.

Solution

1. [Configure certificate mapping in IdM](#).
2. Ensure that GSSAPI delegation is enabled if an IdM-AD trust is present.
3. Configure the use of radius configuration in IdM for RSA tokens.
4. Configure IdM servers and IdM clients for [smart card authentication](#).

Additional resources

- [Use Ansible to automate your IdM tasks](#) to reduce client configuration time and complexity and to reduce mistakes.

1.3. INTRODUCTION TO IDM SERVERS AND CLIENTS

The Identity Management (IdM) domain includes the following types of systems:

IdM clients

IdM clients are Red Hat Enterprise Linux systems enrolled with the servers and configured to use the IdM services on these servers.

Clients interact with the IdM servers to access services provided by them. For example, clients use the Kerberos protocol to perform authentication and acquire tickets for enterprise single sign-on (SSO), use LDAP to get identity and policy information, and use DNS to detect where the servers and services are located and how to connect to them.

IdM servers

IdM servers are Red Hat Enterprise Linux systems that respond to identity, authentication, and authorization requests from IdM clients within an IdM domain. IdM servers are the central repositories for identity and policy information. They can also host any of the optional services used by domain members:

- [Certificate authority](#) (CA): This service is present in most IdM deployments.
- Key Recovery Authority (KRA)
- DNS
- Active Directory (AD) trust controller
- Active Directory (AD) trust agent

IdM servers are also embedded IdM clients. As clients enrolled with themselves, the servers provide the same functionality as other clients.

To provide services for large numbers of clients, as well as for redundancy and availability, IdM allows deployment on multiple IdM servers in a single domain. It is possible to deploy up to 60 servers. This is the maximum number of IdM servers, also called replicas, that is currently supported in the IdM domain.

When creating a replica, IdM clones the configuration of the existing server. A replica shares with the initial server its core configuration, including internal information about users, systems, certificates, and configured policies.

NOTE

A replica and the server it was created from are functionally identical, except for the *CA renewal* and *CRL publisher* roles. Therefore, the term **server** and **replica** are used interchangeably in RHEL IdM documentation, depending on the context.

However, different IdM servers can provide different services for the client, if so configured. Core components like Kerberos and LDAP are available on every server. Other services like CA, DNS, Trust Controller or Vault are optional. This means that different IdM servers can have distinct roles in the deployment.

If your IdM topology contains an integrated CA, one server has the role of the [Certificate revocation list \(CRL\) publisher server](#) and one server has the role of the [CA renewal server](#).

By default, the first CA server installed fulfills these two roles, but you can assign these roles to separate servers.



WARNING

The *CA renewal server* is critical for your IdM deployment because it is the only system in the domain responsible for tracking CA subsystem [certificates and keys](#). For details about how to recover from a disaster affecting your IdM deployment, see [Performing disaster recovery with Identity Management](#).

NOTE

All IdM servers (for clients, see [Supported versions of RHEL for installing IdM clients](#)) must be running on the same major and minor version of RHEL. Do not spend more than several days applying z-stream updates or upgrading the IdM servers in your topology. For details about how to apply Z-stream fixes and upgrade your servers, see [Updating IdM packages](#). For details about how to migrate to IdM on RHEL 8, see [Migrating your IdM environment from RHEL 7 servers to RHEL 8 servers](#).

1.4. SUPPORTED VERSIONS OF RHEL FOR INSTALLING IDM CLIENTS

An Identity Management deployment in which IdM servers are running on the latest minor version of Red Hat Enterprise Linux 8 supports clients that are running on the latest minor versions of:

- RHEL 7
- RHEL 8
- RHEL 9



NOTE

While other client systems, for example Ubuntu, can work with IdM 8 servers, Red Hat does not provide support for these clients.



IMPORTANT

If you are planning to make your IdM deployment FIPS-compliant, Red Hat strongly recommends migrating your environment to RHEL 9. RHEL 9 is the first major RHEL version that is planned to be compliant with FIPS 140-3.

1.5. IDM AND ACCESS CONTROL IN RHEL: CENTRAL VS. LOCAL

In Red Hat Enterprise Linux, you can manage identities and access control policies using centralized tools for a whole domain of systems, or using local tools for a single system.

Managing identities and policies on multiple Red Hat Enterprise Linux servers

With Identity Management IdM, the IT administrator can:

- Maintain the identities and grouping mechanisms in one central place: the IdM server
- Centrally manage different types of credentials such as passwords, PKI certificates, OTP tokens, or SSH keys
- Apply policies uniformly to multiples of machines at the same time
- Manage POSIX and other attributes for external Active Directory users
- Set different access levels for users by using host-based access control, delegation, and other rules
- Centrally manage privilege escalation rules (sudo) and mandatory access control (SELinux user mapping)
- Maintain central PKI infrastructure and secrets store
- Define how home directories are mounted

Without IdM:

- Each server is administered separately
- All passwords are saved on the local machines
- The IT administrator manages users on every machine, sets authentication and authorization policies separately, and maintains local passwords

1.6. IDM TERMINOLOGY

Active Directory forest

An Active Directory (AD) forest is a set of one or more domain trees which share a common global catalog, directory schema, logical structure, and directory configuration. The forest represents the security boundary within which users, computers, groups, and other objects are accessible. For more information, see the Microsoft document on [Forests](#).

Active Directory global catalog

The global catalog is a feature of Active Directory (AD) that allows a domain controller to provide information about any object in the forest, regardless of whether the object is a member of the domain controller's domain. Domain controllers with the global catalog feature enabled are referred

to as global catalog servers. The global catalog provides a searchable catalog of all objects in every domain in a multi-domain Active Directory Domain Services (AD DS).

Active Directory security identifier

A security identifier (SID) is a unique ID number assigned to an object in Active Directory, such as a user, group, or host. It is the functional equivalent of UIDs and GIDs in Linux.

Ansible play

Ansible plays are the building blocks of [Ansible playbooks](#). The goal of a play is to map a group of hosts to some well-defined roles, represented by Ansible tasks.

Ansible playbook

An Ansible playbook is a file that contains one or more Ansible plays. For more information, see the [official Ansible documentation about playbooks](#).

Ansible task

Ansible tasks are units of action in Ansible. An Ansible play can contain multiple tasks. The goal of each task is to execute a module, with very specific arguments. An Ansible task is a set of instructions to achieve a state defined, in its broad terms, by a specific Ansible role or module, and fine-tuned by the variables of that role or module. For more information, see the [official Ansible tasks documentation](#).

Apache web server

The Apache HTTP Server, colloquially called Apache, is a free and open source cross-platform web server application, released under the terms of Apache License 2.0. Apache played a key role in the initial growth of the World Wide Web, and is currently the leading HTTP server. Its process name is **httpd**, which is short for *HTTP daemon*. Red Hat Enterprise Linux Identity Management (IdM) uses the Apache Web Server to display the IdM Web UI, and to coordinate communication between components, such as the Directory Server and the Certificate Authority.

Certificate

A certificate is an electronic document used to identify an individual, a server, a company, or other entity and to associate that identity with a public key. Such as a driver's license or passport, a certificate provides generally recognized proof of a person's identity. Public-key cryptography uses certificates to address the problem of impersonation.

Certificate Authorities (CAs) in IdM

An entity that issues digital certificates. In Red Hat Identity Management, the primary CA is **ipa**, the IdM CA. The **ipa** CA certificate is one of the following types:

- Self-signed. In this case, the **ipa** CA is the root CA.
- Externally signed. In this case, the **ipa** CA is subordinated to the external CA.

In IdM, you can also create multiple **sub-CAs**. Sub-CAs are IdM CAs whose certificates are one of the following types:

- Signed by the **ipa** CA.
- Signed by any of the intermediate CAs between itself and **ipa** CA. The certificate of a sub-CA cannot be self-signed.

See also [Planning your CA services](#).

Cross-forest trust

A trust establishes an access relationship between two Kerberos realms, allowing users and services in one domain to access resources in another domain.

With a cross-forest trust between an Active Directory (AD) forest root domain and an IdM domain, users from the AD forest domains can interact with Linux machines and services from the IdM domain. From the perspective of AD, Identity Management represents a separate AD forest with a single AD domain. For more information, see [How the trust between IdM and AD works](#).

Directory Server

A Directory Server centralizes user identity and application information. It provides an operating system-independent, network-based registry for storing application settings, user profiles, group data, policies, and access control information. Each resource on the network is considered an object by the Directory Server. Information about a particular resource is stored as a collection of attributes associated with that resource or object. Red Hat Directory Server conforms to LDAP standards.

DNS PTR records

DNS pointer (PTR) records resolve an IP address of a host to a domain or host name. PTR records are the opposite of DNS A and AAAA records, which resolve host names to IP addresses. DNS PTR records enable reverse DNS lookups. PTR records are stored on the DNS server.

DNS SRV records

A DNS service (SRV) record defines the hostname, port number, transport protocol, priority and weight of a service available in a domain. You can use SRV records to locate IdM servers and replicas.

Domain Controller (DC)

A domain controller (DC) is a host that responds to security authentication requests within a domain and controls access to resources in that domain. IdM servers work as DCs for the IdM domain. A DC authenticates users, stores user account information and enforces security policy for a domain. When a user logs into a domain, the DC authenticates and validates their credentials and either allows or denies access.

Fully qualified domain name

A fully qualified domain name (FQDN) is a domain name that specifies the exact location of a host within the hierarchy of the Domain Name System (DNS). A device with the hostname **myhost** in the parent domain **example.com** has the FQDN **myhost.example.com**. The FQDN uniquely distinguishes the device from any other hosts called **myhost** in other domains.

If you are installing an IdM client on host **machine1** using DNS autodiscovery and your DNS records are correctly configured, the FQDN of **machine1** is all you need. For more information, see [Host name and DNS requirements for IdM](#).

GSSAPI

The Generic Security Service Application Program Interface (GSSAPI, or GSS-API) allows developers to abstract how their applications protect data that is sent to peer applications. Security-service vendors can provide GSSAPI implementations of common procedure calls as libraries with their security software. These libraries present a GSSAPI-compatible interface to application writers who can write their application to use only the vendor-independent GSSAPI. With this flexibility, developers do not have to tailor their security implementations to any particular platform, security mechanism, type of protection, or transport protocol.

Kerberos is the dominant GSSAPI mechanism implementation, which allows Red Hat Enterprise Linux and Microsoft Windows Active Directory Kerberos implementations to be API compatible.

Hidden replica

A hidden replica is an IdM replica that has all services running and available, but its server roles are disabled, and clients cannot discover the replica because it has no SRV records in DNS.

Hidden replicas are primarily designed for services such as backups, bulk importing and exporting, or actions that require shutting down IdM services. Since no clients use a hidden replica, administrators can temporarily shut down the services on this host without affecting any clients. For more information, see [The hidden replica mode](#).

HTTP server

See [Web server](#).

ID mapping

SSSD can use the SID of an AD user to algorithmically generate POSIX IDs in a process called *ID mapping*. ID mapping creates a map between SIDs in AD and IDs on Linux.

- When SSSD detects a new AD domain, it assigns a range of available IDs to the new domain. Therefore, each AD domain has the same ID range on every SSSD client machine.
- When an AD user logs in to an SSSD client machine for the first time, SSSD creates an entry for the user in the SSSD cache, including a UID based on the user's SID and the ID range for that domain.
- Because the IDs for an AD user are generated in a consistent way from the same SID, the user has the same UID and GID when logging in to any Red Hat Enterprise Linux system.

ID ranges

An ID range is a range of ID numbers assigned to the IdM topology or a specific replica. You can use ID ranges to specify the valid range of UIDs and GIDs for new users, hosts and groups. ID ranges are used to avoid ID number conflicts. There are two distinct types of ID ranges in IdM:

- *IdM ID range*
Use this ID range to define the UIDs and GIDs for users and groups in the whole IdM topology. Installing the first IdM server creates the IdM ID range. You cannot modify the IdM ID range after creating it. However, you can create an additional IdM ID range, for example when the original one nears depletion.
- *Distributed Numeric Assignment (DNA) ID range*
Use this ID range to define the UIDs and GIDs a replica uses when creating new users. Adding a new user or host entry to an IdM replica for the first time assigns a DNA ID range to that replica. An administrator can modify the DNA ID range, but the new definition must fit within an existing IdM ID range.

Note that the IdM range and the DNA range match, but they are not interconnected. If you change one range, ensure you change the other to match.

For more information, see [ID ranges](#).

ID views

ID views enable you to specify new values for POSIX user or group attributes, and to define on which client host or hosts the new values will apply. For example, you can use ID views to:

- Define different attribute values for different environments.
- Replace a previously generated attribute value with a different value.

In an IdM-AD trust setup, the **Default Trust View** is an ID view applied to AD users and groups. Using the **Default Trust View**, you can define custom POSIX attributes for AD users and groups, therefore overriding the values defined in AD.

For more information, see [Using an ID view to override a user attribute value on an IdM client](#) .

IdM CA server

An IdM server on which the IdM certificate authority (CA) service is installed and running.

Alternative names: **CA server**

IdM deployment

A term that refers to the entirety of your IdM installation. You can describe your IdM deployment by answering the following questions:

- Is your IdM deployment a testing deployment or production deployment?
 - How many IdM servers do you have?
- Does your IdM deployment contain [an integrated CA](#)?
 - If it does, is the integrated CA self-signed or externally signed?
 - If it does, on which servers is the [CA role](#) available? On which servers is the KRA role available?
- Does your IdM deployment contain [an integrated DNS](#)?
 - If it does, on which servers is the DNS role available?
- Is your IdM deployment in a trust agreement with an [AD forest](#)?
 - If it is, on which servers is the [AD trust controller](#) or [AD trust agent](#) role available?

IdM server and replicas

To install the first server in an IdM deployment, you must use the **ipa-server-install** command. Administrators can then use the **ipa-replica-install** command to install **replicas** in addition to the first server that was installed. By default, installing a replica creates a [replication agreement](#) with the IdM server from which it was created, enabling receiving and sending updates to the rest of IdM.

