Red Hat Enterprise Linux 7

Windows Integration Guide

Integrating Linux systems with Active Directory environments

Last Updated: 2023-02-25

Red Hat Enterprise Linux 7 Windows Integration Guide

Integrating Linux systems with Active Directory environments

Florian Delehaye
Red Hat Customer Content Services
fdelehay@redhat.com

Marc Muehlfeld
Red Hat Customer Content Services

Filip Hanzelka
Red Hat Customer Content Services

Lucie Maňásková
Red Hat Customer Content Services

Aneta Šteflová Petrová
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

Legal Notice

Abstract

Heterogeneous IT environments often contain various different domains and operating systems that need to be able to seamlessly communicate. Red Hat Enterprise Linux offers multiple ways to tightly integrate Linux domains with Active Directory (AD) on Microsoft Windows. The integration is possible on different domain objects that include users, groups, services, or systems. This guide also covers different integration scenarios, ranging from lightweight AD pass-through authentication to full-fledged Kerberos trusted realms. In addition to this guide, you can find documentation on other features and services related to Red Hat Enterprise Linux Identity Management in the following guides: The Linux Domain Identity, Authentication, and Policy Guide documents Red Hat Identity Management, a solution that provides a centralized and unified way to manage identity stores as well as authentication and authorization policies in a Linux-based domain. The System-Level Authentication Guide documents different applications and services available to configure

authentication on local systems, including the authconfig utility, the System Security Services Daemon (SSSD) service, the Pluggable Authentication Module (PAM) framework, Kerberos, the certmonger utility, and single sign-on (SSO) for applications.

Table of Contents

CHAPTER 1. WAYS TO INTEGRATE ACTIVE DIRECTORY AND LINUX ENVIRONMENTS

IT environments have a structure. The systems in them are arranged with a purpose. Integrating two separate infrastructures requires an assessment of the purpose of each of those environments and an understanding of how and where they interact.

1.1. DEFINING WINDOWS INTEGRATION

Windows integration can mean very different things, depending on the required interaction between the Linux environment and the Windows environment. It could mean that individual Linux systems are enrolled into a Windows domain, it could mean that a Linux domain is configured to be a peer to the Windows domain, or it could simply mean that information is copied between environments.

There are several points of contact between a Windows domain and Linux systems. Each of these points revolve around identifying different domain objects (users, groups, systems, services) and the services which are used in that identification.

User Identities and Authentication

- Where are user accounts located; in a central authentication system running on Windows (AD domain) or in a central identity and authentication server running on Linux?

- How are users authenticated on a Linux system; through a local Linux authentication system or a central authentication system running on Windows?

- How is group membership configured for users? How is that group membership determined?

- Will users authenticate using a user name/password pair, Kerberos tickets, certificates, or a combination of methods?

- POSIX attributes are required to access services on Linux machines. How are these attributes stored: are they set in the Windows domain, configured locally on the Linux system, or dynamically mapped (for UID/GID numbers and Windows SIDs)?

- What users will be accessing what resources? Will Windows-defined users access Linux resources? Will Linux-defined users access Windows resources?

In most environments, the Active Directory domain is the central hub for user information, which means that there needs to be some way for Linux systems to access that user information for authentication requests. The real question then is *how* to obtain that user information and how much of that information is available to external systems. There also needs to be a balance between information required for Linux systems (POSIX attributes) and Linux users (certain application administrators) and how that information is managed.

Host and Service Principals

- What resources will be accessed?

- What authentication protocols are required?

- How will Kerberos tickets be obtained? How will SSL certificates be requested or verified?

- Will users need access to a single domain or to both Linux and Windows domains?

DNS Domains, Queries, and Name Resolution

- What will be the DNS configuration?

- Is there a single DNS domain? Are there subdomains?

- How will system host names be resolved?

- How will service discovery be configured?

Security Policies

- Where are access control instructions set?

- Which administrators are configured for each domain?

Change Management

- How frequently are systems added to the domain?

- If the underlying configuration for something related to Windows integration is changed, for example the DNS service, how are those changes propagated?

- Is configuration maintained through domain-related tools or a provisioning system?

- Does the integration path require additional applications or configuration on the Windows server?

As important as which elements in the domains are integrated, is how that integration is maintained. If a particular instrument of integration is heavily manual, yet the environment has a large number of systems which are frequently updated, then that one instrument may not work for that environment from a maintenance standpoint.

The following sections outline the main scenarios for integration with Windows. In direct integration, Linux systems are connected to Active Directory without any additional intermediaries. Indirect integration, on the other hand, involves an identity server that centrally manages Linux systems and connects the whole environment to Active Directory of the server-to-server level.

1.2. DIRECT INTEGRATION

You need two components to connect a Linux system to Active Directory (AD). One component interacts with the central identity and authentication source, which is AD in this case. The other component detects available domains and configures the first component to work with the right identity source. There are different options that can be used to retrieve information and perform authentication against AD. Among them are:

Native LDAP and Kerberos PAM and NSS modules

Among these modules are **nss_ldap**, **pam_ldap**, and **pam_krb5**. As PAM and NSS modules are loaded into every application process, they directly affect the execution environment. With no caching, offline support, or sufficient protection of access credentials, use of the basic LDAP and Kerberos modules for NSS and PAM is discouraged due to their limited functionality.

Samba Winbind

Samba Winbind had been a traditional way of connecting Linux systems to AD. Winbind emulates a Windows client on a Linux system and is able to communicate to AD servers.

Note that:

- The Winbind service must be running if you configured Samba as a domain member.

- Direct integration with Winbind in a multi-forest AD setup requires bidirectional trusts.

- Remote forests must trust the local forest to ensure that the **idmap_ad** plug-in handles remote forest users correctly.

System Security Services Daemon (SSSD)

The primary function of SSSD is to access a remote identity and authentication resource through a common framework that provides caching and offline support to the system. SSSD is highly configurable; it provides PAM and NSS integration and a database to store local users, as well as core and extended user data retrieved from a central server. SSSD is the recommended component to connect a Linux system with an identity server of your choice, be it Active Directory, Identity Management (IdM) in Red Hat Enterprise Linux, or any generic LDAP or Kerberos server.

Note that:

- Direct integration with SSSD works only within a single AD forest by default.

- Remote forests must trust the local forest to ensure that the **idmap_ad** plug-in handles remote forest users correctly.

The main reason to transition from Winbind to SSSD is that SSSD can be used for both direct and indirect integration and allows to switch from one integration approach to another without significant migration costs. The most convenient way to configure SSSD or Winbind in order to directly integrate a Linux system with AD is to use the **realmd** service. It allows callers to configure network authentication and domain membership in a standard way. The **realmd** service automatically discovers information about accessible domains and realms and does not require advanced configuration to join a domain or realm.

Direct integration is a simple way to introduce Linux systems to AD environment. However, as the share of Linux systems grows, the deployments usually see the need for a better centralized management of the identity-related policies such as host-based access control, sudo, or SELinux user mappings. At first, the configuration of these aspects of the Linux systems can be maintained in local configuration files. With a growing number of systems though, distribution and management of the configuration files is easier with a provisioning system such as Red Hat Satellite. This approach creates an overhead of changing the configuration files and then distributing them. When direct integration does not scale anymore, it is more beneficial to consider indirect integration described in the next section.

1.2.1. Supported Windows Platforms for direct integration

You can directly integrate your Linux machine with Active Directory forests that use the following forest and domain functional levels:

- Forest functional level range: Windows Server 2008 - Windows Server 2016[1]

- Domain functional level range: Windows Server 2008 - Windows Server 2016[1]

Direct integration has been tested on the following supported operating systems using the mentioned functional levels:

- Windows Server 2019

- Windows Server 2016

- Windows Server 2012 R2

1.3. INDIRECT INTEGRATION

The main advantage of the indirect integration is to manage Linux systems and policies related to those systems centrally while enabling users from Active Directory (AD) domains to transparently access Linux systems and services. There are two different approaches to the indirect integration:

Trust-based solution

The recommended approach is to leverage Identity Management (IdM) in Red Hat Enterprise Linux as the central server to control Linux systems and then establish cross-realm Kerberos trust with AD, enabling users from AD to log on to and to use single sign-on to access Linux systems and resources. This solution uses the Kerberos capability to establish trusts between different identity sources. IdM presents itself to AD as a separate forest and takes advantage of the forest-level trusts supported by AD.

In complex environments, a single IdM forest can be connected to multiple AD forests. This setup enables better separation of duties for different functions in the organization. AD administrators can focus on users and policies related to users while Linux administrators have full control over the Linux infrastructure. In such a case, the Linux realm controlled by IdM is analogous to an AD resource domain or realm but with Linux systems in it.

> **NOTE**
>
> In Windows, every domain is a Kerberos realm and a DNS domain at the same time. Every domain managed by the domain controller needs to have its own dedicated DNS zone. The same applies when IdM is trusted by AD as a forest. AD expects IdM to have its own DNS domain. For the trust setup to work, the DNS domain needs to be dedicated to the Linux environment.

Note that in trust environments, IdM enables you to use *ID views* to configure POSIX attributes for AD users on the IdM server. For details, see:

- Chapter 8, *Using ID Views in Active Directory Environments*

- SSSD Client-side Views in the *System-Level Authentication Guide*

Synchronization-based solution

An alternative to a trust-based solution is to leverage user synchronization capability, also available in IdM or Red Hat Directory Server (RHDS), allowing user accounts (and with RHDS also group accounts) to be synchronized from AD to IdM or RHDS, but not in the opposite direction. User synchronization has certain limitations, including:

- duplication of users

- the need to synchronize passwords, which requires a special component on all domain controllers in an AD domain

- to be able to capture passwords, all users must first manually change them

- synchronization supports only a single domain

- only one domain controller in AD can be used to synchronize data to one instance of IdM or RHDS

In some integration scenarios, the user synchronization may be the only available option, but in general, use of the synchronization approach is discouraged in favor of the cross-realm trust-based integration.

---

[1] Windows Server 2019 does not introduce a new functional level. The highest functional level Windows Server 2019 uses are Windows Server 2016.

PART I. ADDING A SINGLE LINUX SYSTEM TO AN ACTIVE DIRECTORY DOMAIN

This part describes how the System Security Services Daemon (**SSSD**) works with an Active Directory (**AD**) domain, how to use the **realmd** system to achieve direct domain integration, and finally, how to use **Samba** for **AD** integration.

CHAPTER 2. USING ACTIVE DIRECTORY AS AN IDENTITY PROVIDER FOR SSSD

The System Security Services Daemon (SSSD) is a system service to access remote directories and authentication mechanisms. It connects a local system (an SSSD *client*) to an external back-end system (a *domain*). This provides the SSSD client with access to identity and authentication remote services using an SSSD provider. For example, these remote services include: an LDAP directory, an Identity Management (IdM) or Active Directory (AD) domain, or a Kerberos realm.

When used as an identity management service for AD integration, SSSD is an alternative to services such as NIS or Winbind. This chapter describes how SSSD works with AD. For more details on SSSD, see the *System-Level Authentication Guide*.

2.1. HOW THE AD PROVIDER HANDLES TRUSTED DOMAINS

This section describes how SSSD handles trusted domains if you set **id_provider = ad** in the **/etc/sssd/sssd.conf** file.

- SSSD only supports domains in a single Active Directory forest. If SSSD requires access to multiple domains from multiple forests, consider using IdM with trusts (preferred) or the **winbindd** service instead of SSSD.

- By default, SSSD discovers all domains in the forest and, if a request for an object in a trusted domain arrives, SSSD tries to resolve it.

  If the trusted domains are not reachable or geographically distant, which makes them slow, you can set the ***ad_enabled_domains*** parameter in **/etc/sssd/sssd.conf** to limit from which trusted domains SSSD resolves objects.

- By default, you must use fully-qualified user names to resolve users from trusted domains.

2.2. CONFIGURING AN AD PROVIDER FOR SSSD

The AD provider enables SSSD to use the LDAP identity provider and the Kerberos authentication provider with optimizations for AD environments.

2.2.1. Overview of the Integration Options

Linux and Windows systems use different identifiers for users and groups:

- Linux uses *user IDs* (UID) and *group IDs* (GID). See Managing Users and Groups in the *System Administrator's Guide*. Linux UIDs and GIDs are compliant with the POSIX standard.

- Windows use *security IDs* (SID).
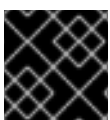
  > **IMPORTANT**
  >
  > Do not use the same user name in Windows and Active Directory.

Users authenticating to a Red Hat Enterprise Linux system, including AD users, must have a UID and GID assigned. For this purpose, SSSD provides the following integration options:

Automatically generate new UIDs and GIDs for AD users

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.

See Section 2.2.2, "Configuring an AD Domain with ID Mapping as a Provider for SSSD" .

NOTE

When all client systems use SSSD to map SIDs to Linux IDs, the mapping is consistent. If some clients use different software, choose one of the following:

- Ensure that the same mapping algorithm is used on all clients.

- Use explicit POSIX attributes, as described in Use POSIX attributes defined in AD.

