Red Hat Enterprise Linux 8

Planning Identity Management

Documentation for planning Identity Management and setting up access control

Last Updated: 2020-04-28

Red Hat Enterprise Linux 8 Planning Identity Management

Documentation for planning Identity Management and setting up access control

Legal Notice

Copyright © 2020 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

This document describes the planning of Identity Management services on Red Hat Enterprise Linux 8. The current version of the document contains only selected preview user stories.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. OVERVIEW OF PLANNING FOR IDM AND ACCESS CONTROL IN RHEL

The following sections provide an overview of the options for identity management (IdM) and access control in Red Hat Enterprise Linux. After reading these sections, you will be able to approach the planning stage for your environment.

1.1. INTRODUCTION TO IDM

This module explains the purpose of Identity Management (IdM) in Red Hat Enterprise Linux. It also provides basic information about the IdM domain, including the client and server machines that are part of the domain.

The goal of IdM in Red Hat Enterprise Linux

IdM in Red Hat Enterprise Linux provides a centralized and unified way to manage identity stores, authentication, policies, and authorization policies in a Linux-based domain. 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. Windows client cannot be a member of the IdM domain but user 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, thus 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.
- Knowledge Base article about Standard protocols.
- Red Hat Enterprise Linux 8 Beta Release Notes

1.2. INTRODUCTION TO IDM SERVERS AND CLIENTS

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

IdM servers

IdM servers are Red Hat Enterprise Linux systems that respond to identity, authentication, and authorization requests within an IdM domain. In most deployments, an integrated certificate authority (CA) is also installed with the IdM server.
IdM servers are the central repositories for identity and policy information. IdM servers can also host any of the optional services used by domain members:

- Certificate authority (CA)
- Key Recovery Authority (KRA)
- DNS
- Active Directory (AD) trust controller
- Active Directory (AD) trust agent

The first server installed to create the domain is the *IdM master* or *master server*. The IdM master is not to be confused with the *master CA* server: they can run on two different machines.

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 infromation, use DNS to detect where the servers and services are located and how to connect to them.

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. IdM servers provide different services for the client. Not all the servers need to provide all the possible services. Some server components like Kerberos and LDAP are always available on every server. Other services like CA, DNS, Trust Controller or Vault are optional. This means that different servers in general play different roles in the deployment.

If your IdM topology contains an integrated CA, one server also has the role of the Certificate revocation list (CRL) generation master and the CA renewal master. This server is the *master CA*.

WARNING

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

For redundancy and load balancing, administrators create additional servers by creating a *replica* of any existing server, either the master server or another replica. 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 role of the CRL generation master. Therefore, the term *server* and *replica* are used interchangeably here depending on the context.

1.3. 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 and without IdM

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.

1.4. IDM TERMINOLOGY

IdM master and replicas

The first server installed using the **ipa-server-install** command to create the domain is known as the **master server** or **IdM master**. A **replica** is any server that is installed by the system administrator running the **ipa-replica-install** command. A replication agreement exists by default between a replica and the IdM server from which it was created, enabling it to both receive and send updates to the rest of IdM.
There is no functional difference between a master and a replica. Both are fully functional IdM servers.

Alternative names: **master**, **master server**, **IdM master server**

Master CA server

If your IdM topology contains an integrated certificate authority (CA), one server has the role of the Certificate revocation list (CRL) generation master and the CA renewal master. This server is the **master CA server**. In a deployment without integrated CA, there is no master CA server.
Alternative names: **master CA**

IMPORTANT

IdM master and **master CA server** are two different terms. For example, the first server is the IdM master and the replica is the master CA server in the following deployment scenario:

1. You install the first IdM server in your environment without integrated CA.
2. You install a replica.
3. You install a CA on the replica.

In this scenario, the first server is the IdM master and the replica is the master CA server.