There is no functional difference between the first server that was installed and a replica. Both are fully functional read/write [IdM servers](#).

Deprecated names: **master server**

IdM CA renewal server

If your IdM topology contains an integrated certificate authority (CA), one server has the unique role of the [CA renewal server](#). This server maintains and renews IdM system certificates.

By default, the first CA server you install fulfills this role, but you can configure any CA server to be the CA renewal server. In a deployment without integrated CA, there is no CA renewal server.

Deprecated names: **master CA**

IdM CRL publisher server

If your IdM topology contains an integrated certificate authority (CA), one server has the unique role of the [Certificate revocation list \(CRL\) publisher server](#). This server is responsible for maintaining the CRL.

By default, the server that fulfills the **CA renewal server** role also fulfills this role, but you can configure any CA server to be the CRL publisher server. In a deployment without integrated CA, there is no CRL publisher server.

IdM topology

A term that refers to the [structure of your IdM solution](#), especially the replication agreements between and within individual data centers and clusters.

Kerberos authentication indicators

Authentication indicators are attached to Kerberos tickets and represent the initial authentication method used to acquire a ticket:

- **otp** for two-factor authentication (password + One-Time Password)
- **radius** for Remote Authentication Dial-In User Service (RADIUS) authentication (commonly for 802.1x authentication)
- **pkinit** for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT), smart card, or certificate authentication
- **hardened** for passwords hardened against brute-force attempts

For more information, see [Kerberos authentication indicators](#).

Kerberos keytab

While a password is the default authentication method for a user, keytabs are the default authentication method for hosts and services. A Kerberos keytab is a file that contains a list of Kerberos principals and their associated encryption keys, so a service can retrieve its own Kerberos key and verify a user's identity.

For example, every IdM client has an **/etc/krb5.keytab** file that stores information about the **host** principal, which represents the client machine in the Kerberos realm.

Kerberos principal

Unique Kerberos principals identify each user, service, and host in a Kerberos realm:

Entity	Naming convention	Example
Users	identifier@REALM	admin@EXAMPLE.COM
Services	service/fully-qualified-hostname@REALM	http/server.example.com@EXAMPLE.COM
Hosts	host/fully-qualified-hostname@REALM	host/client.example.com@EXAMPLE.COM

Kerberos protocol

Kerberos is a network authentication protocol that provides strong authentication for client and server applications by using secret-key cryptography. IdM and Active Directory use Kerberos for authenticating users, hosts and services.

Kerberos realm

A Kerberos realm encompasses all the principals managed by a Kerberos Key Distribution Center (KDC). In an IdM deployment, the Kerberos realm includes all IdM users, hosts, and services.

Kerberos ticket policies

The Kerberos Key Distribution Center (KDC) enforces ticket access control through connection policies, and manages the duration of Kerberos tickets through ticket lifecycle policies. For example, the default global ticket lifetime is one day, and the default global maximum renewal age is one week. For more information, see [IdM Kerberos ticket policy types](#).

Key Distribution Center (KDC)

The Kerberos Key Distribution Center (KDC) is a service that acts as the central, trusted authority that manages Kerberos credential information. The KDC issues Kerberos tickets and ensures the authenticity of data originating from entities within the IdM network.

For more information, see [The role of the IdM KDC](#).

LDAP

The Lightweight Directory Access Protocol (LDAP) is an open, vendor-neutral, application protocol for accessing and maintaining distributed directory information services over a network. Part of this specification is a directory information tree (DIT), which represents data in a hierarchical tree-like structure consisting of the Distinguished Names (DNs) of directory service entries. LDAP is a "lightweight" version of the Directory Access Protocol (DAP) described by the ISO X.500 standard for directory services in a network.

Lightweight sub-CA

In IdM, a lightweight sub-CA is a certificate authority (CA) whose certificate is signed by an IdM root CA or one of the CAs that are subordinate to it. A lightweight sub-CA issues certificates only for a specific purpose, for example to secure a VPN or HTTP connection.

For more information, see [Restricting an application to trust only a subset of certificates](#).

Password policy

A password policy is a set of conditions that the passwords of a particular IdM user group must meet. The conditions can include the following parameters:

- The length of the password
- The number of character classes used
- The maximum lifetime of a password.

For more information, see [What is a password policy](#).

POSIX attributes

POSIX attributes are user attributes for maintaining compatibility between operating systems.

In a Red Hat Enterprise Linux Identity Management environment, POSIX attributes for users include:

- **cn**, the user's name
- **uid**, the account name (login)
- **uidNumber**, a user number (UID)
- **gidNumber**, the primary group number (GID)
- **homeDirectory**, the user's home directory

In a Red Hat Enterprise Linux Identity Management environment, POSIX attributes for groups include:

- **cn**, the group's name
- **gidNumber**, the group number (GID)

These attributes identify users and groups as separate entities.

Replication agreement

A replication agreement is an agreement between two IdM servers in the same IdM deployment. The replication agreement ensures that the data and configuration is continuously replicated between the two servers.

IdM uses two types of replication agreements: *domain replication* agreements, which replicate identity information, and *certificate replication* agreements, which replicate certificate information.

For more information, see:

- [Replication agreements](#)
- [Determining the appropriate number of replicas](#)
- [Connecting the replicas in a topology](#)
- [Replica topology examples](#)

Smart card

A smart card is a removable device or card used to control access to a resource. They can be plastic credit card-sized cards with an embedded integrated circuit (IC) chip, small USB devices such as a Yubikey, or other similar devices. Smart cards can provide authentication by allowing users to connect a smart card to a host computer, and software on that host computer interacts with key material stored on the smart card to authenticate the user.

SSSD

The System Security Services Daemon (SSSD) is a system service that manages user authentication and user authorization on a RHEL host. SSSD optionally keeps a cache of user identities and credentials retrieved from remote providers for offline authentication. For more information, see [Understanding SSSD and its benefits](#).

SSSD backend

An SSSD backend, often also called a data provider, is an SSSD child process that manages and creates the SSSD cache. This process communicates with an LDAP server, performs different lookup queries and stores the results in the cache. It also performs online authentication against LDAP or Kerberos and applies access and password policy to the user that is logging in.

Ticket-granting ticket (TGT)

After authenticating to a Kerberos Key Distribution Center (KDC), a user receives a ticket-granting ticket (TGT), which is a temporary set of credentials that can be used to request access tickets to other services, such as websites and email.

Using a TGT to request further access provides the user with a Single Sign-On experience, as the user only needs to authenticate once to access multiple services. TGTs are renewable, and Kerberos ticket policies determine ticket renewal limits and access control.

For more information, see [Managing Kerberos ticket policies](#).

Web server

A web server is computer software and underlying hardware that accepts requests for web content, such as pages, images, or applications. A user agent, such as a web browser, requests a specific resource using HTTP, the network protocol used to distribute web content, or its secure variant HTTPS. The web server responds with the content of that resource or an error message. The web server can also accept and store resources sent from the user agent. Red Hat Enterprise Linux Identity Management (IdM) uses the Apache Web Server to display the IdM Web UI, and to coordinate communication between components, such as the Directory Server and the Certificate Authority (CA). See [Apache web server](#).

Additional Glossaries

If you are unable to find an Identity Management term in this glossary, see the Directory Server and Certificate System glossaries:

- [Directory Server 11 Glossary](#)
- [Certificate System 9 Glossary](#)

CHAPTER 2. FAILOVER, LOAD-BALANCING, AND HIGH-AVAILABILITY IN IDM

Identity Management (IdM) has built-in failover mechanisms for IdM clients, and load-balancing and high-availability features for IdM servers.

Client-side failover capability

By default, the SSSD service on an IdM client is configured to use DNS service (SRV) resource records so that the client can automatically determine the best IdM server to connect to.

Primary and backup server configuration

The server resolution behavior is controlled by the `_srv_` option in the `ipa_server` parameter of the `/etc/sss/sss.conf` file:

Example `/etc/sss/sss.conf`

```
[domain/<idm_domain_name>]
id_provider = ipa
ipa_server = _srv_, <primary_idm_server1>, <primary_idm_server2>
ipa_backup_server = <backup_idm_server1>, <backup_idm_server2>
...
```

With the `_srv_` option specified, SSSD retrieves a list of IdM servers ordered by preference. If a primary server goes offline, the SSSD service on the IdM client automatically connects to another available IdM server.

Primary servers are specified in the `ipa_server` parameter. SSSD attempts to connect to primary servers first and switches to backup servers only if no primary servers are available.

The `_srv_` option is not supported for backup servers.



NOTE

SSSD queries SRV records from the DNS server. By default, SSSD waits for **6** seconds for a reply from the DNS resolver before attempting to query another DNS server. If all DNS servers are unreachable, the domain will continue to operate in offline mode. You can use the `dns_resolver_timeout` option to increase the time the client waits for a reply from the DNS resolver.

If you prefer to bypass DNS lookups for performance reasons, remove the `_srv_` entry from the `ipa_server` parameter and specify which IdM servers the client should connect to, in order of preference:

Example `/etc/sss/sss.conf`

```
[domain/<idm_domain_name>]
id_provider = ipa
ipa_server = <primary_idm_server1>, <primary_idm_server2>
ipa_backup_server = <backup_idm_server1>, <backup_idm_server2>
...
```

Failover behavior for IdM servers and services

SSSD failover mechanism treats an IdM server and its services independently. If the hostname resolution

for a server succeeds, SSSD considers the machine is online and tries to connect to the required service on that machine. If the connection to the service fails, SSSD considers only that specific service as offline, not the entire machine or other services on it.

If hostname resolution fails, SSSD considers the entire machine as offline, and does not attempt to connect to any services on that machine.

When all primary servers are unavailable, SSSD attempts to connect to a configured backup server. While connected to a backup server, SSSD periodically attempts to reconnect to one of the primary servers and connects immediately once a primary server becomes available. The interval between these attempts is controlled by the **failover_primary_timeout** option, which defaults to 31 seconds.

If all IdM servers become unreachable, SSSD switches to offline mode. In this state, SSSD retries connections every 30 seconds until a server becomes available.

Server-side load-balancing and service availability

You can achieve load-balancing and high-availability in IdM by installing multiple IdM replicas:

- If you have a geographically dispersed network, you can shorten the path between IdM clients and the nearest accessible server by configuring multiple IdM replicas per data center.
- Red Hat supports environments with up to 60 replicas.
- The IdM replication mechanism provides active/active service availability: services at all IdM replicas are readily available at the same time.



NOTE

Red Hat recommends against combining IdM and other load-balancing or high-availability (HA) software.

Many third-party high availability solutions assume active/passive scenarios and cause unnecessary service interruption to IdM availability. Other solutions use virtual IPs or a single hostname per clustered service. All these methods do not typically work well with the type of service availability provided by the IdM solution. They also integrate very poorly with Kerberos, decreasing the overall security and stability of the deployment.

Additional resources

- **sssd.conf(5)** man pages on your system

CHAPTER 3. PLANNING THE REPLICA TOPOLOGY

Review guidance on determining the appropriate replica topology for your use case.

3.1. MULTIPLE REPLICA SERVERS AS A SOLUTION FOR HIGH PERFORMANCE AND DISASTER RECOVERY

You can achieve continuous functionality and high-availability of Identity Management (IdM) services by creating replicas of the existing IdM servers.

When you create an appropriate number of IdM replicas, you can use load balancing to distribute client requests across multiple servers to optimize performance of IdM services. With IdM, you can place additional servers in geographically dispersed data centers to reflect your enterprise organizational structure. In this way, the path between IdM clients and the nearest accessible server is shortened. In addition, having multiple servers allows spreading the load and scaling for more clients.

Replicating IdM servers is also a common backup mechanism to mitigate or prevent server loss. For example, if one server fails, the remaining servers continue providing services to the domain. You can also recover the lost server by creating a new replica based on one of the remaining servers.

3.2. INTRODUCTION TO IDM SERVERS AND CLIENTS

The Identity Management (IdM) domain includes the following types of systems:

IdM clients

IdM clients are Red Hat Enterprise Linux systems enrolled with the servers and configured to use the IdM services on these servers.

Clients interact with the IdM servers to access services provided by them. For example, clients use the Kerberos protocol to perform authentication and acquire tickets for enterprise single sign-on (SSO), use LDAP to get identity and policy information, and use DNS to detect where the servers and services are located and how to connect to them.

IdM servers

IdM servers are Red Hat Enterprise Linux systems that respond to identity, authentication, and authorization requests from IdM clients within an IdM domain. IdM servers are the central repositories for identity and policy information. They can also host any of the optional services used by domain members:

- [Certificate authority](#) (CA): This service is present in most IdM deployments.
- Key Recovery Authority (KRA)
- DNS
- Active Directory (AD) trust controller
- Active Directory (AD) trust agent

IdM servers are also embedded IdM clients. As clients enrolled with themselves, the servers provide the same functionality as other clients.

To provide services for large numbers of clients, as well as for redundancy and availability, IdM allows deployment on multiple IdM servers in a single domain. It is possible to deploy up to 60 servers. This is the maximum number of IdM servers, also called replicas, that is currently supported in the IdM domain.

When creating a replica, IdM clones the configuration of the existing server. A replica shares with the initial server its core configuration, including internal information about users, systems, certificates, and configured policies.

NOTE

A replica and the server it was created from are functionally identical, except for the *CA renewal* and *CRL publisher* roles. Therefore, the term **server** and **replica** are used interchangeably in RHEL IdM documentation, depending on the context.

However, different IdM servers can provide different services for the client, if so configured. Core components like Kerberos and LDAP are available on every server. Other services like CA, DNS, Trust Controller or Vault are optional. This means that different IdM servers can have distinct roles in the deployment.

If your IdM topology contains an integrated CA, one server has the role of the [Certificate revocation list \(CRL\) publisher server](#) and one server has the role of the [CA renewal server](#).