Use POSIX attributes defined in AD

AD can create and store POSIX attributes, such as **uidNumber**, **gidNumber**, **unixHomeDirectory**, or **loginShell**.

When using ID mapping described in Automatically generate new UIDs and GIDs for AD users , SSSD creates new UIDs and GIDs, which overrides the values defined in AD. To keep the AD-defined values, you must disable ID mapping in SSSD.

See Section 2.2.3, "Configuring SSSD to Use POSIX Attributes Defined in AD" .

2.2.2. Configuring an AD Domain with ID Mapping as a Provider for SSSD

Prerequisites
Make sure that both the AD system and the Linux system are properly configured:

- Verify the configuration for name resolution. In particular, verify the DNS SRV records. For example, for a domain named **ad.example.com**:

  - To verify the DNS SRV LDAP records:

    ```
    # dig -t SRV _ldap._tcp.ad.example.com
    ```

  - To verify AD records:

    ```
    # dig -t SRV _ldap._tcp.dc._msdcs.ad.example.com
    ```

  If you later connect SSSD to a particular AD domain controller, it is not necessary to verify the DNS SRV records.

- Verify that system time on both systems is synchronized. This ensures that Kerberos is able to work properly.

- Ensure that the following ports on the AD domain controllers are open and accessible to the RHEL host.

Table 2.1. Ports Required for Direct Integration of Linux Systems into AD Using SSSD

Service	Port	Protocol	Notes
DNS	53	UDP and TCP	
LDAP	389	UDP and TCP	
Kerberos	88	UDP and TCP	
Kerberos	464	UDP and TCP	Used by kadmin for setting and changing a password
LDAP Global Catalog	3268	TCP	If the **id_provider = ad** option is being used
NTP	123	UDP	Optional
Samba	445	UDP and TCP	For AD Group Policy Objects (GPOs)

Configure the Local System

Red Hat recommends using the **realm join** command to configure the system. See Chapter 3, *Using realmd to Connect to an Active Directory Domain* . The **realmd** suite edits all required configuration files automatically. For example:

> # realm join *ad.example.com*

If you do not want to use **realmd**, you can configure the system manually. See Manually Connecting an SSSD Client to an Active Directory Domain in the Red Hat Knowledgebase.

Optional: Configure User Home Directories and Shells

The **pam_oddjob_mkhomedir.so** library automatically creates home directories when users first log in to the Linux system. By default, SSSD retrieves the format of the home directory from the AD identity provider. To customize the directory format on Linux clients:

1. Open the **/etc/sssd/sssd.conf** file.

2. In the **[domain]** section, use one of these options:

   - *fallback_homedir* sets a fallback home directory format, which is used only if a home directory is not defined in AD

   - *override_homedir* sets a home directory template, which always overrides the home directory defined in AD

   For example, to always use the format **/home/domain_name/user_name**:

```
[domain/EXAMPLE]
[... file truncated ...]
override_homedir = /home/%d/%u
```

For details, see the sssd.conf(5) man page.

By default, SSSD retrieves information about user shells from the ***loginShell*** parameter configured in AD. To customize the user shell settings on Linux clients:

1. Open the **/etc/sssd/sssd.conf** file.

2. Define the required user shell settings using these options:

   ○ ***shell_fallback*** sets a fallback value, which is used only if no shells are defined in AD

   ○ ***override_shell*** sets a value that always overrides the shell defined in AD

   ○ ***default_shell*** sets a default shell value

   ○ ***allowed_shells*** and ***vetoed_shells*** set lists of allowed or blacklisted shells

   For details, see the sssd.conf(5) man page.

Load the New Configuration

- Restart SSSD after changing the configuration file.

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

Additional Resources

- See the sssd-ldap(5) and sssd-krb5(5) man pages for other configuration options for LDAP and Kerberos providers.

- See the sssd-ad(5) man page for other configuration options for AD providers.

2.2.3. Configuring SSSD to Use POSIX Attributes Defined in AD

NOTE

Previously, the *Identity Management for UNIX* extension was available to provide POSIX attributes to user accounts. The extension is now deprecated. See the Microsoft Developer Network for details.

If you have been using Identity Management for UNIX, see this Knowledgebase article for answers to frequently asked questions.

For old procedures that reference Identity Management for Unix and the *Services for Unix* package, see these Red Hat Knowledgebase articles:

- Configuring an Active Directory Domain with POSIX Attributes

- Configuring Active Directory as an LDAP Domain

Recommendations

For best performance, publish the POSIX attributes to the AD global catalog. If POSIX attributes are not present in the global catalog, SSSD connects to the individual domain controllers directly on the LDAP port.

Join the Linux System to the AD Domain
Follow the steps in Section 2.2.2, "Configuring an AD Domain with ID Mapping as a Provider for SSSD" .

Disable ID Mapping in SSSD

1. Open the **/etc/sssd/sssd.conf** file.

2. In the AD domain section, add the ***ldap_id_mapping = false*** setting.

   > **NOTE**
   >
   > If you used the **realm** utility to join the domain and added the **--automatic-id-mapping=no** switch, the **realm** utility already set up SSSD with **ldap_id_mapping = false**.

3. If you previously requested any users with the default ID mapping configuration, remove the SSSD caches:

   ```
   rm -f /var/lib/sss/db/*
   ```

SSSD will now use POSIX attributes from AD, instead of creating them locally.

Additional Resources
For further details about ID mapping and the ***ldap_id_mapping*** parameter, see the sssd-ldap(8) man page.

2.3. AUTOMATIC KERBEROS HOST KEYTAB RENEWAL

SSSD automatically renews the Kerberos host keytab file in an AD environment if the adcli package is installed. The daemon checks daily if the machine account password is older than the configured value and renews it if necessary.

The default renewal interval is 30 days. To change the default:

1. Add the following parameter to the AD provider in your **/etc/sssd/sssd.conf** file:

   ```
   ad_maximum_machine_account_password_age = value_in_days
   ```

2. Restart SSSD:

   ```
   # systemctl restart sssd
   ```

To disable the automatic Kerberos host keytab renewal, set ***ad_maximum_machine_account_password_age = 0***.

2.4. ENABLING DYNAMIC DNS UPDATES

AD allows its clients to refresh their DNS records automatically. AD also actively maintains DNS records to make sure they are updated, including timing out (aging) and removing (scavenging) inactive records. DNS scavenging is not enabled by default on the AD side.

SSSD allows the Linux system to imitate a Windows client by refreshing its DNS record, which also prevents its record from being marked inactive and removed from the DNS record. When dynamic DNS updates are enabled, the client's DNS record is refreshed:

- when the identity provider comes online (always)

- when the Linux system reboots (always)

- at a specified interval (optional configuration); by default, the AD provider updates the DNS record every 24 hours

  You can set this behavior to the same interval as the DHCP lease. In this case, the Linux client is renewed after the lease is renewed.

DNS updates are sent to the AD server using Kerberos/GSSAPI for DNS (GSS-TSIG). This means that only secure connections need to be enabled.

The dynamic DNS configuration is set for each domain. For example:

```
[domain/ad.example.com]
id_provider = ad
auth_provider = ad
chpass_provider = ad
access_provider = ad

ldap_schema = ad

dyndns_update = true
dyndns_refresh_interval = 43200
dyndns_update_ptr = true
dyndns_ttl = 3600
```

For details on these options, see the sssd-ad(5) man page.

2.5. USING RANGE RETRIEVAL SEARCHES WITH SSSD

SSSD supports AD's *Searching Using Range Retrieval* feature. For details on range retrieval searches, see the Microsoft Developer Network.

> **IMPORTANT**
>
> If you set custom filters in the group or search bases, the filters might not work well with very large groups.

2.6. GROUP POLICY OBJECT ACCESS CONTROL

Group Policy is a Microsoft Windows feature that enables administrators to centrally manage policies for users and computers in Active Directory (AD) environments. A *group policy object* (GPO) is a collection of policy settings that are stored on a domain controller (DC) and can be applied to policy targets, such as computers and users. GPO policy settings related to Windows *logon rights* are commonly used to manage computer-based access control in AD environments.

2.6.1. How SSSD Works with GPO Access Control

When you configure SSSD to apply GPO access control, SSSD retrieves GPOs applicable to host systems and AD users. Based on the retrieved GPO configuration, SSSD determines if a user is allowed to log in to a particular host. This enables the administrator to define login policies honored by both Linux and Windows clients centrally on the AD domain controller.

IMPORTANT

Security filtering is a feature that enables you to further limit the scope of GPO access control to specific users, groups, or hosts by listing them in the security filter. However, SSSD only supports users and groups in the security filter. SSSD ignores host entries in the security filter.

To ensure that SSSD applies the GPO access control to a specific system, create a new OU in the AD domain, move the system to the OU, and then link the GPO to this OU.

2.6.2. GPO Settings Supported by SSSD

Table 2.2. GPO access control options retrieved by SSSD

GPO option [a]	Corresponding **sssd.conf** option [b]
Allow log on locally Deny log on locally	*ad_gpo_map_interactive*
Allow log on through Remote Desktop Services Deny log on through Remote Desktop Services	*ad_gpo_map_remote_interactive*
Access this computer from the network Deny access to this computer from the network	*ad_gpo_map_network*
Allow log on as a batch job Deny log on as a batch job	*ad_gpo_map_batch*
Allow log on as a service Deny log on as a service	*ad_gpo_map_service*

[a] As named in the Group Policy Management Editor on Windows.

[b] See the sssd-ad(5) man page for details about these options and for lists of pluggable authentication module (PAM) services to which the GPO options are mapped by default.

2.6.3. Configuring GPO-based Access Control for SSSD

GPO-based access control can be configured in the **/etc/sssd/sssd.conf** file. The *ad_gpo_access_control* option specifies the mode in which the GPO-based access control runs. It can be set to the following values:

ad_gpo_access_control = permissive

The **permissive** value specifies that GPO-based access control is evaluated but not enforced; a **syslog** message is recorded every time access would be denied. This is the default setting.

ad_gpo_access_control = enforcing

The **enforcing** value specifies that GPO-based access control is evaluated and enforced.

ad_gpo_access_control = disabled

The **disabled** value specifies that GPO-based access control is neither evaluated nor enforced.

IMPORTANT

Before starting to use the GPO-based access control and setting **ad_gpo_access_control** to enforcing mode, it is recommended to ensure that **ad_gpo_access_control** is set to permissive mode and examine the logs. By reviewing the **syslog** messages, you can test and adjust the current GPO settings as necessary before finally setting the enforcing mode.

The following parameters related to the GPO-based access control can also be specified in the **sssd.conf** file:

- The **ad_gpo_map_*** options and the **ad_gpo_default_right** option configure which PAM services are mapped to specific Windows logon rights.

  To add a PAM service to the default list of PAM services mapped to a specific GPO setting, or to remove the service from the list, use the **ad_gpo_map_*** options. For example, to remove the **su** service from the list of PAM services mapped to interactive login (GPO settings Allow log on locally and Deny log on locally):

  ```
  ad_gpo_map_interactive = -su
  ```

- The **ad_gpo_cache_timeout** option specifies the interval during which subsequent access control requests can reuse the files stored in the cache, instead of retrieving them from the DC anew.

For a detailed list of available GPO parameters as well as their descriptions and default values, see the sssd-ad(5) man page.

2.6.4. Additional Resources

- For more details on configuring SSSD to work with GPOs, see Configure SSSD to respect Active Directory SSH or Console/GUI GPOs in Red Hat Knowledgebase.

2.7. CREATING USER PRIVATE GROUPS AUTOMATICALLY USING SSSD

An SSSD client directly integrated into AD can automatically create a user private group for every AD user retrieved, ensuring that its GID matches the user's UID unless the GID number is already taken. To avoid conflicts, make sure that no groups with the same GIDs as user UIDs exist on the server.

The GID is not stored in AD. This ensures that AD users benefit from group functionality, while the LDAP database does not contain unnecessary empty groups.

2.7.1. Activating the Automatic Creation of User Private Groups for AD users

To activate the automatic creation of user private groups for AD users:

1. Edit the **/etc/sssd/sssd.conf** file, adding in the **[domain/LDAP]** section:

   ```
   auto_private_groups = true
   ```

2. Restart the sssd service, removing the sssd database:

   ```
   # service sssd stop ; rm -rf /var/lib/sss/db/* ; service sssd start
   ```

After performing this procedure, every AD user has a GID which is identical to the UID:

```
# id ad_user1
uid=121298(ad_user1) gid=121298(ad_user1) groups=121298(ad_user1),10000(Group1)
# id ad_user2
uid=121299(ad_user2) gid=121299(ad_user2) groups=121299(ad_user2),10000(Group1)
```

2.7.2. Deactivating the Automatic Creation of User Private Groups for AD users

To deactivate the automatic creation of user private groups for AD users:

1. Edit the **/etc/sssd/sssd.conf** file, adding in the **[domain/LDAP]** section:

   ```
   auto_private_groups = false
   ```

2. Restart the sssd service, removing the sssd database:

   ```
   # service sssd stop ; rm -rf /var/lib/sss/db/* ; service sssd start
   ```

After performing this procedure, all AD users have an identical, generic GID:

```
# id ad_user1
uid=121298(ad_user1) gid=10000(group1) groups=10000(Group1)
# id ad_user2
uid=121299(ad_user2) gid=10000(group1) groups=10000(Group1)
```

2.8. SSSD CLIENTS AND ACTIVE DIRECTORY DNS SITE AUTODISCOVERY

Active Directory forests can be very large, with numerous different domain controllers, domains and child domains, and physical sites. Active Directory uses the concept of sites to identify the physical location for its domain controllers. This enables clients to connect to the domain controller that is geographically closest, which increases client performance.