IdM CA server

An IdM server on which the **ipa** certificate authority service is installed and running.
Alternative names: **CA server**

Certificate authorities (CA) in IdM

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

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

In IdM, you can 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**. The certificate of a sub-CA cannot be self-signed.

IdM topology

A term that refers to the structure of your IdM solution , more specifically how replication agreements are set up between and within individual data centers and clusters.

IdM deployment

A term that refers to the type of your IdM solution. 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?

1.5. ADDITIONAL RESOURCES

- For general information on Red Hat IdM, see the Red Hat Identity Management product page on the Red Hat Customer Portal.

CHAPTER 2. PLANNING THE REPLICA TOPOLOGY

The following sections provide advice on determining the appropriate replica topology for your use case.

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

Continuous functionality and high availability of Identity Management (IdM) services is vital for users who access resources. One of the built-in solutions for accomplishing continuous functionality and high availability of the IdM infrastructure through load balancing is the replication of the central directory by creating replica servers of the master server.

IdM allows placing 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.

Maintaining multiple redundant IdM servers and letting them replicate with each other is also a common backup mechanism to mitigate or prevent server loss. For example, if one server fails, the other servers keep providing services to the domain. You can also recover the lost server by creating a new replica based on one of the remaining servers.

2.2. INTRODUCTION TO IDM SERVERS AND CLIENTS

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

IdM servers

IdM servers are Red Hat Enterprise Linux systems that respond to identity, authentication, and authorization requests within an IdM domain. In most deployments, an integrated certificate authority (CA) is also installed with the IdM server.
IdM servers are the central repositories for identity and policy information. IdM servers can also host any of the optional services used by domain members:

- Certificate authority (CA)
- Key Recovery Authority (KRA)
- DNS
- Active Directory (AD) trust controller
- Active Directory (AD) trust agent

The first server installed to create the domain is the *IdM master* or *master server*. The IdM master is not to be confused with the *master CA* server: they can run on two different machines.

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 infromation, use DNS to detect where the servers and services are located and how to connect to them.

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

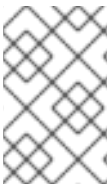
To provide services for large numbers of clients, as well as for redundancy and availability, IdM allows deployment on multiple IdM servers in a single domain. It is possible to deploy up to 60 servers. This is the maximum number of IdM servers, also called replicas, that is currently supported in the IdM domain. IdM servers provide different services for the client. Not all the servers need to provide all the possible services. Some server components like Kerberos and LDAP are always available on every server. Other services like CA, DNS, Trust Controller or Vault are optional. This means that different servers in general play different roles in the deployment.

If your IdM topology contains an integrated CA, one server also has the role of the Certificate revocation list (CRL) generation master and the CA renewal master. This server is the *master CA*.

WARNING

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

For redundancy and load balancing, administrators create additional servers by creating a *replica* of any existing server, either the master server or another replica. 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 role of the CRL generation master. Therefore, the term *server* and *replica* are used interchangeably here depending on the context.