By default, the first CA server installed fulfills these two roles, but you can assign these roles to separate servers.



WARNING

The *CA renewal server* is critical for your IdM deployment because it is the only system in the domain responsible for tracking CA subsystem [certificates and keys](#). For details about how to recover from a disaster affecting your IdM deployment, see [Performing disaster recovery with Identity Management](#).

NOTE

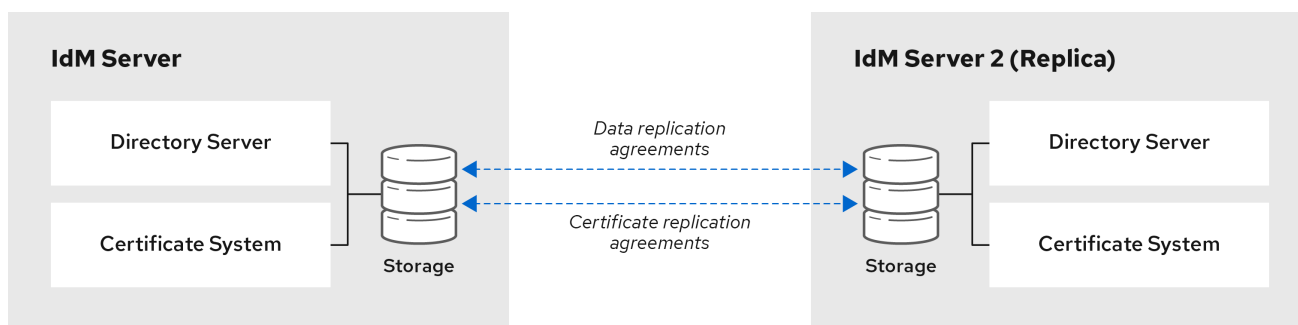
All IdM servers (for clients, see [Supported versions of RHEL for installing IdM clients](#)) must be running on the same major and minor version of RHEL. Do not spend more than several days applying z-stream updates or upgrading the IdM servers in your topology. For details about how to apply Z-stream fixes and upgrade your servers, see [Updating IdM packages](#). For details about how to migrate to IdM on RHEL 8, see [Migrating your IdM environment from RHEL 7 servers to RHEL 8 servers](#).

3.3. REPLICATION AGREEMENTS BETWEEN IDM REPLICAS

When an administrator creates a replica based on an existing server, Identity Management (IdM) creates a *replication agreement* between the initial server and the replica. The replication agreement ensures that the data and configuration is continuously replicated between the two servers.

IdM uses *multiple read/write replica replication*. In this configuration, all replicas joined in a replication agreement receive and provide updates, and are therefore considered suppliers and consumers. Replication agreements are always bilateral.

Figure 3.1. Server and replica agreements



64_RHEL_0120

IdM uses two types of replication agreements:

- **Domain replication agreements** replicate the identity information.
- **Certificate replication agreements** replicate the certificate information.

Both replication channels are independent. Two servers can have one or both types of replication agreements configured between them. For example, when server A and server B have only domain replication agreement configured, only identity information is replicated between them, not the certificate information.

3.4. GUIDELINES FOR DETERMINING THE APPROPRIATE NUMBER OF IDM REPLICAS IN A TOPOLOGY

Plan IdM topology to match your organization's requirements and ensure optimal performance and service availability.

Set up at least two replicas in each data center

Deploy at least two replicas in each data center to ensure that if one server fails, the replica can take over and handle requests.

Set up a sufficient number of servers to serve your clients

One Identity Management (IdM) server can provide services to 2000 - 3000 clients. This assumes the clients query the servers multiple times a day, but not, for example, every minute. If you expect frequent queries, plan for more servers.

Set up a sufficient number of Certificate Authority (CA) replicas

Only replicas with the CA role installed can replicate certificate data. If you use the IdM CA, ensure your environment has at least two CA replicas with certificate replication agreements between them.

Set up a maximum of 60 replicas in a single IdM domain

Red Hat supports environments with up to 60 replicas.

3.5. GUIDELINES FOR CONNECTING IDM REPLICAS IN A TOPOLOGY

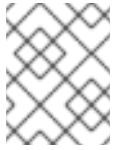
Connect each replica to at least two other replicas

This ensures that information is replicated not just between the initial replica and the first server you installed, but between other replicas as well.

Connect a replica to a maximum of four other replicas (not a hard requirement)

A large number of replication agreements per server does not add significant benefits. A receiving replica can only be updated by one other replica at a time and meanwhile, the other replication

agreements are idle. More than four replication agreements per replica typically means a waste of resources.



NOTE

This recommendation applies to both certificate replication and domain replication agreements.

There are two exceptions to the limit of four replication agreements per replica:

- You want failover paths if certain replicas are not online or responding.
- In larger deployments, you want additional direct links between specific nodes.

Configuring a high number of replication agreements can have a negative impact on overall performance: when multiple replication agreements in the topology are sending updates, certain replicas can experience a high contention on the changelog database file between incoming updates and the outgoing updates.

If you decide to use more replication agreements per replica, ensure that you do not experience replication issues and latency. However, note that large distances and high numbers of intermediate nodes can also cause latency problems.

Connect the replicas in a data center with each other

This ensures domain replication within the data center.

Connect each data center to at least two other data centers

This ensures domain replication between data centers.

Connect data centers using at least a pair of replication agreements

If data centers A and B have a replication agreement from A1 to B1, having a replication agreement from A2 to B2 ensures that if one of the servers is down, the replication can continue between the two data centers.

3.6. REPLICA TOPOLOGY EXAMPLES

You can create a reliable replica topology by using one of the following examples.

Figure 3.2. Replica topology with four data centers, each with four servers that are connected with replication agreements

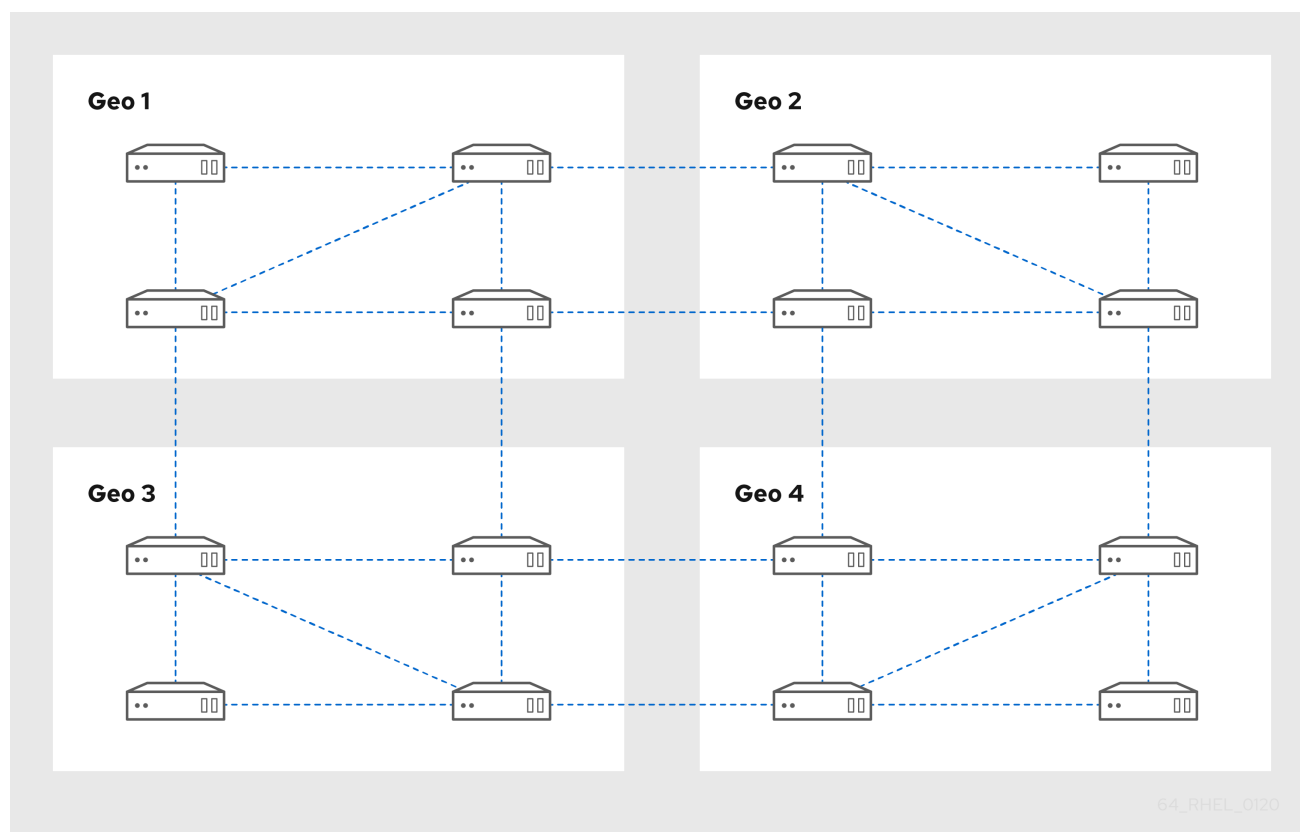
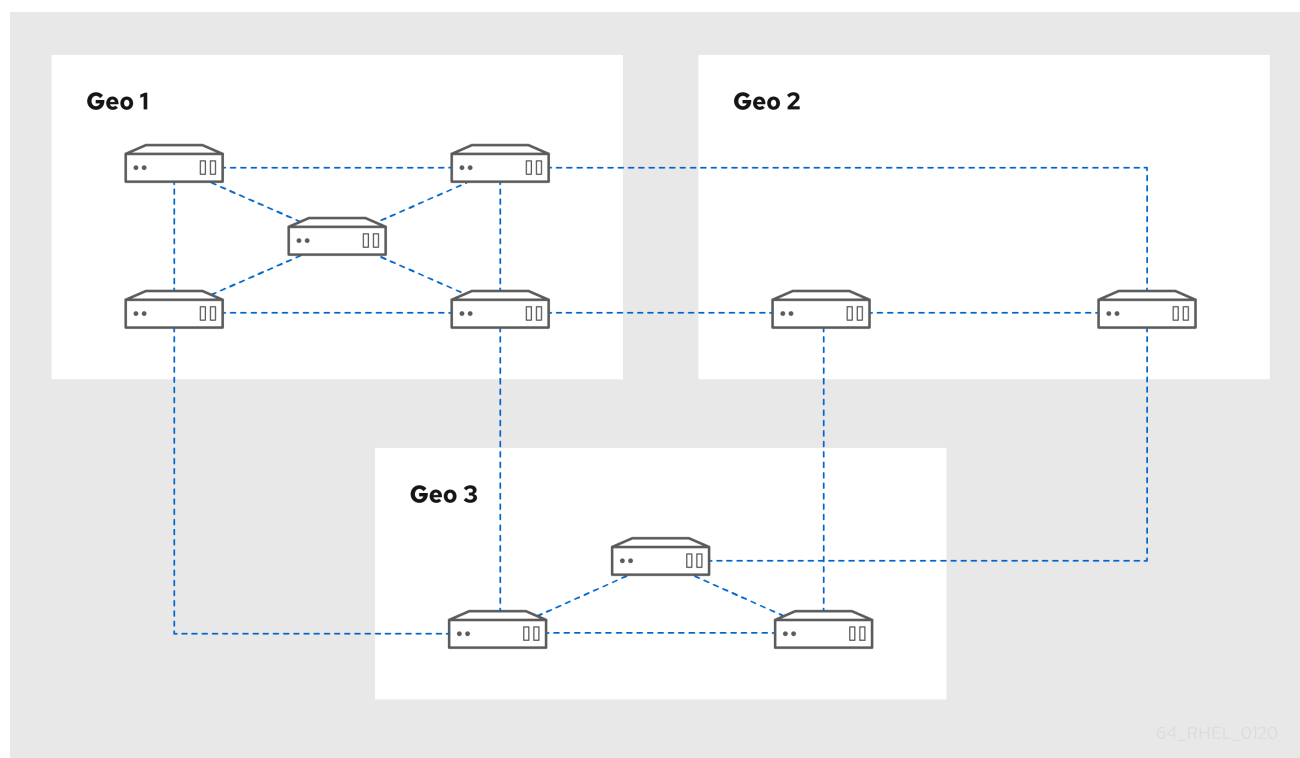


Figure 3.3. Replica topology with three data centers, each with a different number of servers that are all interconnected through replication agreements



3.7. THE HIDDEN REPLICA MODE

A **hidden replica** is an IdM server that has all services running and available. However, a hidden replica has no SRV records in DNS, and LDAP server roles are not enabled. Therefore, clients cannot use service discovery to detect hidden replicas.

By default, when you set up a replica, the installation program automatically creates service (SRV) resource records for it in DNS. These records enable clients to auto-discover the replica and its services. When installing a replica as hidden, add the **--hidden-replica** parameter to the **ipa-replica-install** command.



NOTE

The hidden replica feature, introduced in RHEL 8.1 as a Technology Preview, is fully supported starting with RHEL 8.2.

Hidden replicas are primarily designed for dedicated services that might disrupt clients. For example, a full backup of IdM requires shutting down all IdM services on the server. As no clients use a hidden replica, administrators can temporarily shut down the services on this host without affecting any clients.

Other use cases include high-load operations on the IdM API or the LDAP server, such as a mass import or extensive queries.

Before backing up a hidden replica, you must install all required server roles used in a cluster, especially the Certificate Authority role if the integrated CA is used. Therefore, restoring a backup from a hidden replica on a new host always results in a regular replica.

Additional resources

- [Installing an Identity Management replica](#)
- [Backing up and restoring IdM](#)
- [Demoting or promoting hidden replicas](#)

CHAPTER 4. PLANNING YOUR DNS SERVICES AND HOST NAMES

Identity Management (IdM) provides different types of DNS configurations in the IdM server. The following sections describe them and provide advice on how to determine which is the best for your use case.