By default, SSSD clients use autodiscovery to find its AD site and connect to the closest domain controller. The process consists of these steps:

1. SSSD queries SRV records from the DNS server in the AD forest. The returned records contain the names of DCs in the forest.

2. SSSD sends an LDAP ping to each of these DCs. If a DC does not respond within a configured interval, the request times out and SSSD sends the LDAP ping to the next one. If the connection succeeds, the response contains information about the AD site the SSSD client belongs to.

3. SSSD then queries SRV records from the DNS server to locate DCs within the site it belongs to, and connects to one of them.

> **NOTE**
>
> SSSD remembers the AD site it belongs to by default. In this way, SSSD can send the LDAP ping directly to a DC in this site during the autodiscovery process to refresh the site information. Consequently, the procedure of autodiscovery is very fast as no timeouts occur normally.
>
> 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 the whole process again.

To override the autodiscovery, specify the AD site to which you want the client to connect by using the **ad_site** option in the [domain] section of the **/etc/sssd/sssd.conf** file.

Additional Resources

- See the sssd-ad(5) man page for details on **ad_site**.

- For environments with a trust between Identity Management and Active Directory, see Section 5.6, "Restricting Identity Management or SSSD to Selected Active Directory Servers or Sites in a Trusted Active Directory Domain".

2.9. TROUBLESHOOTING SSSD

For details about troubleshooting SSSD, see the *Troubleshooting SSSD* appendix in the *System-Level Authentication Guide*.

CHAPTER 3. USING REALMD TO CONNECT TO AN ACTIVE DIRECTORY DOMAIN

The **realmd** system provides a clear and simple way to discover and join identity domains to achieve direct domain integration. It configures underlying Linux system services, such as SSSD or Winbind, to connect to the domain.

Chapter 2, *Using Active Directory as an Identity Provider for SSSD* describes how to use the System Security Services Daemon (SSSD) on a local system and Active Directory as a back-end identity provider. Ensuring that the system is properly configured for this can be a complex task: there are a number of different configuration parameters for each possible identity provider and for SSSD itself. In addition, all domain information must be available in advance and then properly formatted in the SSSD configuration for SSSD to integrate the local system with AD.

The **realmd** system simplifies that configuration. It can run a discovery search to identify available AD and Identity Management domains and then join the system to the domain, as well as set up the required client services used to connect to the given identity domain and manage user access. Additionally, because SSSD as an underlying service supports multiple domains, **realmd** can discover and support multiple domains as well.

3.1. SUPPORTED DOMAIN TYPES AND CLIENTS

The **realmd** system supports the following domain types:

- Microsoft Active Directory

- Red Hat Enterprise Linux Identity Management

The following domain clients are supported by **realmd**:

- SSSD for both Red Hat Enterprise Linux Identity Management and Microsoft Active Directory

- Winbind for Microsoft Active Directory

3.2. PREREQUISITES FOR USING REALMD

To use the **realmd** system, install the realmd package.

```
# yum install realmd
```

In addition, make sure that the oddjob, oddjob-mkhomedir, sssd, and adcli packages are installed. These packages are required to be able to manage the system using **realmd**.

> **NOTE**
>
> As mentioned in Section 3.4, "Discovering and Joining Identity Domains", you can simply use **realmd** to find out which packages to install.

3.3. REALMD COMMANDS

The **realmd** system has two major task areas:

- managing system enrollment in a domain

- setting which domain users are allowed to access the local system resources

The central utility in **realmd** is called **realm**. Most **realm** commands require the user to specify the action that the utility should perform, and the entity, such as a domain or user account, for which to perform the action:

> realm *command arguments*

For example:

> realm join *ad.example.com*
> realm permit *user_name*

Table 3.1. realmd Commands

Command	Description
Realm Commands	
discover	Run a discovery scan for domains on the network.
join	Add the system to the specified domain.
leave	Remove the system from the specified domain.
list	List all configured domains for the system or all discovered and configured domains.
Login Commands	
permit	Enable access for specified users or for all users within a configured domain to access the local system.
deny	Restrict access for specified users or for all users within a configured domain to access the local system.

For more information about the **realm** commands, see the realm(8) man page.

3.4. DISCOVERING AND JOINING IDENTITY DOMAINS

The **realm discover** command returns complete domain configuration and a list of packages that must be installed for the system to be enrolled in the domain.

The **realm join** command then sets up the local machine for use with a specified domain by configuring both the local system services and the entries in the identity domain. The process run by **realm join** follows these steps:

1. Running a discovery scan for the specified domain.

2. Automatic installation of the packages required to join the system to the domain.

This includes SSSD and the PAM home directory job packages. Note that the automatic installation of packages requires the **PackageKit** suite to be running.

> **NOTE**
>
> If **PackageKit** is disabled, the system prompts you for the missing packages, and you will be required to install them manually using the **yum** utility.

3. Joining the domain by creating an account entry for the system in the directory.

4. Creating the **/etc/krb5.keytab** host keytab file.

5. Configuring the domain in SSSD and restarting the service.

6. Enabling domain users for the system services in PAM configuration and the **/etc/nsswitch.conf** file.

Discovering Domains

When run without any options, the **realm discover** command displays information about the default DNS domain, which is the domain assigned through the Dynamic Host Configuration Protocol (DHCP):

```
# realm discover
ad.example.com
  type: kerberos
  realm-name: AD.EXAMPLE.COM
  domain-name: ad.example.com
  configured: no
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  required-package: adcli
  required-package: samba-common
```

It is also possible to run a discovery for a specific domain. To do this, run **realm discover** and add the name of the domain you want to discover:

```
# realm discover ad.example.com
```

The **realmd** system will then use DNS SRV lookups to find the domain controllers in this domain automatically.

> **NOTE**
>
> The **realm discover** command requires NetworkManager to be running; in particular, it depends on the D–Bus interface of NetworkManager. If your system does not use NetworkManager, always specify the domain name in the **realm discover** command.

The **realmd** system can discover both Active Directory and Identity Management domains. If both domains exist in your environment, you can limit the discovery results to a specific type of server using the **--server-software** option. For example:

```
# realm discover --server-software=active-directory
```

One of the attributes returned in the discovery search is **login-policy**, which shows if domain users are allowed to log in as soon as the join is complete. If logins are not allowed by default, you can allow them manually by using the **realm permit** command. For details, see Section 3.7, "Managing Login Permissions for Domain Users".

For more information about the **realm discover** command, see the realm(8) man page.

Joining a Domain

IMPORTANT

Note that Active Directory domains require unique computer names to be used. Both NetBIOS computer name and its DNS host name should be uniquely defined and correspond to each other.

To join the system to an identity domain, use the **realm join** command and specify the domain name:

```
# realm join ad.example.com
realm: Joined ad.example.com domain
```

By default, the join is performed as the domain administrator. For AD, the administrator account is called **Administrator**; for IdM, it is called **admin**. To connect as a different user, use the **-U** option:

```
# realm join ad.example.com -U user
```

The command first attempts to connect without credentials, but it prompts for a password if required.

If Kerberos is properly configured on a Linux system, joining can also be performed with a Kerberos ticket for authentication. To select a Kerberos principal, use the **-U** option.

```
# kinit user
  # realm join ad.example.com -U user
```

The **realm join** command accepts several other configuration options. For more information about the **realm join** command, see the realm(8) man page.

Example 3.1. Example Procedure for Enrolling a System into a Domain

1. Run the **realm discover** command to display information about the domain.

```
# realm discover ad.example.com
ad.example.com
  type: kerberos
  realm-name: AD.EXAMPLE.COM
  domain-name: ad.example.com
  configured: no
  server-software: active-directory
  client-software: sssd
```

2. Run the **realm join** command and pass the domain name to the command. Provide the administrator password if the system prompts for it.

```
# realm join ad.example.com
Password for Administrator: password
```

Note that when discovering or joining a domain, **realmd** checks for the DNS SRV record:

- **_ldap._tcp.domain.example.com.** for Identity Management records

- **_ldap._tcp.dc._msdcs.domain.example.com.** for Active Directory records

The record is created by default when AD is configured, which enables it to be found by the service discovery.

Testing the System Configuration after Joining a Domain
To test whether the system was successfully enrolled into a domain, verify that you can log in as a user from the domain and that the user information is displayed correctly:

1. Run the **id _user@domain_name_** command to display information about a user from the domain.

   ```
   # id user@ad.example.com
   uid=1348601103(user@ad.example.com) gid=1348600513(domain
   group@ad.example.com) groups=1348600513(domain group@ad.example.com)
   ```

2. Using the **ssh** utility, log in as the same user.

   ```
   # ssh -l user@ad.example.com linux-client.ad.example.com
   user@ad.example.com@linux-client.ad.example.com's password:
   Creating home directory for user@ad.example.com.
   ```

3. Verify that the **pwd** utility prints the user's home directory.

   ```
   $ pwd
   /home/ad.example.com/user
   ```

4. Verify that the **id** utility prints the same information as the **id _user@domain_name_** command from the first step.

   ```
   $ id
   uid=1348601103(user@ad.example.com) gid=1348600513(domain
   group@ad.example.com) groups=1348600513(domain group@ad.example.com)
   context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
   ```

The **kinit** utility is also useful when testing whether the domain join was successful. Note that to use the utility, the krb5-workstation package must be installed.

3.5. REMOVING A SYSTEM FROM AN IDENTITY DOMAIN

To remove a system from an identity domain, use the **realm leave** command. The command removes the domain configuration from SSSD and the local system.

```
# realm leave ad.example.com
```

By default, the removal is performed as the default administrator. For AD, the administrator account is called **Administrator**; for IdM, it is called **admin**. If a different user was used to join to the domain, it might be required to perform the removal as that user. To specify a different user, use the **-U** option:

```
# realm leave ad.example.com -U 'AD.EXAMPLE.COM\user'
```

The command first attempts to connect without credentials, but it prompts for a password if required.

Note that when a client leaves a domain, the computer account is not deleted from the directory; the local client configuration is only removed. If you want to delete the computer account, run the command with the **--remove** option specified.

For more information about the **realm leave** command, see the realm(8) man page.

3.6. LISTING DOMAINS

The **realm list** command lists every configured domain for the system, as well as the full details and default configuration for that domain. This is the same information as is returned by the **realm discovery** command, only for a domain that is already in the system configuration.

```
# realm list --all --name-only
ad.example.com
```

The most notable options accepted by **realm list** are:

--all

>The **--all** option lists all discovered domains, both configured and unconfigured.

--name-only

>The **--name-only** option limits the results to the domain names and does not display the domain configuration details.

For more information about the **realm list** command, see the realm(8) man page.

3.7. MANAGING LOGIN PERMISSIONS FOR DOMAIN USERS

By default, *domain-side access control* is applied, which means that login policies for domain users are defined in the domain itself. This default behavior can be overridden so that *client-side access control* is used. With client-side access control, login permission are defined by local policies only.

If a domain applies client-side access control, you can use the **realmd** system to configure basic allow or deny access rules for users from that domain. Note that these access rules either allow or deny access to all services on the system. More specific access rules must be set on a specific system resource or in the domain.

To set the access rules, use the following two commands:

realm deny

>The **realm deny** command simply denies access to all users within the domain. Use this command with the **--all** option.

realm permit

The **realm permit** command can be used to:

- grant access to all users by using the **--all** option, for example:

  ```
  $ realm permit --all
  ```

- grant access to specified users, for example:

  ```
  $ realm permit user@example.com
  $ realm permit 'AD.EXAMPLE.COM\user'
  ```

- deny access to specified users by using the **-x** option, for example:

  ```
  $ realm permit -x 'AD.EXAMPLE.COM\user'
  ```

Note that allowing access currently only works for users in primary domains, not for users in trusted domains. This is because while user logins must contain the domain name, SSSD currently cannot provide **realmd** with information about available child domains.

> **IMPORTANT**
>
> It is safer to only allow access to specifically selected users or groups than to deny access to some, while enabling it to everyone else. Therefore, it is not recommended to allow access to all by default while only denying it to specified users with **realm permit -x**. Instead, Red Hat recommends to maintain a default no access policy for all users and only grant access to selected users using **realm permit**.

For more information about the **realm deny** and **realm permit** commands, see the realm(8) man page.