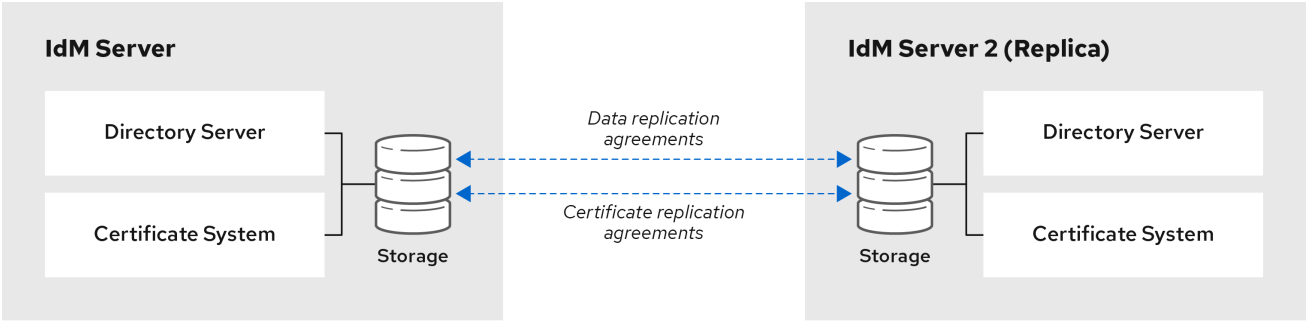
2.3. REPLICATION AGREEMENTS

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

Replication agreements are always bilateral: the data is replicated from one server to the other as well as from the other server to the first server.

IdM uses *multi-master replication*. In multi-master replication, all replicas joined in a replication agreement receive updates, and are therefore considered data masters.

Figure 2.1. Server and replica agreements



64_RHEL_0120

IdM uses two types of replication agreements:

Domain replication agreements

These agreements replicate the identity information.

Certificate replication agreements

These agreements replicate the certificate information.

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

2.4. DETERMINING THE APPROPRIATE NUMBER OF REPLICAS

Set up at least two replicas in each data center (not a hard requirement)

A data center can be, for example, a main office or a geographical location.

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 more frequent queries, plan for more servers.

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

Red Hat guarantees to support environments with 60 replicas or fewer.

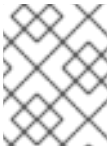
2.5. CONNECTING THE REPLICAS IN A TOPOLOGY

Connect each replica to at least two other replicas

Configuring additional replication agreements ensures that information is replicated not just between the initial replica and the master server, 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.

2.6. REPLICA TOPOLOGY EXAMPLES

The figures below show examples of Identity Management (IdM) topologies based on the guidelines for creating a reliable topology.

Figure 2.2, “Replica Topology Example 1” shows four data centers, each with four servers. The servers are connected with replication agreements.