4.1. DNS SERVICES AVAILABLE IN AN IDM SERVER

You can install an Identity Management (IdM) server with or without integrated DNS.

Table 4.1. Comparing IdM with integrated DNS and without integrated DNS

	With integrated DNS	Without integrated DNS
Overview:	IdM runs its own DNS service for the IdM domain.	IdM uses DNS services provided by an external DNS server.
Limitations:	<p>The integrated DNS server provided by IdM only supports features related to IdM deployment and maintenance. It does not support some of the advanced features of a general-purpose DNS server. Specific limitations are as follows:</p> <ul style="list-style-type: none"> ● IdM DNS nameserver must be authoritative for its zones. ● The supported record types are A, AAAA, A6, AFSDDB, CERT, CNAME, DLV, DNAME, DS, KX, LOC, MX, NAPTR, NS, PTR, SRV, SSHFP, TLSA, TXT, and URI. ● Split DNS, also known as split-view, split-horizon, split-brain DNS, is not supported. ● There are known issues if the DNS nameserver restarts in a multi-core environment. For example, if log rotation causes a nameserver to restart, the nameserver might crash. If you must use a multi-core setup, allow systemd to restart the nameserver after a failure occurs. 	<p>DNS is not integrated with native IdM tools. For example, IdM does not update the DNS records automatically after a change in the topology.</p>
Works best for:	<p>Basic usage within the IdM deployment.</p> <p>When the IdM server manages DNS, DNS is tightly integrated with native IdM tools, which enables automating some of the DNS record management tasks.</p>	<p>Environments where advanced DNS features beyond the scope of the IdM DNS are needed.</p> <p>Environments with a well-established DNS infrastructure where you want to keep using an external DNS server.</p>

Even if an Identity Management server is used as a primary DNS server, other external DNS servers can still be used as secondary servers. For example, if your environment is already using another DNS server, such as a DNS server integrated with Active Directory (AD), you can delegate only the IdM primary

domain to the DNS integrated with IdM. It is not necessary to migrate DNS zones to the IdM DNS.



NOTE

If you need to issue certificates for IdM clients with an IP address in the Subject Alternative Name (SAN) extension, you must use the IdM integrated DNS service.

4.2. GUIDELINES FOR PLANNING THE DNS DOMAIN NAME AND KERBEROS REALM NAME

When installing the first Identity Management (IdM) server, the installation prompts for a primary DNS name of the IdM domain and Kerberos realm name. These guidelines can help you set the names correctly.



WARNING

You will not be able to change the IdM primary domain name and Kerberos realm name after the server is already installed. Do not expect to be able to move from a testing environment to a production environment by changing the names, for example from **lab.example.com** to **production.example.com**.

A separate DNS domain for service records

Ensure that the *primary DNS domain* used for IdM is not shared with any other system. This helps avoid conflicts on the DNS level.

Proper DNS domain name delegation

Ensure you have valid delegation in the public DNS tree for the DNS domain. Do not use a domain name that is not delegated to you, not even on a private network.

Multi-label DNS domain

Do not use single-label domain names, for example **.company**. The IdM domain must be composed of one or more subdomains and a top level domain, for example **example.com** or **company.example.com**.

A unique Kerberos realm name

Ensure the realm name is not in conflict with any other existing Kerberos realm name, such as a name used by Active Directory (AD).

Kerberos realm name as an upper-case version of the primary DNS name

Consider setting the realm name to an upper-case (**EXAMPLE.COM**) version of the primary DNS domain name (**example.com**).

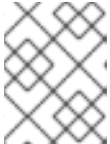


WARNING

If you do not set the Kerberos realm name to be the upper-case version of the primary DNS name, you will not be able to use AD trusts.

Additional notes on planning the DNS domain name and Kerberos realm name

- One IdM deployment always represents one Kerberos realm.
- You can join IdM clients from multiple distinct DNS domains (*example.com*, *example.net*, *example.org*) to a single Kerberos realm (*EXAMPLE.COM*).
- IdM clients do not need to be in the primary DNS domain. For example, if the IdM domain is *idm.example.com*, the clients can be in the *clients.example.com* domain, but clear mapping must be configured between the DNS domain and the Kerberos realm.



NOTE

The standard method to create the mapping is using the `_kerberos` TXT DNS records. The IdM integrated DNS adds these records automatically.

Planning DNS forwarding

- If you want to use only one forwarder for your entire IdM deployment, configure a **global forwarder**.
- If your company is spread over multiple sites in geographically distant regions, global forwarders might be impractical. Configure **per-server forwarders**.
- If your company has an internal DNS network that is not resolvable from the public internet, configure a **forward zone** and **zone forwarders** so that the hosts in the IdM domain can resolve hosts from this other internal DNS network.

CHAPTER 5. PLANNING YOUR CA SERVICES

Identity Management (IdM) in Red Hat Enterprise Linux provides different types of certificate authority (CA) configurations. The following sections describe different scenarios and provide advice to help you determine which configuration is best for your use case.

CA subject DN

The Certificate Authority (CA) subject distinguished name (DN) is the name of the CA. It must be globally unique in the Identity Management (IdM) CA infrastructure and cannot be changed after the installation. In case you need the IdM CA to be externally signed, you might need to consult the administrator of the external CA about the form your IdM CA Subject DN should take.

5.1. CA SERVICES AVAILABLE IN AN IDM SERVER

You can install an Identity Management (IdM) server with an integrated IdM certificate authority (CA) or without a CA.

Table 5.1. Comparing IdM with integrated CA and without a CA

	Integrated CA	Without a CA
Overview:	<p>IdM uses its own public key infrastructure (PKI) service with a <i>CA signing certificate</i> to create and sign the certificates in the IdM domain.</p> <ul style="list-style-type: none"> • If the root CA is the integrated CA, IdM uses a self-signed CA certificate. • If the root CA is an external CA, the integrated IdM CA is subordinate to the external CA. The CA certificate used by IdM is signed by the external CA, but all certificates for the IdM domain are issued by the integrated Certificate System instance. • Integrated CA is also able to issue certificates for users, hosts, or services. <p>The external CA can be a corporate CA or a third-party CA.</p>	<p>IdM does not set up its own CA, but uses signed host certificates from an external CA.</p> <p>Installing a server without a CA requires you to request the following certificates from a third-party authority:</p> <ul style="list-style-type: none"> • An LDAP server certificate • An Apache server certificate • A PKINIT certificate • Full CA certificate chain of the CA that issued the LDAP and Apache server certificates

	Integrated CA	Without a CA
Limitations:	<p>If the integrated CA is subordinate to an external CA, the certificates issued within the IdM domain are potentially subject to restrictions set by the external CA for various certificate attributes, such as:</p> <ul style="list-style-type: none"> • The validity period. • Constraints on what subject names can appear on certificates issued by the IDM CA or its subordinates.. • Constraints on whether the IDM CA can itself, issue subordinate CA certificates, or how "deep" the chain of subordinate certificates can go. 	<p>Managing certificates outside of IdM causes many additional activities, such as :</p> <ul style="list-style-type: none"> • Creating, uploading, and renewing certificates is a manual process. • The certmonger service does not track the IPA certificates (LDAP server, Apache server, and PKINIT certificates) and does not notify you when the certificates are about to expire. The administrators must manually set up notifications for externally issued certificates, or set tracking requests for those certificates if they want certmonger to track them.
Works best for:	Environments that allow you to create and use your own certificate infrastructure.	Very rare cases when restrictions within the infrastructure do not allow you to install certificate services integrated with the server.



NOTE

Switching from the self-signed CA to an externally-signed CA, or the other way around, as well as changing which external CA issues the IdM CA certificate, is possible even after the installation. It is also possible to configure an integrated CA even after an installation without a CA. For more details, see [Installing an IdM server: With integrated DNS, without a CA](#).

Additional resources

- [Understanding the certificates used internally by IdM](#)

5.2. GUIDELINES FOR DISTRIBUTION OF CA SERVICES

The following steps provide guidelines for the distribution of your certificate authority (CA) services.

Procedure

1. Install the CA services on more than one server in the topology.
Replicas configured without a CA forward all certificate operation requests to the CA servers in your topology.

**WARNING**

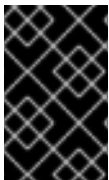
If you lose all servers with a CA, you lose all the CA configuration without any chance of recovery. In this case you must configure a new CA and issue and install new certificates.

2. Maintain a sufficient number of CA servers to handle the CA requests in your deployment.

See the following table for further recommendations on appropriate number of CA servers:

Table 5.2. Guidelines for setting up appropriate number of CA servers

Description of the deployment	Suggested number of CA servers
A deployment with a very large number of certificates issued	Three or four CA servers
A deployment with bandwidth or availability problems between multiple regions	One CA server per region, with a minimum of three servers total for the deployment
All other deployments	Two CA servers

**IMPORTANT**

Four CA servers in the topology are usually enough if the number of concurrent certificate requests is not high. The replication processes between more than four CA servers can increase processor usage and lead to performance degradation.

CHAPTER 6. PLANNING INTEGRATION WITH AD

The following sections introduce the options for integrating Red Hat Enterprise Linux with Active Directory (AD).

6.1. DIRECT INTEGRATION OF LINUX SYSTEMS INTO ACTIVE DIRECTORY

In direct integration, Linux systems are connected directly to Active Directory (AD). The following types of integration are possible:

Integration with the System Security Services Daemon (SSSD)

SSSD can connect a Linux system with various identity and authentication stores: AD, Identity Management (IdM), or a generic LDAP or Kerberos server.

Notable requirements for integration with SSSD:

- When integrating with AD, SSSD works only within a single AD forest by default. For multi-forest setup, configure manual domain enumeration.
- Remote AD forests must trust the local forest to ensure that the **idmap_ad** plug-in handles remote forest users correctly.

SSSD supports both direct and indirect integration. It also enables switching from one integration approach to the other without significant migration costs.

Integration with Samba Winbind

The Winbind component of the Samba suite emulates a Windows client on a Linux system and communicates with AD servers.

Notable requirements for integration with Samba Winbind:

- Direct integration with Winbind in a multi-forest AD setup requires bidirectional trusts.
- A bidirectional path from the local domain of a Linux system must exist to the domain of a user in a remote AD forest to allow full information about the user from the remote AD domain to be available to the **idmap_ad** plug-in.

Recommendations

- SSSD satisfies most of the use cases for AD integration and provides a robust solution as a generic gateway between a client system and different types of identity and authentication providers – AD, IdM, Kerberos, and LDAP.
- Winbind is recommended for deployment on those AD domain member servers on which you plan to deploy Samba FS.

6.2. INDIRECT INTEGRATION OF LINUX SYSTEMS INTO ACTIVE DIRECTORY BY USING IDENTITY MANAGEMENT

In indirect integration, Linux systems are first connected to a central server which is then connected to Active Directory (AD). Indirect integration enables the administrator to manage Linux systems and policies centrally, while users from AD can transparently access Linux systems and services.

Integration based on cross-forest trust with AD

The Identity Management (IdM) server acts as the central server to control Linux systems. A cross-realm Kerberos trust with AD is established, enabling users from AD to log on to access Linux systems and resources. IdM presents itself to AD as a separate forest and takes advantage of the forest-level trusts supported by AD.

When using a trust:

- AD users can access IdM resources.
- IdM servers and clients can resolve the identities of AD users and groups.
- AD users and groups access IdM under the conditions defined by IdM, such as host-based access control.
- AD users and groups continue being managed on the AD side.

Integration based on synchronization

This approach is based on the WinSync tool. A WinSync replication agreement synchronizes user accounts from AD to IdM.



WARNING

WinSync is no longer actively developed in Red Hat Enterprise Linux 8. The preferred solution for indirect integration is cross-forest trust.

The limitations of integration based on synchronization include:

- Groups are not synchronized from IdM to AD.
- Users are duplicated in AD and IdM.
- WinSync supports only a single AD domain.
- Only one domain controller in AD can be used to synchronize data to one instance of IdM.
- User passwords must be synchronized, which requires the PassSync component to be installed on all domain controllers in the AD domain.
- After configuring the synchronization, all AD users must manually change passwords before PassSync can synchronize them.

6.3. GUIDELINES FOR DECIDING BETWEEN DIRECT AND INDIRECT INTEGRATION

These guidelines can help you decide which type of integration fits your use case.

Number of systems to be connected to Active Directory

Connecting less than 30-50 systems (not a hard limit)

If you connect less than 30-50 systems, consider direct integration. Indirect integration might introduce unnecessary overhead.

Connecting more than 30-50 systems (not a hard limit)

If you connect more than 30-50 systems, consider indirect integration with Identity Management. With this approach, you can benefit from the centralized management for Linux systems.

Managing a small number of Linux systems, but expecting the number to grow rapidly

In this scenario, consider indirect integration to avoid having to migrate the environment later.

Frequency of deploying new systems and their type**Deploying bare metal systems on an irregular basis**

If you deploy new systems rarely and they are usually bare metal systems, consider direct integration. In such cases, direct integration is usually simplest and easiest.

Deploying virtual systems frequently

If you deploy new systems often and they are usually virtual systems provisioned on demand, consider indirect integration. With indirect integration, you can use a central server to manage the new systems dynamically and integrate with orchestration tools, such as Red Hat Satellite.

Active Directory is the required authentication provider**Do your internal policies state that all users must authenticate against Active Directory?**

You can choose either direct or indirect integration. If you use indirect integration with a trust between Identity Management and Active Directory, the users that access Linux systems authenticate against Active Directory. Policies that exist in Active Directory are executed and enforced during authentication.

CHAPTER 7. PLANNING A CROSS-FOREST TRUST BETWEEN IDM AND AD

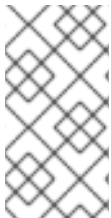
Active Directory (AD) and Identity Management (IdM) are two alternative environments managing a variety of core services, such as Kerberos, LDAP, DNS, and certificate services. A *cross-forest trust* relationship transparently integrates these two diverse environments by enabling all core services to interact seamlessly. The following sections provide advice on how to plan and design a cross-forest trust deployment.