3.8. CHANGING DEFAULT USER CONFIGURATION

The **realmd** system supports modifying the default user home directory and shell POSIX attributes. For example, this might be required when some POSIX attributes are not set in the Windows user accounts or when these attributes are different from POSIX attributes of other users on the local system.

> **IMPORTANT**
>
> Changing the configuration as described in this section only works if the **realm join** command has not been run yet. If a system is already joined, change the default home directory and shell in the **/etc/sssd/sssd.conf** file, as described in the section called "Optional: Configure User Home Directories and Shells".

To override the default home directory and shell POSIX attributes, specify the following options in the **[users]** section in the **/etc/realmd.conf** file:

default-home

The **default-home** option sets a template for creating a home directory for accounts that have no home directory explicitly set. A common format is **/home/%d/%u**, where **%d** is the domain name and **%u** is the user name.

default-shell

The **default-shell** option defines the default user shell. It accepts any supported system shell.

For example:

```
[users]
default-home = /home/%u
default-shell = /bin/bash
```

For more information about the options, see the realmd.conf(5) man page.

3.9. ADDITIONAL CONFIGURATION FOR THE ACTIVE DIRECTORY DOMAIN ENTRY

Custom settings for each individual domain can be defined in the **/etc/realmd.conf** file. Each domain can have its own configuration section; the name of the section must match the domain name. For example:

```
[ad.example.com]
attribute = value
attribute = value
```

> **IMPORTANT**
>
> Changing the configuration as described in this section only works if the **realm join** command has not been run yet. If a system is already joined, changing these settings does not have any effect. In such situations, you must leave the domain, as described in Section 3.5, "Removing a System from an Identity Domain" , and then join again, as described in the section called "Joining a Domain" . Note that joining requires the domain administrator's credentials.

To change the configuration for a domain, edit the corresponding section in **/etc/realmd.conf**. The following example disables ID mapping for the **ad.example.com** domain, sets the host principal, and adds the system to the specified subtree:

```
[ad.example.com]
computer-ou = ou=Linux Computers,DC=domain,DC=example,DC=com
user-principal = host/linux-client@AD.EXAMPLE.COM
automatic-id-mapping = no
```

Note that the same configuration can also be set when originally joining the system to the domain using the **realm join** command, described in the section called "Joining a Domain" :

```
# realm join --computer-ou="ou=Linux Computers,dc=domain,dc=com" --automatic-id-mapping=no --user-principal=host/linux-client@AD.EXAMPLE.COM
```

Table 3.2, "Realm Configuration Options" lists the most notable options that can be set in the domain default section in **/etc/realmd.conf**. For complete information about the available configuration options, see the realmd.conf(5) man page.

Table 3.2. Realm Configuration Options

Option	Description
computer-ou	Sets the directory location for adding computer accounts to the domain. This can be the full DN or an RDN, relative to the root entry. The subtree must already exist.
user-principal	Sets the **userPrincipalName** attribute value of the computer account to the provided Kerberos principal.
automatic-id-mapping	Sets whether to enable dynamic ID mapping or disable the mapping and use POSIX attributes configured in Active Directory.

CHAPTER 4. USING SAMBA FOR ACTIVE DIRECTORY INTEGRATION

Samba implements the Server Message Block (SMB) protocol in Red Hat Enterprise Linux. The SMB protocol is used to access resources on a server, such as file shares and shared printers.

You can use Samba to authenticate Active Directory (AD) domain users to a Domain Controller (DC). Additionally, you can use Samba to share printers and local directories to other SMB clients in the network.

4.1. USING WINBINDD TO AUTHENTICATE DOMAIN USERS

Samba's **winbindd** service provides an interface for the Name Service Switch (NSS) and enables domain users to authenticate to AD when logging into the local system.

Using **winbindd** provides the benefit that you can enhance the configuration to share directories and printers without installing additional software. For further detail, see the section about Samba in the Red Hat System Administrator's Guide.

4.1.1. Joining an AD Domain

If you want to join an AD domain and use the **Winbind** service, use the **realm join --client-software=winbind** *domain_name* command. The **realm** utility automatically updates the configuration files, such as those for Samba, Kerberos, and PAM.

For further details and examples, see the *Setting up Samba as a Domain Member* section in the Red Hat System Administrator's Guide.

4.2. USING SMB SHARES WITH SSSD AND WINBIND

This section describes how you can use SSSD clients to access and fully use shares based on the Server Message Block (SMB) protocol, also known as the Common Internet File System (CIFS) protocol.

> **IMPORTANT**
>
> Using SSSD as a client in IdM or Active Directory domains has certain limitations, and Red Hat does not recommend using SSSD as ID mapping plug-in for Winbind. For further details, see the "What is the support status for Samba file server running on IdM clients or directly enrolled AD clients where SSSD is used as the client daemon" article.
>
> SSSD does not support all the services that Winbind provides. For example, SSSD does not support authentication using the NT LAN Manager (NTLM) or NetBIOS name lookup. If you need these services, use Winbind. Note that in Identity Management domains, Kerberos authentication and DNS name lookup are available for the same purposes.

4.2.1. How SSSD Works with SMB

The SMB file-sharing protocol is widely used on Windows machines. In Red Hat Enterprise Linux environments with a trust between Identity Management and Active Directory, SSSD enables seamless use of SMB as if it was a standard Linux file system.

To access a SMB share, the system must be able to translate Windows SIDs to Linux POSIX UIDs and GIDs. SSSD clients use the SID-to-ID or SID-to-name algorithm, which enables this ID mapping.

4.2.2. Switching Between SSSD and Winbind for SMB Share Access

This procedure describes how you can switch between SSSD and Winbind plug-ins that are used for accessing SMB shares from SSSD clients. For Winbind to be able to access SMB shares, you need to have the cifs-utils package installed on your client. To make sure that cifs-utils is installed on your machine:

```
$ rpm -q cifs-utils
```

1. *Optional*. Find out whether you are currently using SSSD or Winbind to access SMB shares from the SSSD client:

   ```
   # alternatives --display cifs-idmap-plugin
   cifs-idmap-plugin - status is auto.
    link currently points to /usr/lib64/cifs-utils/cifs_idmap_sss.so
   /usr/lib64/cifs-utils/cifs_idmap_sss.so - priority 20
   /usr/lib64/cifs-utils/idmapwb.so - priority 10
   Current `best' version is /usr/lib64/cifs-utils/cifs_idmap_sss.so.
   ```

   If the SSSD plug-in (**cifs_idmap_sss.so**) is installed, it has a higher priority than the Winbind plug-in (**idmapwb.so**) by default.

2. Before switching to the Winbind plug-in, make sure Winbind is running on the system:

   ```
   # systemctl is-active winbind.service
   active
   ```

   Before switching to the SSSD plug-in, make sure SSSD is running on the system:

   ```
   # systemctl is-active sssd.service
   active
   ```

3. To switch to a different plug-in, use the **alternatives --set cifs-idmap-plugin** command, and specify the path to the required plug-in. For example, to switch to Winbind:

   ```
   # alternatives --set cifs-idmap-plugin /usr/lib64/cifs-utils/idmapwb.so
   ```

> **NOTE**
>
> The 32-bit version platform, such as i686 in RHEL 7, uses the **/usr/lib/cifs-utils/** directory instead of **/usr/lib64/cifs-utils/**.

4.3. ADDITIONAL RESOURCES

For details about Samba, see the corresponding section in the Red Hat System Administrator's Guide.

PART II. INTEGRATING A LINUX DOMAIN WITH AN ACTIVE DIRECTORY DOMAIN: CROSS-FOREST TRUST

This part provides recommended practices for integrating a **Linux Domain** with an **Active Directory** domain by creating, configuring, and managing a cross-forest trust environment.

CHAPTER 5. CREATING CROSS-FOREST TRUSTS WITH ACTIVE DIRECTORY AND IDENTITY MANAGEMENT

This chapter describes creating cross-forest trusts between Active Directory and Identity Management. A cross-forest trust is the recommended one of the two methods to integrate Identity Management and Active Directory (AD) environments indirectly. The other method is synchronization. If you are unsure which method to choose for your environment, read Section 1.3, "Indirect Integration".

Kerberos implements a concept of a *trust*. In a trust, a principal from one Kerberos realm can request a ticket to a service in another Kerberos realm. Using this ticket, the principal can authenticate against resources on machines belonging to the other realm.

Kerberos also has the ability to create a relationship between two otherwise separate Kerberos realms: a *cross-realm trust*. Realms that are part of a trust use a shared pair of a ticket and key; a member of one realm then counts as a member of both realms.

Red Hat Identity Management supports configuring a cross-forest trust between an IdM domain and an Active Directory domain.

5.1. INTRODUCTION TO CROSS-FOREST TRUSTS

Kerberos realm only concerns authentication. Other services and protocols are involved in complementing identity and authorization for resources running on the machines in the Kerberos realm.

As such, establishing Kerberos cross-realm trust is not enough to allow users from one realm to access resources in the other realm; a support is required at other levels of communication as well.

5.1.1. The Architecture of a Trust Relationship

Both Active Directory and Identity Management manage a variety of core services such as Kerberos, LDAP, DNS, or certificate services. To transparently integrate these two diverse environments, all core services must interact seamlessly with one another.

Active Directory Trusts, Forests, and Cross-forest Trusts
Kerberos cross-realm trust plays an important role in authentication between Active Directory environments. All activities to resolve user and group names in a trusted AD domain require authentication, regardless of how access is performed: using LDAP protocol or as part of the Distributed Computing Environment/Remote Procedure Calls (DCE/RPC) on top of the Server Message Block (SMB) protocol. Because there are more protocols involved in organizing access between two different Active Directory domains, trust relationship has a more generic name, *Active Directory trust*.

Multiple AD domains can be organized together into an *Active Directory forest*. A root domain of the forest is the first domain created in the forest. Identity Management domain cannot be part of an existing AD forest, thus it is always seen as a separate forest.

When trust relationship is established between two separate forest root domains, allowing users and services from *different* AD forests to communicate, a trust is called *Active Directory cross-forest trust*.

Trust Flow and One-way Trusts
A trust establishes an access relationship between two domains. Active Directory environments can be complex so there are different possible types and arrangements for Active Directory trusts, between child domains, root domains, or forests. A trust is a path from one domain to another. The way that identities and information move between the domains is called a *trust flow*.

The *trusted domain* contains users, and the *trusting domain* allows access to resources. In a one-way

trust, trust flows only in one direction: users can access the trusting domain's resources but users in the trusting domain cannot access resources in the trusted domain. In Figure 5.1, "One-way Trust", Domain A is trusted by Domain B, but Domain B is not trusted by Domain A.
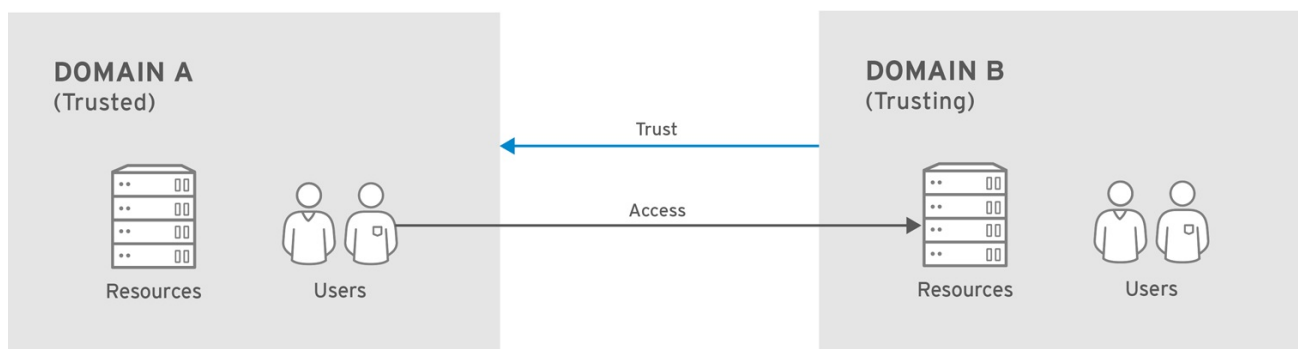
Figure 5.1. One-way Trust

IdM allows the administrator to configure both one-way and two-way trusts. For details, see Section 5.1.4, "One-Way and Two-Way Trusts".