Figure 2.2. Replica Topology Example 1

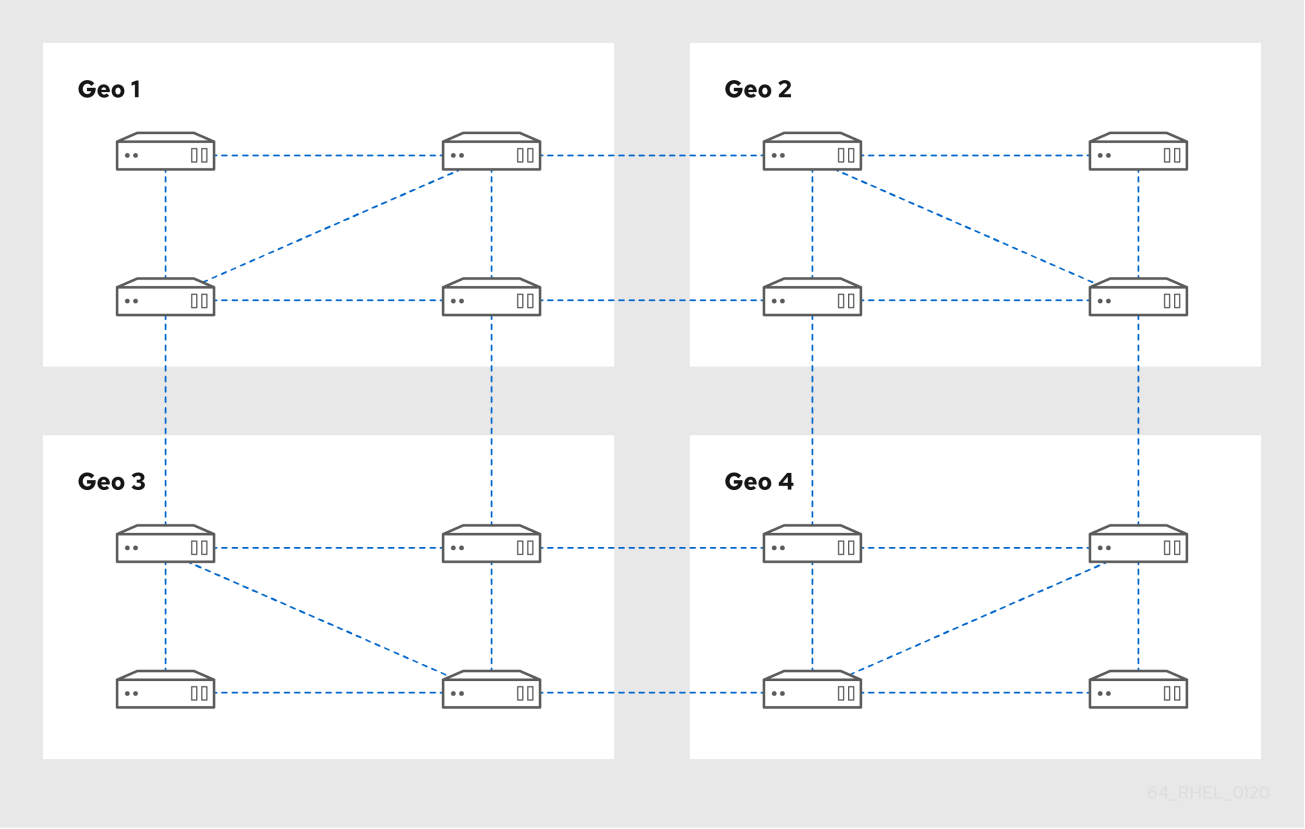
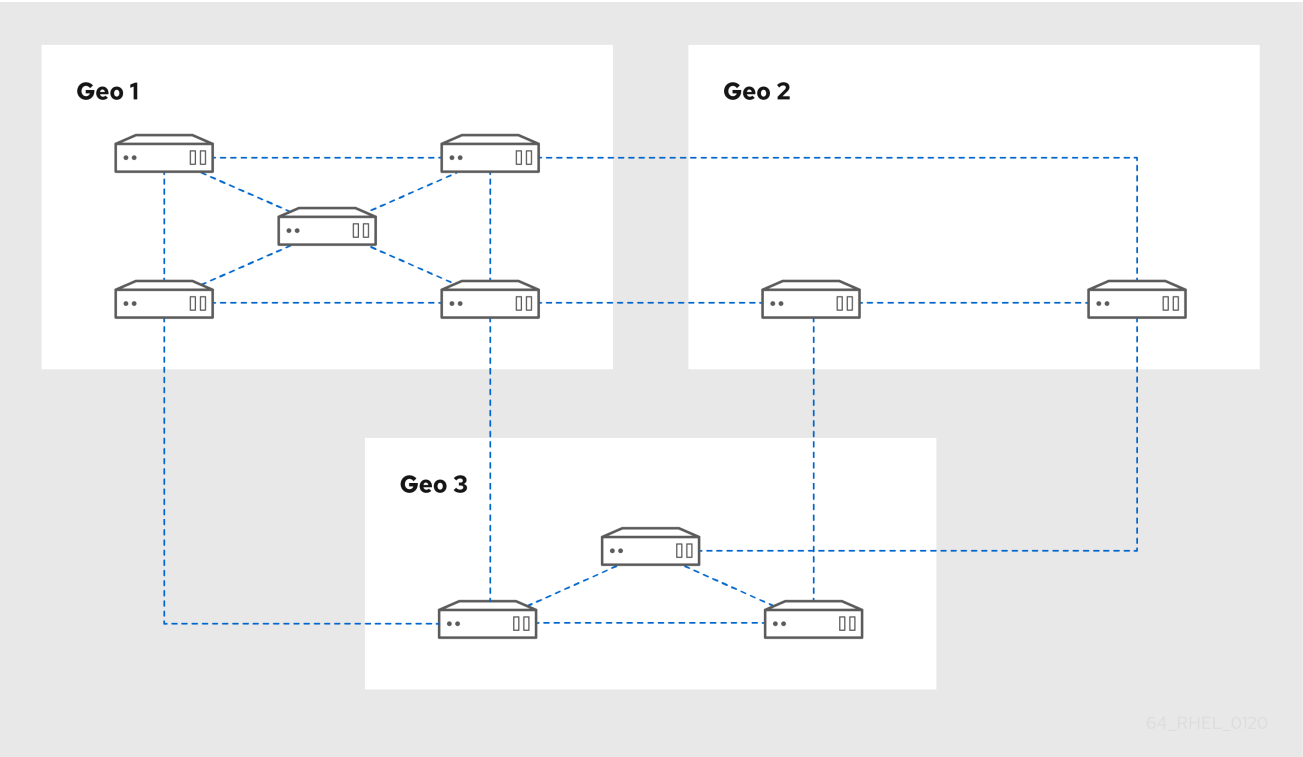