7.1. CROSS-FOREST AND EXTERNAL TRUSTS BETWEEN IDM AND AD

A cross-forest trust between IdM and AD

In a pure Active Directory (AD) environment, a cross-forest trust connects two separate AD forest root domains. When you create a cross-forest trust between AD and IdM, the IdM domain presents itself to AD as a separate forest with a single domain. A trust relationship is then established between the AD forest root domain and the IdM domain. As a result, users from the AD forest can access the resources in the IdM domain.

IdM can establish a trust with one AD forest or multiple unrelated forests.



NOTE

Two separate Kerberos realms can be connected in a *cross-realm trust*. However, a Kerberos realm only concerns authentication, not other services and protocols involved in identity and authorization operations. Therefore, establishing a Kerberos cross-realm trust is not enough to enable users from one realm to access resources in another realm.

An external trust to an AD domain

An external trust is a trust relationship between IdM and an Active Directory domain. While a forest trust always requires establishing a trust between IdM and the root domain of an Active Directory forest, an external trust can be established from IdM to any domain within a forest.

7.2. TRUST CONTROLLERS AND TRUST AGENTS

Identity Management (IdM) provides the following types of IdM servers that support trust to Active Directory (AD):

Trust controllers

IdM servers that can perform identity lookups against AD domain controllers. They also run the Samba suite so they can establish trust with AD. AD domain controllers contact trust controllers when establishing and verifying the trust to AD. AD-enrolled machines communicate with IdM trust controllers for Kerberos authentication requests.

The first trust controller is created when you configure the trust. In environments with multiple domain controllers across different geographic locations, it is beneficial to designate additional RHEL IdM servers as trust controllers in these locations.

Trust controllers run more network-facing services than trust agents, and thus present a greater attack surface for potential intruders.

Trust agents

IdM servers that can resolve identity lookups from RHEL IdM clients against AD domain controllers. Unlike trust controllers, trust agents cannot process Kerberos authentication requests.

In addition to trust agents and controllers, the IdM domain can also include standard IdM servers. However, these servers do not communicate with AD. Therefore, clients that communicate with these standard servers cannot resolve AD users and groups or authenticate and authorize AD users.

By default, IdM servers cannot resolve users and groups from trusted domains. To allow IdM servers to interact with trusted domains, you must explicitly configure them to operate as trust controllers or trust agents.

Table 7.1. Comparing the capabilities supported by trust controllers and trust agents

Capability	Trust agent	Trust controller
Resolve AD users and groups	Yes	Yes
Enroll IdM clients that run services accessible by users from trusted AD forests	Yes	Yes
Add, modify, or remove trust agreements	No	Yes
Assign the trust agent role to an IdM server	No	Yes

When planning the deployment of trust controllers and trust agents, consider these guidelines:

- Configure at least two trust controllers per IdM deployment.
- Configure at least two trust controllers in each data center.

If you ever want to create additional trust controllers or if an existing trust controller fails, create a new trust controller by promoting a trust agent or a standard server. To do this, use the **ipa-adtrust-install** utility on the IdM server.



IMPORTANT

You cannot downgrade an existing trust controller to a trust agent.

Additional resources

- [Creating a trust agent](#)

7.3. ONE-WAY TRUSTS AND TWO-WAY TRUSTS

In one way trusts, Identity Management (IdM) trusts Active Directory (AD) but AD does not trust IdM. AD users can access resources in the IdM domain but users from IdM cannot access resources within the AD domain. The IdM server connects to AD using a special account, and reads identity information that is then delivered to IdM clients over LDAP.

In two way trusts, IdM users can authenticate to AD, and AD users can authenticate to IdM. AD users can authenticate to and access resources in the IdM domain as in the one way trust case. IdM users can authenticate but cannot access most of the resources in AD. They can only access those Kerberized services in AD forests that do not require any access control check.

To be able to grant access to the AD resources, IdM needs to implement the Global Catalog service. This service does not yet exist in the current version of the IdM server. Because of that, a two-way trust between IdM and AD is nearly functionally equivalent to a one-way trust between IdM and AD.

7.4. KERBEROS FAST FOR TRUSTED DOMAINS

Kerberos Flexible Authentication Secure Tunneling (FAST) is also called Kerberos armoring in an Active Directory (AD) environment. Kerberos FAST provides an additional security layer for the Kerberos communication between the clients and the Key Distribution Center (KDC). In IdM, the KDCs are running on the IdM servers and FAST is enabled by default. The Two-Factor Authentication (2FA) in IdM also requires enabling FAST.

In AD, Kerberos armoring is disabled by default on the AD Domain Controllers (DC). You can enable it on the Domain Controller on the **Tools>Group Policy Management>Default Domain Controller Policy**:

- Right-click **Default Domain Controller Policy** and select **edit**. Navigate to **Computer Configuration>Policies>Administrative Templates>System>KDC** and double-click **KDC support for claims, compound authentication, and Kerberos armoring**.

Once you enable KDC support for claims, the policy setting allows the following options:

- **Not supported**
- **Supported**
- **Always provide claims**
- **Fail unarmored authentication requests**

Kerberos FAST is implemented in the Kerberos client libraries on IdM clients. You can configure IdM clients either to use FAST for all trusted domains which advertise FAST or to not use Kerberos FAST at all. If you enable Kerberos armoring in the trusted AD forest the IdM client uses Kerberos FAST by default. FAST establishes a secure tunneling with the help of a cryptographic key. To protect the connection to the domain controllers of a trusted domain, Kerberos FAST must get a cross-realm Ticket Granting Ticket (TGT) from the trusted domain because those keys are valid only inside the Kerberos realm. Kerberos FAST uses the Kerberos hosts keys of the IdM client to request the cross-realm TGT with the help of the IdM servers. That only works when the AD forest trusts the IdM domain which means a two-way trust is required.

If AD policies require the enforcing of Kerberos FAST use, you need to establish a two-way trust between IdM domain and AD forest. You must plan this before the connection is established because both IdM and AD must have records about direction and the type of trust.

If you already established a one-way trust, run the **ipa trust-add ... --two-way=true** command to remove the existing trust agreement and create a two-way trust. This requires use of administrative credentials. As IdM attempts to remove the existing trust agreement from the AD side, it requires administrator permissions for AD access. If you establish the original trust by using a shared secret rather than an AD administrative account, it recreates the trust as a two-way and changes trusted domain objects on the IdM side only. Windows administrators must repeat the same procedure by using Windows UI to choose a bi-directional trust and use the same shared secret to recreate the trust.

If using a two-way trust is not possible, you must disable Kerberos FAST on all IdM clients. The users from the trusted AD forest can authenticate with a password or direct smart card. To disable Kerberos FAST, add the following setting to the **sssd.conf** file in the **[domain]** section:

```
krb5_use_fast = never
```

Note, you do not need to use this option when the authentication is based on ssh-keys, GSSAPI authentication or SSH with smart cards from remote Windows clients. These methods do not use Kerberos FAST because the IdM client does not have to communicate with a DC. Additionally, after disabling FAST on the IdM client, the two-factor authentication IdM feature is also unavailable.

7.5. POSIX AND ID MAPPING ID RANGE TYPES FOR AD USERS

Identity Management (IdM) enforces access control rules based on the POSIX User ID (UID) and Group ID (GID) of a user. In contrast, Active Directory (AD) identifies users by Security Identifiers (SIDs). As an administrator, you can configure AD to store POSIX attributes for your AD users and groups, such as **uidNumber**, **gidNumber**, **unixHomeDirectory**, or **loginShell**.

You configure AD to store POSIX attributes when you are setting up the trust agreement using the **ipa-ad-trust-posix** ID range.

In scenarios when AD does not store POSIX attributes, the System Security Services Daemon (SSSD) can consistently map a unique UID based on a user's SID in a process called **ID mapping**. You can explicitly choose this behavior by selecting the **ipa-ad-trust** ID range during trust creation.

Additional resources

- [Options for automatically mapping private groups for AD users](#)
- [Setting up a trust agreement using the command line](#)

7.6. OPTIONS FOR AUTOMATICALLY MAPPING PRIVATE GROUPS FOR AD USERS: POSIX TRUSTS

Each user in a Linux environment has a primary user group. Red Hat Enterprise Linux (RHEL) uses a user private group (UPG) scheme: a UPG has the same name as the user for which it was created and that user is the only member of the UPG.

If you have allocated UIDs for your AD users, but GIDs were not added, you can configure SSSD to automatically map private groups for users based on their UID by adjusting the `auto_private_groups` setting for that ID range.

auto_private_groups=false

By default, the **auto_private_groups** option is set to **false** for **ipa-ad-trust-posix** ID ranges used in a POSIX trust. With this configuration, SSSD retrieves the **uidNumber** and **gidNumber** from each AD user entry. When **auto_private_groups** is set to **false**, SSSD assigns the **uidNumber** value to the user's UID, the **gidNumber** to the user's GID. A group with that GID must exist in AD, or you will not be able to resolve that user. The following table demonstrates user resolution outcomes based on various AD configurations.

Table 7.2. SSSD behavior when the `auto_private_groups` variable is set to `false` for a POSIX ID range

User configuration in AD	Output of <code>id username</code>
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber is not defined • No group in AD with gidNumber = 4000. 	SSSD cannot resolve the user.
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 4000 • No group in AD with gidNumber = 4000. 	SSSD cannot resolve the user.
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 4000 • AD has a group with gidNumber = 4000. 	<pre># id aduser@AD- DOMAIN.COMuid=4000(aduser@ad- domain.com) gid=4000(adgroup@ad- domain.com) groups=4000(adgroup@ad- domain.com), ...</pre>

If an AD user does not have a primary group configured in AD, or its **gidNumber** does not correspond to an existing group, the IdM server is unable to resolve that user correctly because it cannot look up all the groups the user belongs to. To work around this issue, you can enable automatic private group mapping in SSSD by setting the **auto_private_groups** option to **true** or **hybrid**:

auto_private_groups = true

SSSD always maps a private group with the **gidNumber** set to match the **uidNumber** from the AD user entry.

Table 7.3. SSSD behavior when the **auto_private_groups** variable is set to **true** for a POSIX ID range

User configuration in AD	Output of <code>id username</code>
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber is not defined • AD does not have a group with GID=4000. 	<pre># id aduser@AD- DOMAIN.COMuid=4000(aduser@ad- domain.com) gid=4000(aduser@ad- domain.com) groups=4000(aduser@ad- domain.com), ...</pre>

User configuration in AD	Output of <code>id username</code>
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 5000 • AD does not have a group with gidNumber = 5000. 	<pre># id aduser@AD-DOMAIN.COMuid=4000(aduser@ad-domain.com) gid=4000(aduser@ad-domain.com) groups=4000(aduser@ad-domain.com), ...</pre>
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 4000 • AD does not have a group with gidNumber = 4000. 	<pre># id aduser@AD-DOMAIN.COMuid=4000(aduser@ad-domain.com) gid=4000(aduser@ad-domain.com) groups=4000(aduser@ad-domain.com), ...</pre>
AD user entry has: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 5000 • AD has a group with gidNumber = 5000. 	<pre># id aduser@AD-DOMAIN.COMuid=4000(aduser@ad-domain.com) gid=4000(aduser@ad-domain.com) groups=4000(aduser@ad-domain.com), ...</pre>

auto_private_groups = hybrid

If the **uidNumber** value matches **gidNumber**, but there is no group with this **gidNumber**, SSSD maps a private group as the user's primary user group with a **gidNumber** that matches the **uidNumber**. If the **uidNumber** and **gidNumber** values differ, and there is a group with this **gidNumber**, SSSD uses the value from **gidNumber**.

Table 7.4. SSSD behavior when the `auto_private_groups` variable is set to `hybrid` for a POSIX ID range

User configuration in AD	Output of <code>id username</code>
AD user entry with: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber is not defined • AD does not have a group with gidNumber = 4000. 	SSSD cannot resolve the user.

User configuration in AD	Output of <code>id</code> username
AD user entry with: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 5000 • AD does not have a group with gidNumber = 5000. 	SSSD cannot resolve the user.
AD user entry with: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 4000 • AD does not have a group with gidNumber = 4000. 	# id aduser@AD-DOMAIN.COM uid=4000(aduser@ad-domain.com) gid=4000(aduser@ad-domain.com) groups=4000(aduser@ad-domain.com), ...
AD user entry with: <ul style="list-style-type: none"> • uidNumber = 4000 • gidNumber = 5000 • AD has a group with gidNumber = 5000. 	# id aduser@AD-DOMAIN.COM uid=4000(aduser@ad-domain.com) gid=5000(aduser@ad-domain.com) groups=5000(adgroup@ad-domain.com), ...

Additional resources

- [Enabling automatic private group mapping for a POSIX ID range on the CLI](#)
- [Enabling automatic private group mapping for a POSIX ID range in the IdM WebUI](#)

7.7. OPTIONS FOR AUTOMATICALLY MAPPING PRIVATE GROUPS FOR AD USERS: ID MAPPING TRUSTS

Each user in a Linux environment has a primary user group. Red Hat Enterprise Linux (RHEL) uses a user private group (UPG) scheme: a UPG has the same name as the user for which it was created and that user is the only member of the UPG.

If you have allocated UIDs for your AD users, but GIDs were not added, you can configure SSSD to automatically map private groups for users based on their UID by adjusting the **auto_private_groups** setting for that ID range.

By default, the **auto_private_groups** option is set to **true** for **ipa-ad-trust** ID ranges used in an ID mapping trust. With this configuration, SSSD computes the UID and GID for an AD user based on its Security Identifier (SID). SSSD ignores any POSIX attributes in AD, such as **uidNumber**, **gidNumber**, and also ignores the **primaryGroupID**.

auto_private_groups = true

SSSD always maps a private group with the GID set to match the UID, which is based on the SID of the AD user.