Transitive and Non-transitive Trusts

Trusts can be *transitive* so that a domain trusts another domain and any other domain trusted by that second domain.
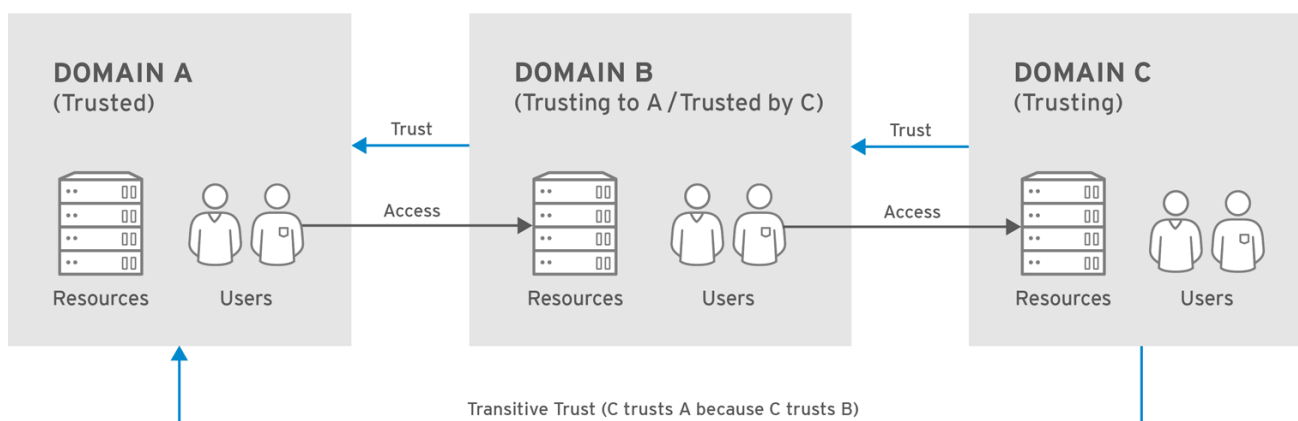
Figure 5.2. Transitive Trusts

Trusts can also be *non-transitive* which means the trust is limited only to the explicitly included domains.

Cross-forest Trust in Active Directory and Identity Management

Within an Active Directory forest, trust relationships between domains are normally two-way and transitive by default.

Because trust between two AD forests is a trust between two forest root domains, it can also be two-way or one-way. The transitivity of the cross-forest trust is explicit: any domain trust within an AD forest that leads to the root domain of the forest is transitive over the cross-forest trust. However, separate cross-forest trusts are not transitive. An explicit cross-forest trust must be established between each AD forest root domain to another AD forest root domain.

From the perspective of AD, Identity Management represents a separate AD forest with a single AD domain. When cross-forest trust between an AD forest root domain and an IdM domain is established, users from the AD forest domains can interact with Linux machines and services from the IdM domain.
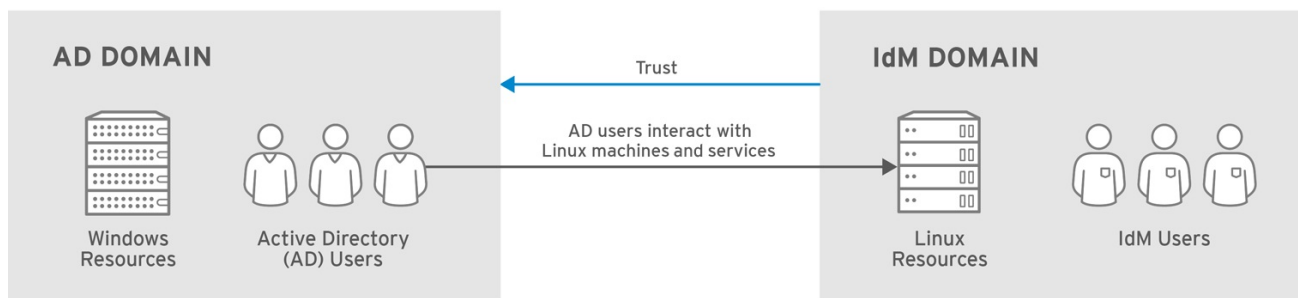
Figure 5.3. Trust Direction

5.1.2. Active Directory Security Objects and Trust

Active Directory Global Catalog
The global catalog contains information about objects of an Active Directory. It stores a full copy of objects within its own domain. From objects of other domains in the Active Directory forest, only a partial copy of the commonly most searched attributes is stored in the global catalog. Additionally, some types of groups are only valid within a specific scope and might not be part of the global catalog.

Note that the cross-forest trust context is wider than a single domain. Therefore, some of these server-local or domain-local security group memberships from a trusted forest might not be visible to IdM servers.

Global Catalog and POSIX Attributes
Active Directory does not replicate POSIX attributes with its default settings. If it is required to use POSIX attributes that are defined in AD Red Hat strongly recommends to replicate them to the global catalog service.

5.1.3. Trust Architecture in IdM

On the Identity Management side, the IdM server has to be able to recognize Active Directory identities and appropriately process their group membership for access controls. The Microsoft PAC (MS-PAC, Privilege Account Certificate) contains the required information about the user; their security ID, domain user name, and group memberships. Identity Management has two components to analyze data in the PAC on the Kerberos ticket:

- SSSD, to perform identity lookups on Active Directory and to retrieve user and group security identifiers (SIDs) for authorization. SSSD also caches user, group, and ticket information for users and maps Kerberos and DNS domains,

- Identity Management (Linux domain management), to associate the Active Directory user with an IdM group for IdM policies and access.
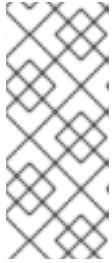
NOTE

Access control rules and policies for Linux domain administration, such as SELinux, sudo, and host-based access controls, are defined and applied through Identity Management. Any access control rules set on the Active Directory side are not evaluated or used by IdM; the only Active Directory configuration which is relevant is group membership.

Trusts with Different Active Directory Forests

IdM can also be part of trust relationships with different AD forests. Once a trust is established, additional trusts with other forests can be added later, following the same commands and procedures. IdM can trust multiple entirely unrelated forests at the same time, allowing users from such unrelated AD forests access to resources in the same shared IdM domain.

5.1.3.1. Active Directory PACs and IdM Tickets

Group information in Active Directory is stored in a list of identifiers in the *Privilege Attribute Certificate* (MS-PAC or PAC) data set. The PAC contains various authorization information, such as group membership or additional credentials information. It also includes *security identifiers* (SIDs) of users and groups in the Active Directory domain. SIDs are identifiers assigned to Active Directory users and groups when they are created. In trust environments, group members are identified by SIDs, rather than by names or DNs.

A PAC is embedded in the Kerberos service request ticket for Active Directory users as a way of identifying the entity to other Windows clients and servers in the Windows domain. IdM maps the group information in the PAC to the Active Directory groups and then to the corresponding IdM groups to determine access.

When an Active Directory user requests a ticket for a service on IdM resources, the process goes as follows:

1. The request for a service contains the PAC of the user. The IdM Kerberos Distribution Centre (KDC) analyzes the PAC by comparing the list of Active Directory groups to memberships in IdM groups.

2. For SIDs of the Kerberos principal defined in the MS-PAC, the IdM KDC evaluates external group memberships defined in the IdM LDAP. If additional mappings are available for an SID, the MS-PAC record is extended with other SIDs of the IdM groups to which the SID belongs. The resulting MS-PAC is signed by the IdM KDC.

3. The service ticket is returned to the user with the updated PAC signed by the IdM KDC. Users belonging to AD groups known to the IdM domain can now be recognized by SSSD running on the IdM clients based on the MS-PAC content of the service ticket. This allows to reduce identity traffic to discover group memberships by the IdM clients.

When the IdM client evaluates the service ticket, the process includes the following steps:

1. The Kerberos client libraries used in the evaluation process send the PAC data to the SSSD PAC responder.

2. The PAC responder verifies the group SIDs in the PAC and adds the user to the corresponding groups in the SSSD cache. SSSD stores multiple TGTs and tickets for each user as new services are accessed.

3. Users belonging to the verified groups can now access the required services on the IdM side.

5.1.3.2. Active Directory Users and Identity Management Groups

When managing Active Directory users and groups, you can add individual AD users and whole AD groups to Identity Management groups.

For a description of how to configure IdM groups for AD users, see Section 5.3.3, "Creating IdM Groups for Active Directory Users".

Non-POSIX External Groups and SID Mapping

Group membership in the IdM LDAP is expressed by specifying a distinguished name (DN) of an LDAP object that is a member of a group. AD entries are not synchronized or copied over to IdM, which means that AD users and groups have no LDAP objects in the IdM LDAP. Therefore, they cannot be directly used to express group membership in the IdM LDAP.

For this reason, IdM creates *non-POSIX external groups*: proxy LDAP objects that contain references to SIDs of AD users and groups as strings. Non-POSIX external groups are then referenced as normal IdM LDAP objects to signify group membership for AD users and groups in IdM.

SIDs of non-POSIX external groups are processed by SSSD; SSSD maps SIDs of groups to which an AD user belongs to POSIX groups in IdM. The SIDs on the AD side are associated with user names. When the user name is used to access IdM resources, SSSD in IdM resolves that user name to its SID, and then looks up the information for that SID within the AD domain, as described in Section 5.1.3.1, "Active Directory PACs and IdM Tickets".

ID Ranges

When a user is created in Linux, it is assigned a user ID number. In addition, a private group is created for the user. The private group ID number is the same as the user ID number. In Linux environment, this does not create a conflict. On Windows, however, the security ID number must be unique for every object in the domain.

Trusted AD users require a UID and GID number on a Linux system. This UID and GID number can be generated by IdM, but if the AD entry already has UID and GID numbers assigned, assigning different numbers creates a conflict. To avoid such conflicts, it is possible to use the AD-defined POSIX attributes, including the UID and GID number and preferred login shell.

> **NOTE**
>
> AD stores a subset of information for all objects within the forest in a *global catalog*. The global catalog includes every entry for every domain in the forest. If you want to use AD-defined POSIX attributes, Red Hat strongly recommends that you first replicate the attributes to the global catalog.

When a trust is created, IdM automatically detects what kind of ID range to use and creates a unique ID range for the AD domain added to the trust. You can also choose this manually by passing one of the following options to the **ipa trust-add** command:

ipa-ad-trust

This range option is used for IDs algorithmically generated by IdM based on the SID.

If IdM generates the SIDs using SID-to-POSIX ID mapping, the ID ranges for AD and IdM users and groups must have unique, non-overlapping ID ranges available.

ipa-ad-trust-posix

This range option is used for IDs defined in POSIX attributes in the AD entry.

IdM obtains the POSIX attributes, including ***uidNumber*** and ***gidNumber***, from the global catalog in AD or from the directory controller. If the AD domain is managed correctly and without ID conflicts, the ID numbers generated in this way are unique. In this case, no ID validation or ID range is required.

For example:

```
[root@ipaserver ~]# ipa trust-add name_of_the_trust --range-type=ipa-ad-trust-posix
```

Recreating a trust with the other ID range
If the ID range of the created trust does not suit your deployment, you can re-create the trust using the other **--range-type** option:

1. View all the ID ranges that are currently in use:

   ```
   [root@ipaserver ~]# ipa idrange-find
   ```

   In the list, identify the name of the ID range that was created by the **ipa trust-add** command. The first part of the name of the ID range is the name of the trust: *name_of_the_trust*_id_range, for example *ad.example.com*.

2. (Optional) If you do not know which **--range-type** option, **ipa-ad-trust** or **ipa-ad-trust-posix**, was used when the trust was created, identify the option:

   ```
   [root@ipaserver ~]# ipa idrange-show name_of_the_trust_id_range
   ```

   Make note of the type so that you choose the opposite type for the new trust in Step 5.

3. Remove the range that was created by the **ipa trust-add** command:

   ```
   [root@ipaserver ~]# ipa idrange-del name_of_the_trust_id_range
   ```

4. Remove the trust:

   ```
   [root@ipaserver ~]# ipa trust-del name_of_the_trust
   ```

5. Create a new trust with the correct **--range-type** option. For example:

   ```
   [root@ipaserver ~]# ipa trust-add name_of_the_trust --range-type=ipa-ad-trust
   ```

5.1.3.3. Active Directory Users and IdM Policies and Configuration

Several IdM policy definitions, such as SELinux, host-based access control, sudo, and netgroups, rely on user groups to identify how the policies are applied.
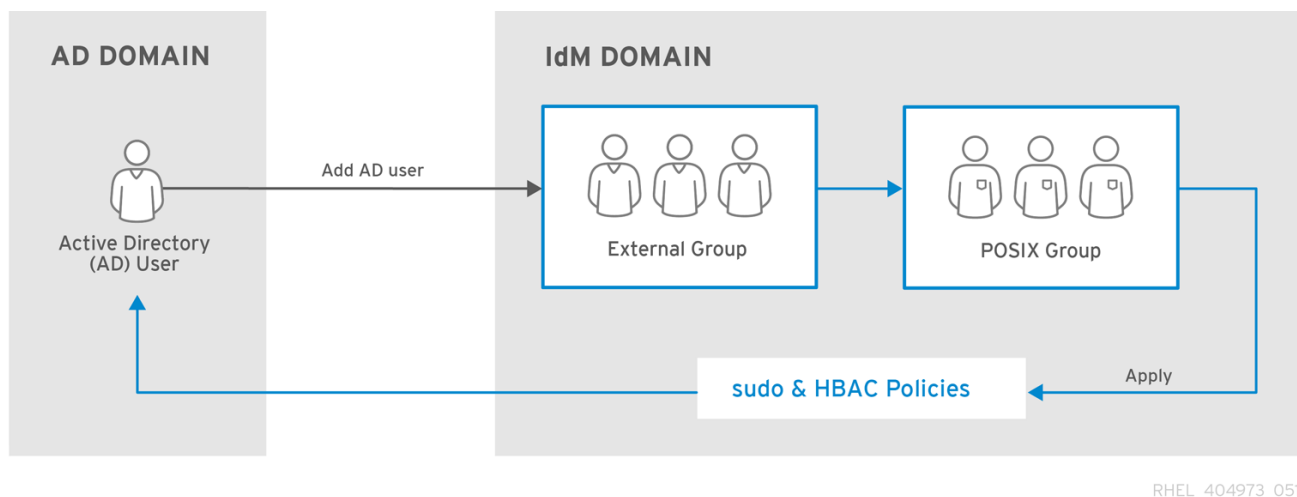
RHEL_404973_0516

Figure 5.4. Active Directory Users and IdM Groups and Policies

Active Directory users are external to the IdM domain, but they can still be added as group members to IdM groups, as long as those groups are configured as external groups described in Section 5.1.3.2, "Active Directory Users and Identity Management Groups". In such cases, the sudo, host-based access controls, and other policies are applied to the external POSIX group and, ultimately, to the AD user when accessing IdM domain resources.