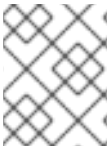
Figure 2.3, “Replica Topology Example 2” shows three data centers, each with a different number of servers. The servers are connected with replication agreements.

Figure 2.3. Replica Topology Example 2



2.7. THE HIDDEN REPLICA MODE

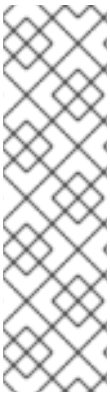
By default, when you set up a new replica, the installer automatically creates service (SRV) resource records in DNS. These records enable clients to auto-discover the replica and its services. A hidden replica is an IdM server that has all services running and available. However, it has no SRV records in DNS, and LDAP server roles are not enabled. Therefore, clients cannot use service discovery to detect these hidden replicas.



NOTE

The hidden replica feature is available in Red Hat Enterprise Linux 8.1 and later as a Technology Preview and, therefore, not supported.

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



NOTE

- Restoring a backup from a hidden replica on a new host always results in a non-hidden (regular) replica.
- All server roles used in a cluster, especially the Certificate Authority role if the integrated CA is used, must be installed on the hidden replica for the backup to be able to restore those services.
- For more information on creating and working with IdM backups, see Backing Up and Restoring IdM.

Other use cases include high-load operations on the IdM API or the LDAP server, such as a mass import or extensive queries. To install a replica as hidden, pass the **--hidden-replica** parameter to the **ipa-replica-install** command.

For further details about installing a replica, see Installing an Identity Management replica .

Alternatively, you can change the state of an existing replica. For details, see Demotion and Promotion of Hidden Replicas.

CHAPTER 3. 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.

3.1. DNS SERVICES AVAILABLE IN AN IDM SERVER

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

Table 3.1. 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:	The integrated DNS server provided by IdM only supports features related to IdM deployment and maintenance. It does not support some of the advanced DNS features. It is not designed to be used as a general-purpose DNS server.	DNS is not integrated with native IdM tools. For example, IdM does not update the DNS records automatically after a change in the topology.
Works best for:	Basic usage within the IdM deployment. When the IdM server manages DNS, DNS is tightly integrated with native IdM tools, which enables automating some of the DNS record management tasks.	Environments where advanced DNS features beyond the scope of the IdM DNS are needed. Environments with a well-established DNS infrastructure where you want to keep using an external DNS server.

Even if an Identity Management server is used as a master DNS server, other external DNS servers can still be used as slave 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.

3.2. 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. The guidelines in this section 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.

CHAPTER 4. 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.

4.1. 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 4.1. Comparing IdM with integrated CA and without a CA

	Integrated CA	Without a CA
Overview:	IdM uses its own public key infrastructure (PKI) service with a *CA signing certificate* to create and sign the certificates in the IdM domain. ● 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. The external CA can be a corporate CA or a third-party CA.	IdM does not set up its own CA, but uses signed host certificates from an external CA. Installing a server without a CA requires you to request the following certificates from a third-party authority: ● 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:	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: • 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.	Managing certificates outside of IdM causes a lot of additional activities, such as : • 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.