Table 7.5. SSSD behavior when the `auto_private_groups` variable is set to `true` for an ID mapping ID range

User configuration in AD	Output of <code>id username</code>
AD user entry where: <ul style="list-style-type: none"> SID maps to 7000 primaryGroupID maps to 8000 	<pre># id aduser@AD-DOMAIN.COM uid=7000(aduser@ad-domain.com) gid=7000(aduser@ad-domain.com) groups=7000(aduser@ad-domain.com), 8000(adgroup@ad-domain.com), ...</pre>

`auto_private_groups = false`

If you set the **`auto_private_groups`** option to **`false`**, SSSD uses the **`primaryGroupID`** set in the AD entry as the GID number. The default value for **`primaryGroupID`** corresponds to the **Domain Users** group in AD.

Table 7.6. SSSD behavior when the `auto_private_groups` variable is set to `false` for an ID mapping ID range

User configuration in AD	Output of <code>id username</code>
AD user entry where: <ul style="list-style-type: none"> SID maps to 7000 primaryGroupID maps to 8000 	<pre># id aduser@AD-DOMAIN.COM uid=7000(aduser@ad-domain.com) gid=8000(adgroup@ad-domain.com) groups=8000(adgroup@ad-domain.com), ...</pre>

7.8. NON-POSIX EXTERNAL GROUPS AND SID MAPPING

Identity Management (IdM) uses LDAP for managing groups. Active Directory (AD) entries are not synchronized or copied over to IdM, which means that AD users and groups have no LDAP objects in the LDAP server, so they cannot be directly used to express group membership in the IdM LDAP. For this reason, administrators in IdM need to create non-POSIX external groups, referenced as normal IdM LDAP objects to signify group membership for AD users and groups in IdM.

Security IDs (SIDs) for non-POSIX external groups are processed by SSSD, which maps the SIDs of groups in Active Directory to POSIX groups in IdM. In Active Directory, SIDs are associated with user names. When an AD user name is used to access IdM resources, SSSD uses the user's SID to build up a full group membership information for the user in the IdM domain.

7.9. GUIDELINES FOR SETTING UP DNS FOR AN IDM-AD TRUST

These guidelines can help you configure DNS to establish a cross-forest trust between Identity Management (IdM) and Active Directory (AD).

Unique primary DNS domains

Ensure both AD and IdM have their own unique primary DNS domains configured. For example:

- ***ad.example.com*** for AD and ***idm.example.com*** for IdM
- ***example.com*** for AD and ***idm.example.com*** for IdM

The most convenient management solution is an environment where each DNS domain is managed by integrated DNS servers, but you can also use any other standard-compliant DNS server.

IdM and AD DNS Domains

Systems joined to IdM can be distributed over multiple DNS domains. Red Hat recommends that you deploy IdM clients in a DNS zone different to the ones owned by Active Directory. The primary IdM DNS domain must have proper SRV records to support AD trusts.



NOTE

In some environments with trusts between IdM and Active Directory, you can install an IdM client on a host that is part of the Active Directory DNS domain. The host can then benefit from the Linux-focused features of IdM. This is not a recommended configuration and has some limitations. See [Configuring IdM clients in an Active Directory DNS domain](#) for more details.

Proper SRV records

Ensure the primary IdM DNS domain has proper SRV records to support AD trusts.

For other DNS domains that are part of the same IdM realm, the SRV records do not have to be configured when the trust to AD is established. The reason is that AD domain controllers do not use SRV records to discover Kerberos key distribution centers (KDCs) but rather base the KDC discovery on name suffix routing information for the trust.

DNS records resolvable from all DNS domains in the trust

Ensure all machines can resolve DNS records from all DNS domains involved in the trust relationship:

- When configuring the IdM DNS, follow the instructions described in [Installing an IdM server with an external CA](#).
- If you are using IdM without integrated DNS, follow the instructions described in [Installing an IdM server without integrated DNS](#).

Kerberos realm names as upper-case versions of primary DNS domain names

Ensure Kerberos realm names are the same as the primary DNS domain names, with all letters uppercase. For example, if the domain names are ***ad.example.com*** for AD and ***idm.example.com*** for IdM, the Kerberos realm names must be ***AD.EXAMPLE.COM*** and ***IDM.EXAMPLE.COM***.

7.10. GUIDELINES FOR CONFIGURING NETBIOS NAMES

The NetBIOS name is usually the far-left component of the domain name. For example:

- In the domain name ***linux.example.com***, the NetBIOS name is ***linux***.
- In the domain name ***example.com***, the NetBIOS name is ***example***.

Different NetBIOS names for the Identity Management (IdM) and Active Directory (AD) domains

Ensure the IdM and AD domains have different NetBIOS names.

The NetBIOS name is critical for identifying the AD domain. If the IdM domain is within a subdomain of the AD DNS, the NetBIOS name is also critical for identifying the IdM domain and services.

Character limit for NetBIOS names

The maximum length of a NetBIOS name is 15 characters.

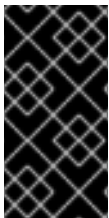
7.11. SUPPORTED VERSIONS OF WINDOWS SERVER

You can establish a trust relationship with Active Directory (AD) forests that use the following forest and domain functional levels:

- Forest functional level range: Windows Server 2012 – Windows Server 2016
- Domain functional level range: Windows Server 2012 – Windows Server 2016

Identity Management (IdM) supports establishing a trust with Active Directory domain controllers running the following operating systems:

- Windows Server 2022 (RHEL 8.7 and later)
- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012



IMPORTANT

In RHEL 8.4, Identity Management (IdM) does not support establishing trust to Active Directory with Active Directory domain controllers running Windows Server 2008 R2 or earlier versions. RHEL IdM now requires SMB encryption when establishing the trust relationship, which is only supported in Windows Server 2012 or later.

7.12. AD SERVER DISCOVERY AND AFFINITY

Server discovery and affinity configuration affects which Active Directory (AD) servers an Identity Management (IdM) client communicates with in a cross-forest trust between IdM and AD.

Configuring clients to prefer servers in the same geographical location helps prevent time lags and other problems that occur when clients contact servers from another, remote data center. To verify clients communicate with local servers, you must ensure that:

- Clients communicate with local IdM servers over LDAP and over Kerberos
- Clients communicate with local AD servers over Kerberos
- Embedded clients on IdM servers communicate with local AD servers over LDAP and over Kerberos

Options for configuring LDAP and Kerberos on the IdM client for communication with local IdM servers

When using IdM with integrated DNS

By default, clients use automatic service lookup based on the DNS records. In this setup, you can also use the *DNS locations* feature to configure DNS-based service discovery.

To override the automatic lookup, you can disable the DNS discovery in one of the following ways:

- During the IdM client installation by providing failover parameters from the command line
- After the client installation by modifying the System Security Services Daemon (SSSD) configuration

When using IdM without integrated DNS

You must explicitly configure clients in one of the following ways:

- During the IdM client installation by providing failover parameters from the command line
- After the client installation by modifying the SSSD configuration

Options for configuring Kerberos on the IdM client for communication with local AD servers

IdM clients are unable to automatically discover which AD servers to communicate with. To specify the AD servers manually, modify the **krb5.conf** file:

- Add the AD realm information
- Explicitly list the AD servers to communicate with

For example:

```
[realms]
AD.EXAMPLE.COM = {
  kdc = server1.ad.example.com
  kdc = server2.ad.example.com
}
```

Options for configuring embedded clients on IdM servers for communication with local AD servers over Kerberos and LDAP

The embedded client on an IdM server works also as a client of the AD server. It can automatically discover and use the appropriate AD site.

When the embedded client performs the discovery, it might first discover an AD server in a remote location. If the attempt to contact the remote server takes too long, the client might stop the operation without establishing the connection. Use the **dns_resolver_timeout** option in the **sssd.conf** file on the client to increase the amount of time for which the client waits for a reply from the DNS resolver. See the *sssd.conf(5)* man page for details.

Once the embedded client has been configured to communicate with the local AD servers, the SSSD remembers the AD site the embedded client belongs to. Thanks to this, SSSD normally sends an LDAP ping directly to a local domain controller to refresh its site information. If the site no longer exists or the client has meanwhile been assigned to a different site, SSSD starts querying for SRV records in the forest and goes through a whole process of autodiscovery.

Using *trusted domain sections* in **sssd.conf**, you can also explicitly override some of the information that is discovered automatically by default.

Additional resources

- [Working with DNS in Identity Management](#)

7.13. OPERATIONS PERFORMED DURING INDIRECT INTEGRATION OF IDM TO AD

The following table lists operations and requests performed by an Identity Management IdM trust controller towards Active Directory AD domain controllers during the creation of an IdM to AD trust.

Table 7.7. Operations performed from an IdM trust controller towards AD domain controllers

Operation	Protocol used	Purpose
DNS resolution against the AD DNS resolvers configured on an IdM trust controller	DNS	To discover the IP addresses of AD domain controllers
Requests to UDP/UDP6 port 389 on an AD DC	Connectionless LDAP (CLDAP)	To perform AD DC discovery
Requests to TCP/TCP6 ports 389 and 3268 on an AD DC	LDAP	To query AD user and group information
Requests to TCP/TCP6 ports 389 and 3268 on an AD DC	DCE RPC and SMB	To set up and support cross-forest trust to AD
Requests to TCP/TCP6 ports 135, 139, 445 on an AD DC	DCE RPC and SMB	To set up and support cross-forest trust to AD
Requests to dynamically opened ports on an AD DC as directed by the Active Directory domain controller, likely in the range of 49152-65535 (TCP/TCP6)	DCE RPC and SMB	To respond to requests by DCE RPC End-point mapper (port 135 TCP/TCP6)
Requests to ports 88 (TCP/TCP6 and UDP/UDP6), 464 (TCP/TCP6 and UDP/UDP6), and 749 (TCP/TCP6) on an AD DC	Kerberos	To obtain a Kerberos ticket; change a Kerberos password; administer Kerberos remotely

The following table lists operations and requests performed by AD domain controllers towards IdM trust controllers during the creation of an IdM to AD trust.

Table 7.8. Operations performed from an AD domain controller towards IdM trust controllers

Operation	Protocol used	Purpose
DNS resolution against the IdM DNS resolvers configured on an AD domain controller	DNS	To discover the IP addresses of IdM trust controllers

Operation	Protocol used	Purpose
Requests to UDP/UDP6 port 389 on an IdM trust controller	CLDAP	To perform IdM trust controller discovery
Requests to TCP/TCP6 ports 135, 139, 445 on an IdM trust controller	DCE RPC and SMB	To verify the cross-forest trust to AD
Requests to dynamically opened ports on an IdM trust controller as directed by the IdM trust controller, likely in the range of 49152-65535 (TCP/TCP6)	DCE RPC and SMB	To respond to requests by DCE RPC End-point mapper (port 135 TCP/TCP6)
Requests to ports 88 (TCP/TCP6 and UDP/UDP6), 464 (TCP/TCP6 and UDP/UDP6), and 749 (TCP/TCP6) on an IdM trust controller	Kerberos	To obtain a Kerberos ticket; change a Kerberos password; administer Kerberos remotely

CHAPTER 8. BACKING UP AND RESTORING IDM

Identity Management lets you manually back up and restore the IdM system after a data loss event.

During a backup, the system creates a directory that stores information about your IdM setup. You can use this backup directory to restore your original IdM setup.



NOTE

The IdM backup and restore features are designed to help prevent data loss. To mitigate the impact of server loss and ensure continued operation, provide alternative servers to clients. For information on establishing a replication topology see [Preparing for server loss with replication](#).

8.1. IDM BACKUP TYPES

With the **ipa-backup** utility, you can create two types of backups:

Full-server backup

- **Contains** all server configuration files related to IdM, and LDAP data in LDAP Data Interchange Format (LDIF) files
- IdM services must be **offline**.
- **Suitable for** rebuilding an IdM deployment from scratch.

Data-only backup

- **Contains** LDAP data in LDIF files and the replication changelog
- IdM services can be **online or offline**.
- **Suitable for** restoring IdM data to a state in the past

8.2. NAMING CONVENTIONS FOR IDM BACKUP FILES

By default, IdM stores backups as **.tar** archives in subdirectories of the **/var/lib/ipa/backup/** directory.

The archives and subdirectories follow these naming conventions:

Full-server backup

An archive named **ipa-full.tar** in a directory named **ipa-full-*<YEAR-MM-DD-HH-MM-SS>***, with the time specified in GMT time.

```
[root@server ~]# ll /var/lib/ipa/backup/ipa-full-2021-01-29-12-11-46
total 3056
-rw-r--r--. 1 root root   158 Jan 29 12:11 header
-rw-r--r--. 1 root root 3121511 Jan 29 12:11 ipa-full.tar
```

Data-only backup

An archive named **ipa-data.tar** in a directory named **ipa-data-*<YEAR-MM-DD-HH-MM-SS>***, with the time specified in GMT time.

```
[root@server ~]# ll /var/lib/ipa/backup/ipa-data-2021-01-29-12-14-23
total 1072
-rw-r--r--. 1 root root    158 Jan 29 12:14 header
-rw-r--r--. 1 root root 1090388 Jan 29 12:14 ipa-data.tar
```



NOTE

Uninstalling an IdM server does not automatically remove any backup files.