The user SID in the PAC in the ticket is resolved to the AD identity. This means that Active Directory users can be added as group members using their fully-qualified user name or their SID.

5.1.4. One-Way and Two-Way Trusts

IdM supports two types of trust agreements, depending on whether the entities that can establish connection to services in IdM are limited to only AD or can include IdM entities as well.

One-way trust

> One-way trust enables AD users and groups to access resources in IdM, but not the other way around. The IdM domain trusts the AD forest, but the AD forest does not trust the IdM domain.
>
> One-way trust is the default mode for creating a trust.

Two-way trust

> Two-way trust enables AD users and groups to access resources in IdM. You must configure a two-way trust for solutions such as Microsoft SQL Server that expect the S4U2Self and S4U2Proxy Microsoft extensions to the Kerberos protocol to work over a trust boundary. An application on a RHEL IdM host might request S4U2Self or S4U2Proxy information from an Active Directory domain controller about an AD user, and a two-way trust provides this feature.
>
> Note that this two-way trust functionality does not allow IdM users to login to Windows systems, and the two-way trust in IdM does not give the users any additional rights compared to the one-way trust solution in AD.

For more general information on one-way and two-way trusts, see Section 5.1.1, "The Architecture of a Trust Relationship".

After a trust is established, it is not possible to modify its type. If you require a different type of trust, run the **ipa trust-add** command again; by doing this, you can delete the existing trust and establish a new one.

5.1.5. External Trusts to Active Directory

An external trust is a trust relationship between domains that are in a different forests. While forest trusts always require to establish the trust between the root domains of Active Directory forests, you can establish an external trust to any domain within the forest.

External trusts are non-transitive. For this reason, users and groups from other Active Directory domains have no access to IdM resources. For further information, see the section called "Transitive and Non-transitive Trusts".

5.1.6. Trust Controllers and Trust Agents

IdM provides the following types of IdM servers that support trust to Active Directory:

Trust controllers

IdM servers that can control the trust and perform identity lookups against Active Directory domain controllers (DC). Active Directory domain controllers contact trust controllers when establishing and verifying the trust to Active Directory. The first trust controller is created when you configure the trust.

For details about configuring an IdM server as a trust controller, see Section 5.2.2, "Creating Trusts".

Trust controllers run an increased amount of network-facing services compared to trust agents, and thus present a greater attack surface for potential intruders.

Trust agents

IdM servers that can perform identity lookups against Active Directory domain controllers.

For details about configuring an IdM server as a trust agent, see Section 5.2.2.1.1, "Preparing the IdM Server for Trust".

In addition to trust controllers and agents, the IdM domain can also include replicas without any role. However, these servers do not communicate with Active Directory. Therefore, clients that communicate with these servers cannot resolve Active Directory users and groups or authenticate and authorize Active Directory users.

Table 5.1. A comparison of the capabilities provided by trust controllers and trust agents

Capability	Trust controllers	Trust agents
Resolve Active Directory users and groups	Yes	Yes
Enroll IdM clients that run services accessible by users from trusted Active Directory forests	Yes	Yes
Manage the trust (for example, add trust agreements)	Yes	No

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

- Configure at least two trust controllers per Identity Management 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 replica. To do this, use the **ipa-adtrust-install** utility on the IdM server as described in .

> **IMPORTANT**
>
> You cannot downgrade an existing trust controller to a trust agent. The trust controller server role, once installed, cannot be removed from the topology.

5.2. CREATING CROSS-FOREST TRUSTS

5.2.1. Environment and Machine Requirements

Before configuring a trust agreement, make sure that both the Active Directory and Identity Management servers, machines, and environments meet the requirements and settings described in this section.

5.2.1.1. Supported Windows Platforms

You can establish a trust relationship with Active Directory 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

The following operating systems are supported and tested for establishing a trust using the mentioned functional levels:

- Windows Server 2012 R2

- Windows Server 2016

Previous versions of Windows Server are not supported for establishing a trust.

5.2.1.2. DNS and Realm Settings

To establish a trust, Active Directory and Identity Management require specific DNS configuration:

Unique primary DNS domains

Each system must have its own unique primary DNS domain configured. For example:

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

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

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

> **IMPORTANT**
>
> If the IdM domain is the parent domain of the AD domain, the IdM servers must run on Red Hat Enterprise Linux 7.5 or later.

The most convenient management solution is an environment where each DNS domain is managed by integrated DNS servers, but it is possible to use any other standard-compliant DNS server as well.

It is not possible for AD or IdM to share the primary DNS domain with another system for identity management. For more information, see documentation for host name and DNS configuration requirements in the Linux Domain Identity, Authentication, and Policy Guide .

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

Kerberos realm names must be 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 are required to be **AD.EXAMPLE.COM** and **IDM.EXAMPLE.COM**.

DNS records resolvable from all DNS domains in the trust

All machines must be able to resolve DNS records from all DNS domains involved in the trust relationship:

- When configuring IdM DNS, follow the instructions described in the section on configuring DNS services within the IdM domain and section on managing DNS forwarding in the *Linux Domain Identity, Authentication, and Policy Guide*.

- If you are using IdM without integrated DNS, follow the instructions described in the section describing the server installation without integrated DNS in the *Linux Domain Identity, Authentication, and Policy Guide*.

No overlap between IdM and AD DNS domains

Systems joined to IdM can be distributed over multiple DNS domains. DNS domains containing IdM clients must not overlap with DNS domains containing machines joined to AD. The primary IdM DNS domain must have proper SRV records to support AD trusts.
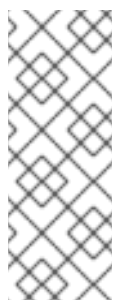
> **NOTE**
>
> In some environments with trusts between IdM and Active Directory, you can install an IdM client on a host that is part of the Active Directory DNS domain. The host can then benefit from the Linux-focused features of IdM. This is not a recommended configuration and has some limitations. Red Hat recommends to always deploy IdM clients in a DNS zone different from the ones owned by Active Directory and access IdM clients through their IdM host names.

You can acquire a list of the required SRV records specific to your system setup by running the **$ ipa dns-update-system-records --dry-run** command.

The generated list can look for example like this:

```
$ ipa dns-update-system-records --dry-run
 IPA DNS records:
  _kerberos-master._tcp.example.com. 86400 IN SRV 0 100 88 server.example.com.
  _kerberos-master._udp.example.com. 86400 IN SRV 0 100 88 server.example.com.
  _kerberos._tcp.example.com. 86400 IN SRV 0 100 88 server.example.com.
  _kerberos._udp.example.com. 86400 IN SRV 0 100 88 server.example.com.
  _kerberos.example.com. 86400 IN TXT "EXAMPLE.COM"
  _kpasswd._tcp.example.com. 86400 IN SRV 0 100 464 server.example.com.
```

> _kpasswd._udp.example.com. 86400 IN SRV 0 100 464 server.example.com.
> _ldap._tcp.example.com. 86400 IN SRV 0 100 389 server.example.com.
> _ntp._udp.example.com. 86400 IN SRV 0 100 123 server.example.com.

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

Verifying the DNS Configuration

Before configuring trust, verify that the Identity Management and Active Directory servers can resolve themselves and also each other.

If running the commands described below does not display the expected results, inspect the DNS configuration on the host where the commands were executed. If the host configuration seems correct, make sure that DNS delegations from the parent to child domains are set up correctly.

Note that AD caches the results of DNS lookups, and changes you make in DNS are therefore sometimes not visible immediately. You can delete the current cache by running the **ipconfig /flushdns** command.

Verify that the IdM-hosted services are resolvable from the IdM domain server used for establishing trust

1. Run a DNS query for the Kerberos over UDP and LDAP over TCP service records.

   > [root@ipaserver ~]# dig +short -t SRV _kerberos._udp.*ipa.example.com.*
   > 0 100 88 ipamaster1.ipa.example.com.
   >
   > [root@ipaserver ~]# dig +short -t SRV _ldap._tcp.*ipa.example.com.*
   > 0 100 389 ipamaster1.ipa.example.com.

   The commands are expected to list all IdM servers.

2. Run a DNS query for the TXT record with the IdM Kerberos realm name. The obtained value is expected to match the Kerberos realm that you specified when installing IdM.

   > [root@ipaserver ~]# dig +short -t TXT _kerberos.*ipa.example.com.*
   > IPA.EXAMPLE.COM

3. After you execute the **ipa-adtrust-install** utility, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust", run a DNS query for the MS DC Kerberos over UDP and LDAP over TCP service records.

   > [root@ipaserver ~]# dig +short -t SRV _kerberos._udp.dc._msdcs.*ipa.example.com.*
   > 0 100 88 ipamaster1.ipa.example.com.
   >
   > [root@ipaserver ~]# dig +short -t SRV _ldap._tcp.dc._msdcs.*ipa.example.com.*
   > 0 100 389 ipamaster1.ipa.example.com.

   The commands are expected to list all IdM servers on which **ipa-adtrust-install** has been executed. Note that the output is empty if **ipa-adtrust-install** has not been executed on any IdM server, which is typically before establishing the very first trust relationship.

Verify that IdM is able to resolve service records for AD

Run a DNS query for the Kerberos over UDP and LDAP over TCP service records.

```
[root@ipaserver ~]# dig +short -t SRV _kerberos._udp.dc._msdcs.ad.example.com.
0 100 88 addc1.ad.example.com.

[root@ipaserver ~]# dig +short -t SRV _ldap._tcp.dc._msdcs.ad.example.com.
0 100 389 addc1.ad.example.com.
```

These commands are expected to return the names of AD domain controllers.

Verify that the IdM-hosted services are resolvable from the AD server

1. On the AD server, set the **nslookup.exe** utility to look up service records.

   ```
   C:\>nslookup.exe
   > set type=SRV
   ```

2. Enter the domain name for the Kerberos over UDP and LDAP over TCP service records.

   ```
   > _kerberos._udp.ipa.example.com.
   _kerberos._udp.ipa.example.com.     SRV service location:
     priority          = 0
     weight            = 100
     port              = 88
     svr hostname   = ipamaster1.ipa.example.com
   > _ldap._tcp.ipa.example.com
   _ldap._tcp.ipa.example.com     SRV service location:
     priority          = 0
     weight            = 100
     port              = 389
     svr hostname   = ipamaster1.ipa.example.com
   ```

   The expected output contains the same set of IdM servers as displayed in Verify that the IdM-hosted services are resolvable from the IdM domain server used for establishing trust.

3. Change the service type to TXT and run a DNS query for the TXT record with the IdM Kerberos realm name.

   ```
   C:\>nslookup.exe
   > set type=TXT
   > _kerberos.ipa.example.com.
   _kerberos.ipa.example.com.     text =

       "IPA.EXAMPLE.COM"
   ```

   The output is expected to contain the same value as displayed in Verify that the IdM-hosted services are resolvable from the IdM domain server used for establishing trust.

4. After you execute the **ipa-adtrust-install** utility, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust", run a DNS query for the MS DC Kerberos over UDP and LDAP over TCP service records.

   ```
   C:\>nslookup.exe
   ```

```
> set type=SRV
> _kerberos._udp.dc._msdcs.ipa.example.com.
_kerberos._udp.dc._msdcs.ipa.example.com.      SRV service location:
   priority = 0
   weight = 100
   port = 88
   svr hostname = ipamaster1.ipa.example.com
> _ldap._tcp.dc._msdcs.ipa.example.com.
_ldap._tcp.dc._msdcs.ipa.example.com.       SRV service location:
   priority = 0
   weight = 100
   port = 389
   svr hostname = ipamaster1.ipa.example.com
```

The command is expected to list all IdM servers on which the **ipa-adtrust-install** utility has been executed. Note that the output is empty if **ipa-adtrust-install** has not been executed on any IdM server, which is typically before establishing the very first trust relationship.