4.2. 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.

4.3. GUIDELINES FOR DISTRIBUTION OF CA SERVICES

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

- Install the CA services on more than one server in the topology

Replicas configured without a CA forward all certificate operations requests to the CA servers in your topology.

WARNING

If you lose all servers with a CA, you will lose all the CA configuration without any chance of recovery. In such case you need to set up new CA and issue and install new certificates.

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

For recommendation see the following table:

Table 4.2. 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

CHAPTER 5. PLANNING INTEGRATION WITH AD

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

- For an overview of direct integration, see Section 5.1, “Direct integration”.
- For an overview of indirect integration, see Section 5.2, “Indirect integration”.
- For advice on how to decide between them, see Section 5.3, “Deciding between indirect and direct integration”.

5.1. DIRECT INTEGRATION

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.

5.2. INDIRECT INTEGRATION

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.

5.3. DECIDING BETWEEN INDIRECT AND DIRECT INTEGRATION

The guidelines in this section can help 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.

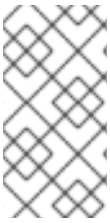
CHAPTER 6. PLANNING A CROSS-FOREST TRUST BETWEEN IDM AND AD

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.

6.1. CROSS-FOREST TRUSTS 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.

6.2. TRUST CONTROLLERS AND TRUST AGENTS

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

Trust agents

IdM servers that can perform identity lookups against AD domain controllers.

Trust controllers

Trust agents that also run the Samba suite. AD domain controllers contact trust controllers when establishing and verifying the trust to AD.
The first trust controller is created when you configure the trust.

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

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 the standard servers cannot resolve AD users and groups or authenticate and authorize AD users.

Table 6.1. 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
Manage the trust (for example, add trust agreements)	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.

6.3. 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.

6.4. 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.

6.5. SETTING UP DNS

These guidelines can help you achieve the right DNS configuration for establishing 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.

No overlap between IdM and AD DNS Domains

Systems joined to IdM can be distributed over multiple DNS domains. Ensure the DNS domains that contain IdM clients do not overlap with DNS domains that contain systems joined to AD.

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***.

6.6. 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.

6.7. 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 2008 – Windows Server 2016
- Domain functional level range: Windows Server 2008 – Windows Server 2016

Identity Management (IdM) supports the following operating systems:

- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows Server 2016
- Windows Server 2019

6.8. CONFIGURING AD SERVER DISCOVERY AND AFFINITY

Server discovery and affinity configuration affects which Active Directory (AD) servers an Identity Management (IdM) client communicates with. This section provides an overview of how discovery and affinity work in an environment with 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 make sure 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.

6.9. OPERATIONS PERFORMED DURING INDIRECT INTEGRATION OF IDM TO AD

Table 6.2, “Operations performed from an IdM trust controller towards AD domain controllers” shows which operations and requests are performed during the creation of an Identity Management (IdM) to Active Directory (AD) trust from the IdM trust controller towards AD domain controllers.

Table 6.2. 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

Table 6.3, "Operations performed from an AD domain controller towards IdM trust controllers" shows which operations and requests are performed during the creation of an IdM to AD trust from the AD domain controller towards IdM trust controllers.

Table 6.3. Operations performed from an AD domain controller towards IdM trust controllers

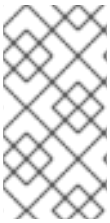
Operation	Protocol used	Purpose
DNS resolution against the IdM DNS resolvers configured on an AD domain controller	DNS	To discover the IP addresses of IdM trust controllers
Requests to UDP/UDP6 port 389 on an IdM trust controller	CLDAP	To perform IdM trust controller discovery
Requests to TCP/TCP6 ports 135, 139, 445 on an IdM trust controller	DCE RPC and SMB	To verify the cross-forest trust to AD

Operation	Protocol used	Purpose
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 7. BACKING UP AND RESTORING IDM

Red Hat Enterprise Linux Identity Management provides a solution to manually back up and restore the IdM system. This may be necessary after a data loss event.