8.3. CONSIDERATIONS WHEN CREATING A BACKUP

The important behaviors and limitations of the **ipa-backup** command include the following:

- By default, the **ipa-backup** utility runs in offline mode, which stops all IdM services. The utility automatically restarts IdM services after the backup is finished.
- A full-server backup must **always** run with IdM services offline, but a data-only backup can be performed with services online.
- By default, the **ipa-backup** utility creates backups on the file system containing the **/var/lib/ipa/backup/** directory. Red Hat recommends creating backups regularly on a file system separate from the production filesystem used by IdM, and archiving the backups to a fixed medium, such as tape or optical storage.
- Consider performing backups on [hidden replicas](#). IdM services can be shut down on hidden replicas without affecting IdM clients.
- Starting with RHEL 8.3.0, the **ipa-backup** utility checks if all of the services used in your IdM cluster, such as a Certificate Authority (CA), Domain Name System (DNS), and Key Recovery Agent (KRA), are installed on the server where you are running the backup. If the server does not have all these services installed, the **ipa-backup** utility exits with a warning, because backups taken on that host would not be sufficient for a full cluster restoration.
For example, if your IdM deployment uses an integrated Certificate Authority (CA), a backup run on a non-CA replica will not capture CA data. Red Hat recommends verifying that the replica where you perform an **ipa-backup** has all of the IdM services used in the cluster installed.

You can bypass the IdM server role check with the **ipa-backup --disable-role-check** command, but the resulting backup will not contain all the data necessary to restore IdM fully.

8.4. CREATING AN IDM BACKUP

Create a full-server and data-only backup in offline and online modes using the **ipa-backup** command.

Prerequisites

- You must have **root** privileges to run the **ipa-backup** utility.

Procedure

- To create a full-server backup in offline mode, use the **ipa-backup** utility without additional options.

```
[root@server ~]# ipa-backup
```

```

Preparing backup on server.example.com
Stopping IPA services
Backing up ipaca in EXAMPLE-COM to LDIF
Backing up userRoot in EXAMPLE-COM to LDIF
Backing up EXAMPLE-COM
Backing up files
Starting IPA service
Backed up to /var/lib/ipa/backup/ipa-full-2020-01-14-11-26-06
The ipa-backup command was successful

```

- To create an offline data-only backup, specify the **--data** option.

```
[root@server ~]# ipa-backup --data
```

- To create a full-server backup that includes IdM log files, use the **--logs** option.

```
[root@server ~]# ipa-backup --logs
```

- To create a data-only backup while IdM services are running, specify both **--data** and **--online** options.

```
[root@server ~]# ipa-backup --data --online
```



NOTE

If the backup fails due to insufficient space in the **/tmp** directory, use the **TMPDIR** environment variable to change the destination for temporary files created by the backup process:

```
[root@server ~]# TMPDIR=/new/location ipa-backup
```

Verification

- Ensure the backup directory contains an archive with the backup.

```
[root@server ~]# ls /var/lib/ipa/backup/ipa-full-2020-01-14-11-26-06
header ipa-full.tar
```

Additional resources

- [ipa-backup command fails to finish](#) (Red Hat Knowledgebase)

8.5. CREATING A GPG2-ENCRYPTED IDM BACKUP

You can create encrypted backups using GNU Privacy Guard (GPG) encryption. The following procedure creates an IdM backup and encrypts it using a GPG2 key.

Prerequisites

- You have created a GPG2 key. See [Creating a GPG2 key](#).

Procedure

- Create a GPG-encrypted backup by specifying the **--gpg** option.

```
[root@server ~]# ipa-backup --gpg
Preparing backup on server.example.com
Stopping IPA services
Backing up ipaca in EXAMPLE-COM to LDIF
Backing up userRoot in EXAMPLE-COM to LDIF
Backing up EXAMPLE-COM
Backing up files
Starting IPA service
Encrypting /var/lib/ipa/backup/ipa-full-2020-01-13-14-38-00/ipa-full.tar
Backed up to /var/lib/ipa/backup/ipa-full-2020-01-13-14-38-00
The ipa-backup command was successful
```

Verification

- Ensure that the backup directory contains an encrypted archive with a **.gpg** file extension.

```
[root@server ~]# ls /var/lib/ipa/backup/ipa-full-2020-01-13-14-38-00
header ipa-full.tar.gpg
```

Additional resources

- [Creating a backup](#).

8.6. CREATING A GPG2 KEY

The following procedure describes how to generate a GPG2 key to use with encryption utilities.

Prerequisites

- You need **root** privileges.

Procedure

1. Install and configure the **pinentry** utility.

```
[root@server ~]# yum install pinentry
[root@server ~]# mkdir ~/.gnupg -m 700
[root@server ~]# echo "pinentry-program /usr/bin/pinentry-curses" >> ~/.gnupg/gpg-agent.conf
```

2. Create a **key-input** file used for generating a GPG keypair with your preferred details. For example:

```
[root@server ~]# cat >key-input <<EOF
%echo Generating a standard key
Key-Type: RSA
Key-Length: 2048
Name-Real: GPG User
Name-Comment: first key
```

```
Name-Email: root@example.com
Expire-Date: 0
%commit
%echo Finished creating standard key
EOF
```

- Optional: By default, GPG2 stores its keyring in the `~/.gnupg` file. To use a custom keyring location, set the **GNUPGHOME** environment variable to a directory that is only accessible by root.

```
[root@server ~]# export GNUPGHOME=/root/backup

[root@server ~]# mkdir -p $GNUPGHOME -m 700
```

- Generate a new GPG2 key based on the contents of the **key-input** file.

```
[root@server ~]# gpg2 --batch --gen-key key-input
```

- Enter a passphrase to protect the GPG2 key. You use this passphrase to access the private key for decryption.

```
Please enter the passphrase to
protect your new key

Passphrase: <passphrase>

<OK>          <Cancel>
```

- Confirm the correct passphrase by entering it again.

```
Please re-enter this passphrase

Passphrase: <passphrase>

<OK>          <Cancel>
```

- Verify that the new GPG2 key was created successfully.

```
gpg: keybox '/root/backup/pubring.kbx' created
gpg: Generating a standard key
gpg: /root/backup/trustdb.gpg: trustdb created
gpg: key BF28FFA302EF4557 marked as ultimately trusted
gpg: directory '/root/backup/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/backup/openpgp-
revocs.d/8F6FCF10C80359D5A05AED67BF28FFA302EF4557.rev'
gpg: Finished creating standard key
```

Verification

- List the GPG keys on the server.

```
[root@server ~]# gpg2 --list-secret-keys
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
/root/backup/pubring.kbx
-----
sec  rsa2048 2020-01-13 [SCEA]
      8F6FCF10C80359D5A05AED67BF28FFA302EF4557
uid      [ultimate] GPG User (first key) <root@example.com>
```

Additional resources

- [GNU Privacy Guard](#)

8.7. WHEN TO RESTORE FROM AN IDM BACKUP

You can respond to several disaster scenarios by restoring from an IdM backup:

- **Undesirable changes were made to the LDAP content** Entries were modified or deleted, replication carried out those changes throughout the deployment, and you want to revert those changes. Restoring a data-only backup returns the LDAP entries to the previous state without affecting the IdM configuration itself.
- **Total Infrastructure Loss, or loss of all CA instances** If a disaster damages all Certificate Authority replicas, the deployment has lost the ability to rebuild itself by deploying additional servers. In this situation, restore a backup of a CA Replica and build new replicas from it.
- **An upgrade on an isolated server failed** The operating system remains functional, but the IdM data is corrupted, which is why you want to restore the IdM system to a known good state. Red Hat recommends working with Technical Support to diagnose and troubleshoot the issue. If those efforts fail, restore from a full-server backup.



IMPORTANT

The preferred solution for hardware or upgrade failure is to rebuild the lost server from a replica. For more information, see [Recovering a single server with replication](#).

8.8. CONSIDERATIONS WHEN RESTORING FROM AN IDM BACKUP

If you have a backup created with the **ipa-backup** utility, you can restore your IdM server or the LDAP content to the state they were in when the backup was performed.

The following are the key considerations while restoring from an IdM backup:

- You can only restore a backup on a server that matches the configuration of the server where the backup was originally created. The server **must** have:
 - The same hostname
 - The same IP address
 - The same version of IdM software

- If one IdM server among many is restored, the restored server becomes the only source of information for IdM. All other servers **must** be re-initialized from the restored server.
- Since any data created after the last backup will be lost, do not use the backup and restore solution for normal system maintenance.
- If a server is lost, Red Hat recommends rebuilding the server by reinstalling it as a replica, instead of restoring from a backup. Creating a new replica preserves data from the current working environment. For more information, see [Preparing for server loss with replication](#).
- The backup and restore features can only be managed from the command line and are not available in the IdM web UI.
- You cannot restore from backup files located in the **/tmp** or **/var/tmp** directories. The IdM Directory Server uses a **PrivateTmp** directory and cannot access the **/tmp** or **/var/tmp** directories commonly available to the operating system.

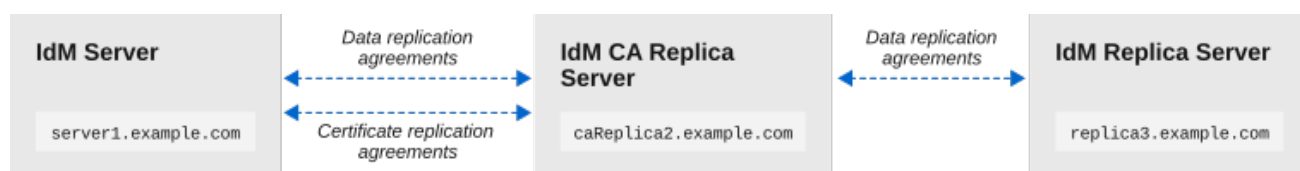
TIP

Restoring from a backup requires the same software (RPM) versions on the target host as were installed when the backup was performed. Due to this, Red Hat recommends restoring from a Virtual Machine snapshot rather than a backup. For more information, see [Recovering from data loss with VM snapshots](#).

8.9. RESTORING AN IDM SERVER FROM A BACKUP

Restore an IdM server, or its LDAP data, from an IdM backup.

Figure 8.1. Replication topology used in this example



137_FHML_001

Table 8.1. Server naming conventions used in this example

Server host name	Function
server1.example.com	The server that needs to be restored from backup.
caReplica2.example.com	A Certificate Authority (CA) replica connected to the server1.example.com host.
replica3.example.com	A replica connected to the caReplica2.example.com host.

Prerequisites

- You have generated a full-server or data-only backup of the IdM server with the **ipa-backup** utility. See [Creating a backup](#).

- Your backup files are not in the **/tmp** or **/var/tmp** directories.
- Before performing a full-server restore from a full-server backup, **uninstall** IdM from the server and **reinstall** IdM using the same server configuration as before.

Procedure

1. Use the **ipa-restore** utility to restore a full-server or data-only backup.

- If the backup directory is in the default **/var/lib/ipa/backup/** location, enter only the name of the directory:

```
[root@server1 ~]# ipa-restore ipa-full-2020-01-14-12-02-32
```

- If the backup directory is not in the default location, enter its full path:

```
[root@server1 ~]# ipa-restore /mybackups/ipa-data-2020-02-01-05-30-00
```



NOTE

The **ipa-restore** utility automatically detects the type of backup that the directory contains, and performs the same type of restore by default. To perform a data-only restore from a full-server backup, add the **--data** option to the **ipa-restore** command:

```
[root@server1 ~]# ipa-restore --data ipa-full-2020-01-14-12-02-32
```

2. Enter the Directory Manager password.

```
Directory Manager (existing master) password:
```

3. Enter **yes** to confirm overwriting current data with the backup.

```
Preparing restore from /var/lib/ipa/backup/ipa-full-2020-01-14-12-02-32 on
server1.example.com
Performing FULL restore from FULL backup
Temporary setting umask to 022
Restoring data will overwrite existing live data. Continue to restore? [no]: yes
```

4. The **ipa-restore** utility disables replication on all servers that are available:

```
Each master will individually need to be re-initialized or
re-created from this one. The replication agreements on
masters running IPA 3.1 or earlier will need to be manually
re-enabled. See the man page for details.
Disabling all replication.
Disabling replication agreement on server1.example.com to caReplica2.example.com
Disabling CA replication agreement on server1.example.com to caReplica2.example.com
Disabling replication agreement on caReplica2.example.com to server1.example.com
Disabling replication agreement on caReplica2.example.com to replica3.example.com
Disabling CA replication agreement on caReplica2.example.com to server1.example.com
Disabling replication agreement on replica3.example.com to caReplica2.example.com
```

The utility then stops IdM services, restores the backup, and restarts the services:

```
Stopping IPA services
Systemwide CA database updated.
Restoring files
Systemwide CA database updated.
Restoring from userRoot in EXAMPLE-COM
Restoring from ipaca in EXAMPLE-COM
Restarting GSS-proxy
Starting IPA services
Restarting SSSD
Restarting oddjobd
Restoring umask to 18
The ipa-restore command was successful
```

5. Re-initialize all replicas connected to the restored server:
 - a. List all replication topology segments for the **domain** suffix, taking note of topology segments involving the restored server.

```
[root@server1 ~]# ipa topologysegment-find domain
-----
2 segments matched
-----
Segment name: server1.example.com-to-caReplica2.example.com
Left node: server1.example.com
Right node: caReplica2.example.com
Connectivity: both

Segment name: caReplica2.example.com-to-replica3.example.com
Left node: caReplica2.example.com
Right node: replica3.example.com
Connectivity: both
-----
Number of entries returned 2
-----
```

- b. Re-initialize the **domain** suffix for all topology segments with the restored server. In this example, perform a re-initialization of **caReplica2** with data from **server1**.

```
[root@caReplica2 ~]# ipa-replica-manage re-initialize --from=server1.example.com
Update in progress, 2 seconds elapsed
Update succeeded
```

- c. Moving on to Certificate Authority data, list all replication topology segments for the **ca** suffix.

```
[root@server1 ~]# ipa topologysegment-find ca
-----
1 segment matched
-----
Segment name: server1.example.com-to-caReplica2.example.com
Left node: server1.example.com
Right node: caReplica2.example.com
Connectivity: both
```

```
-----
Number of entries returned 1
-----
```

- d. Re-initialize all CA replicas connected to the restored server.
In this example, perform a **csreplica** re-initialization of **caReplica2** with data from **server1**.

```
[root@caReplica2 ~]# ipa-csreplica-manage re-initialize --
from=server1.example.com
Directory Manager password:

Update in progress, 3 seconds elapsed
Update succeeded
```

6. Continue moving outward through the replication topology, re-initializing successive replicas, until all servers have been updated with the data from restored server **server1.example.com**. In this example, we only have to re-initialize the **domain** suffix on **replica3** with the data from **caReplica2**:

```
[root@replica3 ~]# ipa-replica-manage re-initialize --from=caReplica2.example.com
Directory Manager password:

Update in progress, 3 seconds elapsed
Update succeeded
```

7. Clear SSSD's cache on every server to avoid authentication problems due to invalid data:
 - a. Stop the SSSD service:

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

- b. Remove all cached content from SSSD:

```
[root@server ~]# sss_cache -E
```

- c. Start the SSSD service:

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

- d. Reboot the server.