Verify that AD services are resolvable from the AD server

1. On the AD server, set the **nslookup.exe** utility to look up service records.

   ```
   C:\>nslookup.exe
   > set type=SRV
   ```

2. Enter the domain name for the Kerberos over UDP and LDAP over TCP service records.

   ```
   > _kerberos._udp.dc._msdcs.ad.example.com.
   _kerberos._udp.dc._msdcs.ad.example.com.  SRV service location:
      priority = 0
      weight = 100
      port = 88
      svr hostname = addc1.ad.example.com
   > _ldap._tcp.dc._msdcs.ad.example.com.
   _ldap._tcp.dc._msdcs.ad.example.com.  SRV service location:
      priority = 0
      weight = 100
      port = 389
      svr hostname = addc1.ad.example.com
   ```

   The expected output contains the same set of AD servers as displayed in Verify that IdM is able to resolve service records for AD.

5.2.1.3. NetBIOS Names

The NetBIOS name is critical for identifying the Active Directory (AD) domain and, if IdM has a trust configured with AD, for identifying the IdM domain and services. As a consequence, you must use a different NetBIOS name for the IdM domain than the NetBIOS names used in the AD domains to which you want to establish the forest trust.

The NetBIOS name of an Active Directory or IdM domain is usually the far-left component of the corresponding DNS domain. For example, if the DNS domain is **ad.example.com**, the NetBIOS name is typically **AD**.

NOTE

The maximum length of a NetBIOS name is 15 characters.

5.2.1.4. Firewalls and Ports

To enable communication between AD domain controllers and IdM servers, make sure you meet the following port requirements:

- Open ports required for an AD trust and ports required by an IdM server in an AD trust on IdM servers and all AD domain controllers in both directions: from the IdM servers to the AD domain controllers and back.

- Open the port required by an IdM client in an AD trust on all AD domain controllers of the trusted AD forest. On the IdM clients, make sure the port is open in the outgoing direction (see Prerequisites for Installing a Client in the *Linux Domain Identity, Authentication, and Policy Guide*).

Table 5.2. Ports Required for an AD Trust

Service	Port	Protocol
Endpoint resolution portmapper	135	TCP
NetBIOS-DGM	138	TCP and UDP
NetBIOS-SSN	139	TCP and UDP
Microsoft-DS	445	TCP and UDP
Endpoint mapper listener range	1024-1300	TCP
AD Global Catalog	3268	TCP
LDAP	389	TCP [a] and UDP

[a] The TCP port 389 is not required to be open on IdM servers for trust, but it is necessary for clients communicating with the IdM server.

Table 5.3. Ports Required by IdM Servers in a Trust

Service	Port	Protocol
Kerberos	See Port Requirements in the *Linux Domain Identity, Authentication, and Policy Guide*.	
LDAP		
DNS		

Table 5.4. Ports Required by IdM Clients in an AD Trust

Service	Port	Protocol	Notes
Kerberos	88	UDP and TCP	The **libkrb5** library uses UDP and falls-back to the TCP protocol if the data sent from the Kerberos Distribution Center (KDC) is too large. Active Directory attaches a Privilege Attribute Certificate (PAC) to the Kerberos ticket, which increases the size and requires in most cases to use the TCP protocol. To avoid the fall-back and resending the request, by default, SSSD in Red Hat Enterprise Linux 7.4 and later uses TCP for user authentication. To configure the size before **libkrb5** uses TCP, set the *udp_preference_limit* in the **/etc/krb.5.conf** file. For details, see the krb5.conf(5) man page.

Additional Resources

- For advice on how to open the required ports, see Port Requirements in the *Linux Domain Identity, Authentication, and Policy Guide*.

5.2.1.5. IPv6 Settings

The IdM system must have the IPv6 protocol enabled in the kernel. If IPv6 is disabled, then the CLDAP plug-in used by the IdM services fails to initialize.

5.2.1.6. Clock Settings

Both the Active Directory server and the IdM server must have their clocks in sync.

5.2.1.7. Creating a Conditional Forwarder for the IdM Domain in AD

Prepare the AD DNS server to forward queries for the IdM domain to the IdM DNS server:

1. On a Windows AD domain controller, open the Active Directory (AD) **DNS** console.

2. Right-click **Conditional Forwarders**, select **New Conditional Forwarder**.

3. Enter the IdM DNS domain name and the IP address of the IdM DNS server

4. Select **Store this conditional forwarder in Active Directory, and replicate it as follows**, and select the replication setting that matches your environment.

5. Click **OK**.
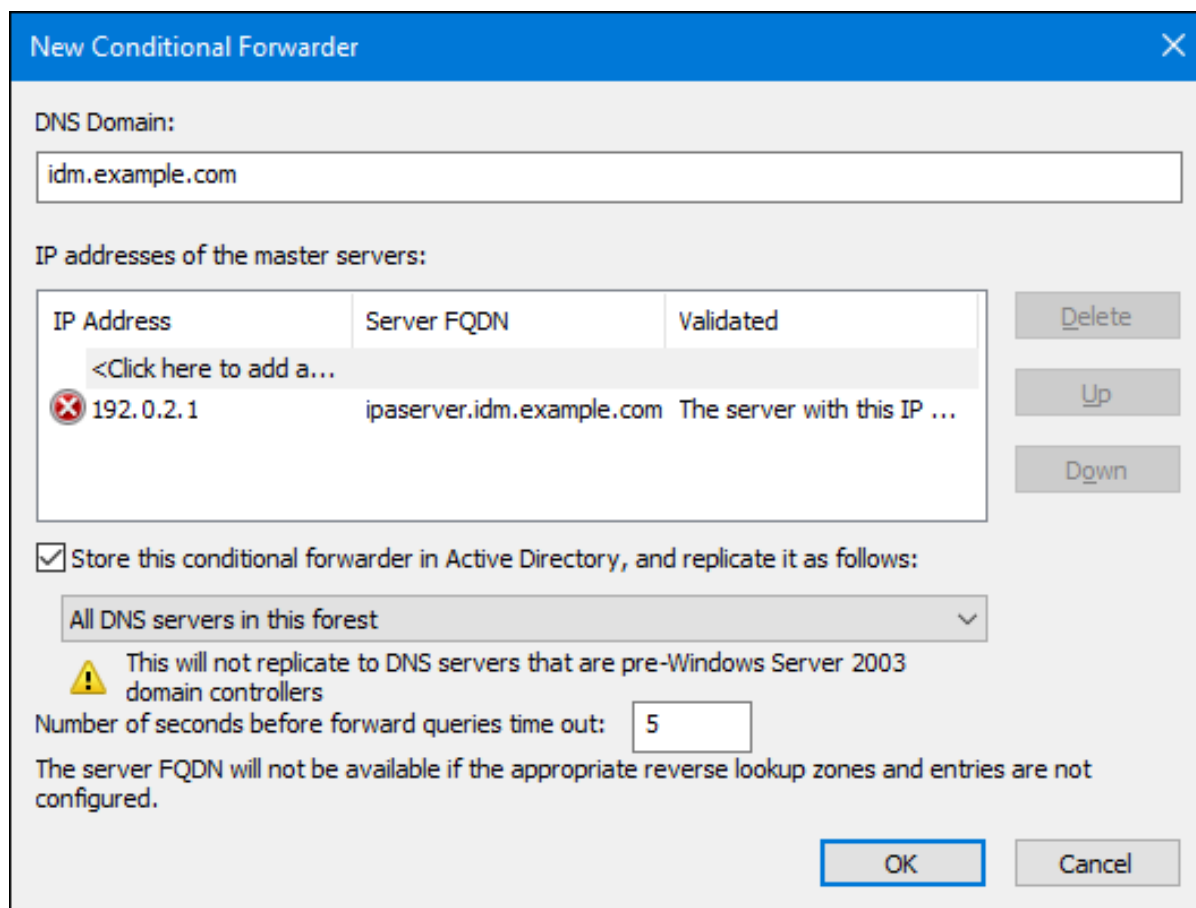
6. To verify that the AD domain controller (DC) can resolve DNS entries from the IdM domain, open a command prompt and enter:

```
C:\> nslookup server.idm.example.com
```

If the command returns the IP address of the IdM server, the conditional forwarder is working correctly.

5.2.1.8. Creating a Forward Zone for the AD Domain in IdM

Prepare the IdM DNS server to forward queries for the AD domain to the AD DNS server:

1. On the IdM server, create a forward zone entry for the AD DNS domain. For further details about creating a DNS forward zone in IdM see the *Configuring Forward Zones* section in the *Linux Domain Identity, Authentication, and Policy Guide* .

2. If the AD DNS server does not support DNSSEC, disable DNSSEC validation on the IdM server:

   a. Edit the **/etc/named.conf** file and set the ***dnssec-validation*** parameter to **no**:

   ```
   dnssec-validation no;
   ```

   b. Restart the **named-pkcs11** service:

   ```
   # systemctl restart named-pkcs11
   ```

3. To verify that the IdM server can resolve DNS entries from the AD domain, enter:

> # host *server.ad.example.com*

If the command returns the IP address of the AD DC, the forward zone is working correctly.

5.2.1.9. Supported User Name Formats

IdM performs user name mapping in the local SSSD client. The default output user name format for users from trusted domains supported by SSSD is **user_name@domain**. Active Directory supports several different kinds of name formats: **user_name**, **user_name@DOMAIN_NAME**, and **DOMAIN_NAME\user_name**.

Users can use either only their user name (**user_name**) or their fully-qualified user name (**user_name@domain_name**), for example, to authenticate to the system.

> ⚠ **WARNING**
>
> Preferably, use the fully-qualified user name to avoid conflicts if the same user name exists in multiple domains.
>
> If a user specifies only the user name with out the domain, SSSD searches the account in all domains configured in the **/etc/sssd/sssd.conf** file and in trusted domains. If you configured a domain resolution order as described in Section 8.5.3, "Configuring the Domain Resolution Order on an IdM Client", SSSD searches for the user in the defined order. In any case, SSSD uses the first entry found. This can lead to problems or confusion if the same user name exists in multiple domains and the first entry found is not the expected one.

By default, SSSD displays user names always in the fully-qualified format. For details about changing the format, see Section 5.5, "Changing the Format of User Names Displayed by SSSD" .

To identify the user name and the domain to which the user name belongs, SSSD uses a regular expression defined in the *re_expression* option. The regular expression is used for IdM back ends or AD back ends and supports all the mentioned formats:

> re_expression = (((?P<domain>[^\\]+)\\(?P<name>.+$))|((?P<name>[^@]+)@(?P<domain>.+$))|(^(?P<name>[^@\\]+)$))

5.2.2. Creating Trusts

The following sections describe creating trusts in various configuration scenarios. Section 5.2.2.1, "Creating a Trust from the Command Line" contains the full procedure for configuring a trust from the command line. The other sections describe the steps which are different from this basic configuration scenario and reference the basic procedure for all other steps.

> **NOTE**
>
> If you set up a replica in an existing trust environment, the replica is not automatically configured as a trust controller. To configure the replica as an additional trust controller, follow the procedures in this section.

After creating a trust, see Section 5.2.3, "Post-installation Considerations for Cross-forest Trusts" .

5.2.2.1. Creating a Trust from the Command Line

Creating a trust relationship between the IdM and Active Directory Kerberos realms involves the following steps:

1. Preparing the IdM server for the trust, described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust"

2. Creating a trust agreement, described in Section 5.2.2.1.2, "Creating a Trust Agreement"

3. Verifying the Kerberos configuration, described in Section 5.2.2.1.3, "Verifying the Kerberos Configuration"

5.2.2.1.1. Preparing the IdM Server for Trust

To set up the IdM server for a trust relationship with AD, follow these steps:

1. Install the required IdM, trust, and Samba packages:

   ```
   [root@ipaserver ]# yum install ipa-server ipa-server-trust-ad samba-client
   ```

2. Configure the IdM server to enable trust services. You can skip this step if you installed the server with the **ipa-replica-install --setup-adtrust** command.

   a. Run the **ipa-adtrust-install** utility:

      ```
      [root@ipaserver ]# ipa-adtrust-install
      ```

      The utility adds DNS service records required for AD trusts. These records are created automatically if IdM was installed with an integrated DNS server.

      If IdM was installed without an integrated DNS server, **ipa-adtrust-install** prints a list of service records that you must manually add to the DNS before you can continue.

      > **IMPORTANT**
      >
      > Red Hat strongly recommends to verify the DNS configuration as described in the section called "Verifying the DNS Configuration" every time after running **ipa-adtrust-install**, especially if IdM or AD do not use integrated DNS servers.

   b. The script prompts to configure the **slapi-nis** plug-in, a compatibility plug-in that allows older Linux clients to work with trusted users.

      ```
      Do you want to enable support for trusted domains in Schema Compatibility plugin?
      This will allow clients older than SSSD 1.9 and non-Linux clients to work with trusted
      users.

      Enable trusted domains support in slapi-nis? [no]: y
      ```

c. At least one user (the IdM administrator) exists when the directory is first installed. The SID generation task can create a SID for any existing users to support the trust environment. This is a resource-intensive task; for a high number of users, this can be run separately.