During backup, the system creates a directory containing information on your IdM setup and stores it. During restore, you can use this backup directory to bring your original IdM setup back.



NOTE

The IdM backup and restore features are designed to help prevent data loss. To mitigate the impact of losing a server, and ensure continued operation by providing alternative servers to clients, ensure you have a replica topology according to Mitigating server loss with replication.

7.1. IDM BACKUP TYPES

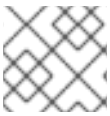
IdM provides two types of backups: a full-server backup, and a data-only backup.

Backup type	Backup contents	Performed Online or Offline	Suitable for
Full-server backup	• All server configuration files related to IdM • LDAP data in LDAP Data Interchange Format (LDIF)	Offline only. IdM services must be temporarily stopped.	Rebuilding an IdM deployment from scratch
Data-only backup	• LDAP data in LDAP Data Interchange Format (LDIF) • Replication Changelog	Online or Offline.	Restoring IdM data to a state in the past

7.2. BACKUP FILE CONVENTIONS

By default, IdM stores backups in the **/var/lib/ipa/backup/** directory, and the naming conventions for these subdirectories are:

- Full-server backup: **ipa-full-YEAR-MM-DD-HH-MM-SS** in GMT time
- Data-only backup: **ipa-data-YEAR-MM-DD-HH-MM-SS** in GMT time

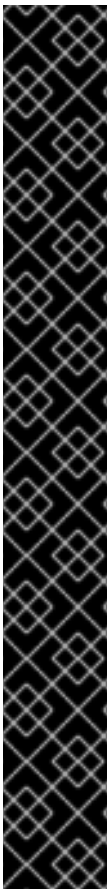


NOTE

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

7.3. CREATING A BACKUP

This section describes how to create a full-server and data-only backup in offline and online modes using the **ipa-backup** command.



IMPORTANT

- By default, **ipa-backup** runs in offline mode, which will stop all IdM services. The services will start automatically after the backup is finished.
- A full-server backup must **always** run with IdM services offline, but a data-only backup may be performed with services online.
- By default, backups are created on the file system containing the **/var/lib/ipa/backup/** directory. We recommend creating backups regularly on a file system separate from the production filesystem used by IdM, and archiving the backups to a fixed medium (tape or optical storage, for example).
- Consider performing backups on hidden replicas. IdM services can be shut down on hidden replicas without affecting IdM clients.
- An IdM backup of a server only captures the server roles installed on that server. For example, if your IdM deployment uses the integrated Certificate Authority (CA), a backup of a non-CA replica will **not** capture CA data. Similarly, a backup of a replica that does **not** have the KRA installed will **not** capture KRA data.
- If the IdM deployment uses the built-in CA, a backup from a CA-less replica will not be enough to rebuild the IdM deployment. Please make sure to create backups on a replica with all of the in-use IdM server roles installed: CA, KRA, DNS.

Examples of using the ipa-backup command

- 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
```

For more details, see ipa-backup Command Fails to Finish .

Verification Steps

- 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
```

7.4. CREATING ENCRYPTED IDM BACKUPS

You can create encrypted backups using GNU Privacy Guard (GPG) encryption. To create encrypted IdM backups, you will first need to create a GPG2 key.

7.4.1. Creating a GPG2 key for encrypting IdM backups

The following procedure describes how to generate a GPG2 key for the **ipa-backup** utility.

Procedure

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

```
[root@server ~]# dnf 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: IPA Backup
Name-Comment: IPA Backup
Name-Email: root@example.com
Expire-Date: 0
```

```
%commit
%echo Finished creating standard key
EOF
```

3. 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
```

4. Begin generating a new GPG2 key based on the contents of **key-input**.

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

a. Enter a passphrase to protect the GPG2 key.