Additional resources

- The **ipa-restore (1)** man page also covers in detail how to handle complex replication scenarios during restoration.

8.10. RESTORING FROM AN ENCRYPTED BACKUP

This procedure restores an IdM server from an encrypted IdM backup. The **ipa-restore** utility automatically detects if an IdM backup is encrypted and restores it using the GPG2 root keyring.

Prerequisites

- A GPG-encrypted IdM backup. See [Creating encrypted IdM backups](#).
- The LDAP Directory Manager password
- The passphrase used when creating the GPG key

Procedure

1. If you used a custom keyring location when creating the GPG2 keys, verify that the **\$GNUPGHOME** environment variable is set to that directory. See [Creating a GPG2 key](#).

```
[root@server ~]# echo $GNUPGHOME
/root/backup
```

2. Provide the **ipa-restore** utility with the backup directory location.

```
[root@server ~]# ipa-restore ipa-full-2020-01-13-18-30-54
```

- a. Enter the Directory Manager password.

```
Directory Manager (existing master) password:
```

- b. Enter the passphrase you used when creating the GPG key.

```
Please enter the passphrase to unlock the OpenPGP secret key:
"GPG User (first key) <root@example.com>"
2048-bit RSA key, ID BF28FFA302EF4557,
created 2020-01-13.

Passphrase: <passphrase>

<OK> <Cancel>
```

3. Re-initialize all replicas connected to the restored server. See [Restoring an IdM server from backup](#).

CHAPTER 9. BACKING UP AND RESTORING IDM SERVERS USING ANSIBLE PLAYBOOKS

Using the **ipabackup** Ansible role, you can automate backing up an IdM server, transferring backup files between servers and your Ansible controller, and restoring an IdM server from a backup.

9.1. USING ANSIBLE TO CREATE A BACKUP OF AN IDM SERVER

You can use the **ipabackup** role in an Ansible playbook to create a backup of an IdM server and store it on the IdM server.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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 `ipadmin_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. Navigate to the `~/MyPlaybooks/` directory:

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

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

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/backup-server.yml backup-my-server.yml
```

3. Open the **backup-my-server.yml** Ansible playbook file for editing.
4. Adapt the file by setting the **hosts** variable to a host group from your inventory file. In this example, set it to the **ipaserver** host group:

```
---
- name: Playbook to backup IPA server
  hosts: ipaserver
  become: true

  roles:
  - role: ipabackup
    state: present
```

5. Save the file.

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

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory
backup-my-server.yml
```

Verification

1. Log into the IdM server that you have backed up.
2. Verify that the backup is in the **/var/lib/ipa/backup** directory.

```
[root@server ~]# ls /var/lib/ipa/backup/
ipa-full-2021-04-30-13-12-00
```

Additional resources

- For more sample Ansible playbooks that use the **ipabackup** role, see:
 - The **README.md** file in the **/usr/share/doc/ansible-freeipa/roles/ipabackup** directory.
 - The **/usr/share/doc/ansible-freeipa/playbooks/** directory.

9.2. USING ANSIBLE TO CREATE A BACKUP OF AN IDM SERVER ON YOUR ANSIBLE CONTROLLER

You can use the **ipabackup** role in an Ansible playbook to create a backup of an IdM server and automatically transfer it on your Ansible controller. Your backup file name begins with the host name of the IdM server.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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. To store the backups, create a subdirectory in your home directory on the Ansible controller.

```
$ mkdir ~/ipabackups
```

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

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

3. Make a copy of the **backup-server-to-controller.yml** file located in the **/usr/share/doc/ansible-freeipa/playbooks** directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/backup-server-to-controller.yml backup-my-server-to-my-controller.yml
```

4. Open the **backup-my-server-to-my-controller.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Set the **hosts** variable to a host group from your inventory file. In this example, set it to the **ipaserver** host group.
 - b. Optional: To maintain a copy of the backup on the IdM server, uncomment the following line:

```
# ipabackup_keep_on_server: true
```

6. By default, backups are stored in the present working directory of the Ansible controller. To specify the backup directory you created in Step 1, add the **ipabackup_controller_path** variable and set it to the **/home/user/ipabackups** directory.

```
---
- name: Playbook to backup IPA server to controller
  hosts: ipaserver
  become: true
  vars:
    ipabackup_to_controller: true
    # ipabackup_keep_on_server: true
    ipabackup_controller_path: /home/user/ipabackups

  roles:
    - role: ipabackup
      state: present
```

7. Save the file.
8. Run the Ansible playbook, specifying the inventory file and the playbook file:

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory backup-my-server-to-my-controller.yml
```

Verification

- Verify that the backup is in the **/home/user/ipabackups** directory of your Ansible controller:

```
[user@controller ~]$ ls /home/user/ipabackups
server.idm.example.com_ipa-full-2021-04-30-13-12-00
```

Additional resources

- For more sample Ansible playbooks that use the **ipabackup** role, see:

- The **README.md** file in the `/usr/share/doc/ansible-freeipa/roles/ipabackup` directory.
- The `/usr/share/doc/ansible-freeipa/playbooks/` directory.

9.3. USING ANSIBLE TO COPY A BACKUP OF AN IDM SERVER TO YOUR ANSIBLE CONTROLLER

You can use an Ansible playbook to copy a backup of an IdM server from the IdM server to your Ansible controller.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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 **ipadmin_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. To store the backups, create a subdirectory in your home directory on the Ansible controller.

```
$ mkdir ~/ipabackups
```

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

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

3. Make a copy of the **copy-backup-from-server.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/copy-backup-from-server.yml copy-backup-from-my-server-to-my-controller.yml
```

4. Open the **copy-my-backup-from-my-server-to-my-controller.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Set the **hosts** variable to a host group from your inventory file. In this example, set it to the **ipaserver** host group.
 - b. Set the **ipabackup_name** variable to the name of the **ipabackup** on your IdM server to copy to your Ansible controller.

- c. By default, backups are stored in the present working directory of the Ansible controller. To specify the directory you created in Step 1, add the **ipabackup_controller_path** variable and set it to the **/home/user/ipabackups** directory.

```
---
- name: Playbook to copy backup from IPA server
  hosts: ipaserver
  become: true
  vars:
    ipabackup_name: ipa-full-2021-04-30-13-12-00
    ipabackup_to_controller: true
    ipabackup_controller_path: /home/user/ipabackups

  roles:
    - role: ipabackup
      state: present
```

6. Save the file.

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

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory copy-
backup-from-my-server-to-my-controller.yml
```

NOTE

To copy **all** IdM backups to your controller, set the **ipabackup_name** variable in the Ansible playbook to **all**:

```
vars:
  ipabackup_name: all
  ipabackup_to_controller: true
```

For an example, see the **copy-all-backups-from-server.yml** Ansible playbook in the **/usr/share/doc/ansible-freeipa/playbooks** directory.

Verification

- Verify your backup is in the **/home/user/ipabackups** directory on your Ansible controller:

```
[user@controller ~]$ ls /home/user/ipabackups
server.idm.example.com_ipa-full-2021-04-30-13-12-00
```

Additional resources

- The **README.md** file in the **/usr/share/doc/ansible-freeipa/roles/ipabackup** directory.
- The **/usr/share/doc/ansible-freeipa/playbooks/** directory.

9.4. USING ANSIBLE TO COPY A BACKUP OF AN IDM SERVER FROM YOUR ANSIBLE CONTROLLER TO THE IDM SERVER

You can use an Ansible playbook to copy a backup of an IdM server from your Ansible controller to the IdM server.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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. Navigate to the `~/MyPlaybooks/` directory:

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

2. Make a copy of the `copy-backup-from-controller.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/copy-backup-from-controller.yml copy-backup-from-my-controller-to-my-server.yml
```

3. Open the `copy-my-backup-from-my-controller-to-my-server.yml` file for editing.
4. Adapt the file by setting the following variables:
 - a. Set the `hosts` variable to a host group from your inventory file. In this example, set it to the `ipaserver` host group.
 - b. Set the `ipabackup_name` variable to the name of the `ipabackup` on your Ansible controller to copy to the IdM server.

```
---
- name: Playbook to copy a backup from controller to the IPA server
  hosts: ipaserver
  become: true

  vars:
    ipabackup_name: server.idm.example.com_ipa-full-2021-04-30-13-12-00
    ipabackup_from_controller: true

  roles:
    - role: ipabackup
      state: copied
```

5. Save the file.

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

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory copy-backup-from-my-controller-to-my-server.yml
```

Additional resources

- The **README.md** file in the `/usr/share/doc/ansible-freeipa/roles/ipabackup` directory.
- The `/usr/share/doc/ansible-freeipa/playbooks/` directory.

9.5. USING ANSIBLE TO REMOVE A BACKUP FROM AN IDM SERVER

You can use an Ansible playbook to remove a backup from an IdM server.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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 `ipadmin_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. Navigate to the `~/MyPlaybooks/` directory:

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

2. Make a copy of the **remove-backup-from-server.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/remove-backup-from-server.yml remove-backup-from-my-server.yml
```

3. Open the **remove-backup-from-my-server.yml** file for editing.
4. Adapt the file by setting the following variables:
 - a. Set the **hosts** variable to a host group from your inventory file. In this example, set it to the **ipaserver** host group.
 - b. Set the **ipabackup_name** variable to the name of the **ipabackup** to remove from your IdM server.

```
---
```

```

- name: Playbook to remove backup from IPA server
  hosts: ipaserver
  become: true

  vars:
    ipabackup_name: ipa-full-2021-04-30-13-12-00

  roles:
    - role: ipabackup
      state: absent

```

5. Save the file.

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

```

$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory
remove-backup-from-my-server.yml

```

NOTE

To remove **all** IdM backups from the IdM server, set the **ipabackup_name** variable in the Ansible playbook to **all**:

```

vars:
  ipabackup_name: all

```

For an example, see the **remove-all-backups-from-server.yml** Ansible playbook in the **/usr/share/doc/ansible-freeipa/playbooks** directory.

Additional resources

- The **README.md** file in the **/usr/share/doc/ansible-freeipa/roles/ipabackup** directory.
- The **/usr/share/doc/ansible-freeipa/playbooks/** directory.

9.6. USING ANSIBLE TO RESTORE AN IDM SERVER FROM A BACKUP STORED ON THE SERVER

You can use an Ansible playbook to restore an IdM server from a backup stored on that host.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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 **ipadmin_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.
- You know the LDAP Directory Manager password.

Procedure

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

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

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

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/restore-server.yml restore-my-server.yml
```

3. Open the **restore-my-server.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables:
 - a. Set the **hosts** variable to a host group from your inventory file. In this example, set it to the **ipaserver** host group.
 - b. Set the **ipabackup_name** variable to the name of the **ipabackup** to restore.
 - c. Set the **ipabackup_password** variable to the LDAP Directory Manager password.

```
---
- name: Playbook to restore an IPA server
  hosts: ipaserver
  become: true

  vars:
    ipabackup_name: ipa-full-2021-04-30-13-12-00
    ipabackup_password: <your_LDAP_DM_password>

  roles:
    - role: ipabackup
      state: restored
```

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

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory
restore-my-server.yml
```

Additional resources

- The **README.md** file in the **/usr/share/doc/ansible-freeipa/roles/ipabackup** directory.
- The **/usr/share/doc/ansible-freeipa/playbooks/** directory.

9.7. USING ANSIBLE TO RESTORE AN IDM SERVER FROM A BACKUP STORED ON YOUR ANSIBLE CONTROLLER

You can use an Ansible playbook to restore an IdM server from a backup stored on your Ansible controller.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.13 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.
- You know the LDAP Directory Manager password.

Procedure

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

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

2. Make a copy of the `restore-server-from-controller.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/restore-server-from-controller.yml restore-my-server-from-my-controller.yml
```

3. Open the `restore-my-server-from-my-controller.yml` file for editing.
4. Adapt the file by setting the following variables:
 - a. Set the `hosts` variable to a host group from your inventory file. In this example, set it to the `ipaserver` host group.
 - b. Set the `ipabackup_name` variable to the name of the `ipabackup` to restore.
 - c. Set the `ipabackup_password` variable to the LDAP Directory Manager password.

```
---
- name: Playbook to restore IPA server from controller
  hosts: ipaserver
  become: true

  vars:
    ipabackup_name: server.idm.example.com_ipa-full-2021-04-30-13-12-00
    ipabackup_password: <your_LDAP_DM_password>
```

```
ipabackup_from_controller: true

roles:
- role: ipabackup
state: restored
```

5. Save the file.

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

```
$ ansible-playbook --vault-password-file=password_file -v -i ~/MyPlaybooks/inventory
restore-my-server-from-my-controller.yml
```

Additional resources

- The **README.md** file in the **/usr/share/doc/ansible-freeipa/roles/ipabackup** directory.
- The **/usr/share/doc/ansible-freeipa/playbooks/** directory.

CHAPTER 10. 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 11. IDM DIRECTORY SERVER RFC SUPPORT

The Directory Server component in Identity Management (IdM) supports many LDAP-related Requests for Comments (RFCs).

Additional resources

- [Directory Server RFC Support](#)
- [Directory Server 11 Deployment Guide](#)