```
┌──────────────────────────────────────────────────────┐
│ Please enter the passphrase to                       │
│ protect your new key                                 │
│                                                      │
│ Passphrase: SecretPassphrase42                       │
│                                                      │
│  <OK>                          <Cancel>              │
└──────────────────────────────────────────────────────┘
```

b. Confirm the correct passphrase by entering it again.

```
┌──────────────────────────────────────────────────────┐
│ Please re-enter this passphrase                      │
│                                                      │
│ Passphrase: SecretPassphrase42                       │
│                                                      │
│  <OK>                          <Cancel>              │
└──────────────────────────────────────────────────────┘
```

c. The new GPG2 key is now created.

```
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 Steps

- 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] IPA Backup (IPA Backup) <root@example.com>
```

Additional resources

- For more information on GPG encryption and its uses, see the GNU Privacy Guard website.

7.4.2. Creating a GPG2-encrypted IdM backup

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 for encrypting IdM backups .

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 Steps

- 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

- For general information on creating a backup, see Creating a backup .

7.5. 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 in order 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 from server loss with replication.

7.6. 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 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 in a multi-master environment is restored, the restored server becomes the only source of information for IdM. All other master 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.

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 .

7.7. RESTORING AN IDM SERVER FROM A BACKUP

The following procedure describes restoring an IdM server, or its LDAP data, from an IdM backup.

Figure 7.1. Replication Topology used in this example

Table 7.1. Server naming conventions used in this example

Server Name	Function
master1.example.com	The server that needs to be restored from backup
caReplica2.example.com	A Certificate Authority (CA) replica connected to master1.example.com.
replica3.example.com	A replica connected to caReplica2.example.com.

Prerequisites

- A full-server or data-only backup of the IdM server was generated with the **ipa-backup** utility. See Creating a backup .
- 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@master1 ~]# 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@master1 ~]# 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 **ipa-restore**:

```
[root@master1 ~]# 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
master1.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 master1.example.com to caReplica2.example.com
Disabling CA replication agreement on master1.example.com to caReplica2.example.com
Disabling replication agreement on caReplica2.example.com to master1.example.com
Disabling replication agreement on caReplica2.example.com to replica3.example.com
Disabling CA replication agreement on caReplica2.example.com to master1.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@master1 ~]# ipa topologysegment-find domain
-----------------
2 segments matched
-----------------
  Segment name: master1.example.com-to-caReplica2.example.com
  Left node: master1.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 **master1**.

```
[root@caReplica2 ~]# ipa-replica-manage re-initialize --from=master1.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@master1 ~]# ipa topologysegment-find ca
-----------------
1 segment matched
-----------------
  Segment name: master1.example.com-to-caReplica2.example.com
  Left node: master1.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 **master1**.

```
[root@caReplica2 ~]# ipa-csreplica-manage re-initialize --
from=master1.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 **master1.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 in order 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.

7.8. RESTORING FROM AN ENCRYPTED BACKUP

The **ipa-restore** utility automatically detects if an IdM backup is encrypted, and restores it using the GPG2 root keyring and **gpg-agent** by default.

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, make sure that the **$GNUPGHOME** environment variable is set to that directory. See Creating a GPG2 key for encrypting IdM backups.

   ```
   [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:        │
│ "IPA Backup (IPA Backup) <root@example.com>"                         │
│ 2048-bit RSA key, ID BF28FFA302EF4557,                               │
│ created 2020-01-13.                                                  │
│                                                                      │
│                                                                      │
│ Passphrase: SecretPassPhrase42                                       │
│                                                                      │
│       <OK>                          <Cancel>                         │
└──────────────────────────────────────────────────────────────────────┘
```

3. Re-initialize all replicas connected to the restored server. See Restoring an IdM server from backup.