> Do you want to run the ipa-sidgen task? [no]: yes

3. Make sure that DNS is properly configured, as described in Section 5.2.1.2, "DNS and Realm Settings".

4. Start the **smb** service:

> [root@ipaserver ~]# systemctl start smb

5. Optionally, configure that the **smb** service starts automatically when the system boots:

> [root@ipaserver ~]# systemctl enable smb

6. Optionally, use the **smbclient** utility to verify that Samba responds to Kerberos authentication from the IdM side.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipa.example.com -k
lp_load_ex: changing to config backend registry

Sharename       Type     Comment
---------       ----     -------
IPC$            IPC      IPC Service (Samba 4.9.1)
Reconnecting with SMB1 for workgroup listing.

Server          Comment
---------       -------

Workgroup       Master
---------       -------
```

5.2.2.1.2. Creating a Trust Agreement

Create a trust agreement for the Active Directory domain and the IdM domain by using the **ipa trust-add** command:

> # ipa trust-add --type=*type ad_domain_name* --admin *ad_admin_username* --password

The **ipa trust-add** command sets up a one-way trust by default. It is not possible to establish a two-way trust in RHEL 7.

To establish an external trust, pass the **--external=true** option to the **ipa trust-add** command. See Section 5.1.5, "External Trusts to Active Directory" for details.

> NOTE
>
> The **ipa trust-add** command configures the server as a trust controller by default. See Section 5.1.6, "Trust Controllers and Trust Agents" for details.

The following example establishes a two-way trust by using the **--two-way=true** option:

```
[root@ipaserver ~]# ipa trust-add --type=ad ad.example.com --admin Administrator --password --
two-way=true
Active Directory domain administrator's password:
-------------------------------------------------------
Added Active Directory trust for realm "ad.example.com"
-------------------------------------------------------
  Realm-Name: ad.example.com
  Domain NetBIOS name: AD
  Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
  SID blacklist incoming: S-1-5-20, S-1-5-3, S-1-5-2, S-1-5-1, S-1-5-7, S-1-5-6, S-1-5-5, S-1-5-4, S-1-
5-9, S-1-5-8, S-1-5-17, S-1-5-16, S-1-5-15, S-1-5-14, S-1-5-13, S-1-5-12, S-1-5-11, S-1-5-10, S-1-3,
S-1-2, S-1-1, S-1-0, S-1-5-19,
                 S-1-5-18
  SID blacklist outgoing: S-1-5-20, S-1-5-3, S-1-5-2, S-1-5-1, S-1-5-7, S-1-5-6, S-1-5-5, S-1-5-4, S-1-
5-9, S-1-5-8, S-1-5-17, S-1-5-16, S-1-5-15, S-1-5-14, S-1-5-13, S-1-5-12, S-1-5-11, S-1-5-10, S-1-3,
S-1-2, S-1-1, S-1-0, S-1-5-19,
                 S-1-5-18
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  Trust status: Established and verified
```

5.2.2.1.3. Verifying the Kerberos Configuration

To verify the Kerberos configuration, test if it is possible to obtain a ticket for an IdM user and if the IdM user can request service tickets.

To verify a two-way trust:

1. Request a ticket for an IdM user:

   ```
   [root@ipaserver ~]# kinit user
   ```

2. Request service tickets for a service within the IdM domain:

   ```
   [root@ipaserver ~]# kvno -S host ipaserver.example.com
   ```

3. Request service tickets for a service within the AD domain:

   ```
   [root@ipaserver ~]# kvno -S cifs adserver.example.com
   ```

   If the AD service ticket is successfully granted, there is a cross-realm ticket-granting ticket (TGT) listed with all of the other requested tickets. The TGT is named **krbtgt/_AD.DOMAIN@IPA.DOMAIN_**.

   ```
   [root@ipaserver ]# klist
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: user@IPA.DOMAIN

   Valid starting     Expires           Service principal
   06/15/12 12:13:04  06/16/12 12:12:55  krbtgt/IPA.DOMAIN@IPA.DOMAIN
   06/15/12 12:13:13  06/16/12 12:12:55  host/ipaserver.ipa.example.com@IPA.DOMAIN
   06/15/12 12:13:23 06/16/12 12:12:55 krbtgt/AD.DOMAIN@IPA.DOMAIN
   06/15/12 12:14:58  06/15/12 22:14:58  cifs/adserver.ad.example.com@AD.DOMAIN
   ```

To verify a one-way trust from the IdM side:

1. Request a ticket for an Active Directory user:

   ```
   [root@ipaserver ~]# kinit user@AD.DOMAIN
   ```

2. Request service tickets for a service within the IdM domain:

   ```
   [root@ipaserver ~]# kvno -S host ipaserver.example.com
   ```

   If the AD service ticket is successfully granted, there is a cross-realm ticket-granting ticket (TGT) listed with all of the other requested tickets. The TGT is named **krbtgt/IPA.DOMAIN@AD.DOMAIN**.

   ```
   [root@ipaserver ]# klist
   Ticket cache: KEYRING:persistent:0:krb_ccache_hRtox00
   Default principal: user@AD.DOMAIN

   Valid starting      Expires            Service principal
   03.05.2016 18:31:06  04.05.2016 04:31:01  host/ipaserver.ipa.example.com@IPA.DOMAIN
    renew until 04.05.2016 18:31:00
   03.05.2016 18:31:06 04.05.2016 04:31:01 krbtgt/IPA.DOMAIN@AD.DOMAIN
    renew until 04.05.2016 18:31:00
   03.05.2016 18:31:01  04.05.2016 04:31:01  krbtgt/AD.DOMAIN@AD.DOMAIN
    renew until 04.05.2016 18:31:00
   ```

The **localauth** plug-in maps Kerberos principals to local SSSD user names. This allows AD users to use Kerberos authentication and access Linux services, which support GSSAPI authentication directly.

> **NOTE**
>
> For more information about the plug-in, see Section 5.3.7.2, "Using SSH Without Passwords".

5.2.2.2. Creating a Trust Using a Shared Secret

A shared secret is a password that is known to trusted peers and can be used by other domains to join the trust. The shared secret can configure both one-way and two-way trusts within Active Directory (AD). In AD, the shared secret is stored as a *trusted domain object* (TDO) within the trust configuration.

IdM supports creating a one-way or two-way trust using a shared secret instead of the AD administrator credentials. Setting up such a trust requires the administrator to create the shared secret in AD and manually validate the trust on the AD side.

5.2.2.2.1. Creating a Two-Way Trust Using a Shared Secret

To create a two-way trust with a shared secret with a Microsoft Windows Server 2012, 2012 R2, or 2016:

1. Prepare the IdM server for the trust, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust".

2. If the IdM and AD hosts use a DNS server that cannot resolve both domains, set up forwarding for the DNS zones:

    a. Prepare the AD DNS server to forward queries for the IdM domain to the IdM DNS server. For details, see Section 5.2.1.7, "Creating a Conditional Forwarder for the IdM Domain in AD".

    b. Prepare the IdM DNS server to forward queries for the AD domain to the AD DNS server. For details, see Section 5.2.1.8, "Creating a Forward Zone for the AD Domain in IdM" .

3. Configure a trust in the **Active Directory Domains and Trusts** console. In particular:

   - Create a new trust.

   - Give the trust the IdM domain name, for example **idm.example.com**.

   - Specify that this is a **forest** type of trust.

   - Specify that this is a **two-way** type of trust.

   - Specify that this is a **forest-wide** authentication.

   - Set the **trust password**.

> **NOTE**
>
> The same password must be used when configuring the trust in IdM.

   When asked to confirm the incoming trust, select **No**.

4. Create a trust agreement, as described in Section 5.2.2.1.2, "Creating a Trust Agreement" . When running the **ipa trust-add** command, use the **--type**, **--trust-secret** and **--two-way=True** options, and omit the **--admin** option. For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad ad.example.com --trust-secret --two-way=True
Shared secret for the trust:
-------------------------------------------------------
Added Active Directory trust for realm "ad.example.com"
-------------------------------------------------------
  Realm-Name: ad.example.com
  Domain NetBIOS name: AD
  Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
  SID blacklist incoming: S-1-5-20, S-1-5-3, S-1-5-2, S-1-5-1, S-1-5-7, S-1-5-6,
                  S-1-5-5, S-1-5-4, S-1-5-9, S-1-5-8, S-1-5-17, S-1-5-16,
                  S-1-5-15, S-1-5-14, S-1-5-13, S-1-5-12, S-1-5-11,
                  S-1-5-10, S-1-3, S-1-2, S-1-1, S-1-0, S-1-5-19, S-1-5-18
  SID blacklist outgoing: S-1-5-20, S-1-5-3, S-1-5-2, S-1-5-1, S-1-5-7, S-1-5-6,
                  S-1-5-5, S-1-5-4, S-1-5-9, S-1-5-8, S-1-5-17, S-1-5-16,
                  S-1-5-15, S-1-5-14, S-1-5-13, S-1-5-12, S-1-5-11,
                  S-1-5-10, S-1-3, S-1-2, S-1-1, S-1-0, S-1-5-19, S-1-5-18
  Trust direction: Trusting forest
  Trust type: Active Directory domain
  Trust status: Waiting for confirmation by remote side
```

5. Retrieve the list of domains:

```
[root@ipaserver ~]# ipa trust-fetch-domains ad_domain
```

6. On the IdM server, verify that the trust relationship is established by using the **ipa trust-show** command.

   ```
   [root@ipaserver ~]# ipa trust-show ad.example.com

     Domain NetBIOS name: AD
     Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
     Trust direction: Trusting forest
     Trust type: Active Directory domain
   ```

7. Optionally, search for the trusted domain:

   ```
   [root@ipaserver ~]# ipa trustdomain-find ad.example.com
   Domain name: ad.example.com
   Domain NetBIOS name: AD
   Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
   Domain enabled: True
   ```

8. Verify the Kerberos configuration, as described in Section 5.2.2.1.3, "Verifying the Kerberos Configuration".

5.2.2.2.2. Creating a One-Way Trust Using a Shared Secret

To create a one-way trust using a shared secret with a Microsoft Windows Server 2012, 2012 R2 or 2016:

1. Prepare the IdM server for the trust, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust".

2. If the IdM and AD hosts use a DNS server that cannot resolve both domains, set up forwarding for the DNS zones:

   a. Prepare the AD DNS server to forward queries for the IdM domain to the IdM DNS server. For details, see Section 5.2.1.7, "Creating a Conditional Forwarder for the IdM Domain in AD".

   b. Prepare the IdM DNS server to forward queries for the AD domain to the AD DNS server. For details, see Section 5.2.1.8, "Creating a Forward Zone for the AD Domain in IdM" .

3. Configure a trust in the **Active Directory Domains and Trusts** console:

   a. Right click to the domain name, and select **Properties**.

   b. On the **Trusts** tab, click **New Trust**.

   c. Enter the IdM domain name, and click **Next**.

   d. Select **Forest trust**, and click **Next**.

   e. Select **One-way: incoming**, and click **Next**.

   f. Select **This domain only**, and click **Next**.

   g. Enter a shared secret (trust password), and click **Next**.

   h. Verify the settings, and click **Next**.

    i. When the system asks if you want to confirm the incoming trust, select **No, do not confirm the incoming trust**, and click **Next**.

    j. Click **Finish**.

4. Create a trust agreement:

```
[root@ipaserver ~]# ipa trust-add --type=ad --trust-secret ad.example.com
Shared secret for the trust: password
-------------------------------------------------------
Added Active Directory trust for realm "ad.example.com"
-------------------------------------------------------
  Realm name: ad.example.com
  Domain NetBIOS name: AD
  Domain Security Identifier: S-1-5-21-1762709870-351891212-3141221786
  Trust direction: Trusting forest
  Trust type: Active Directory domain
  Trust status: Waiting for confirmation by remote side
```

Enter the shared secret you set in the AD Domains and Trusts console.

5. Validate the trust in the **Active Directory Domains and Trusts** console:

    a. Right click to the domain name, and select **Properties**.

    b. On the **Trusts** tab, select the domain in the **Domains that trust this domain (incoming trusts) pane**, and click **Properties**.

    c. Click the **Validate** button.

    d. Select **Yes, validate the incoming trust**, and enter the credentials of the IdM *admin* user.

6. Update the list of trusted domains:

```
[root@ipaserver ~]# ipa trust-fetch-domains ad.example.com
```

```
--------------------------------------------------------------------------------
List of trust domains successfully refreshed. Use trustdomain-find command to list them.
--------------------------------------------------------------------------------
---------------------------
Number of entries returned 0
---------------------------
```

7. List the trusted domains:

```
[root@ipaserver ~]# ipa trustdomain-find ad.example.com
  Domain name: ad.example.com
  Domain NetBIOS name: AD
  Domain Security Identifier: S-1-5-21-1762709870-351891212-3141221786
  Domain enabled: True
---------------------------
Number of entries returned 1
---------------------------
```

8. Optionally, verify that the IdM server can retrieve user information from AD domain:

```
[root@ipaserver ~]# getent passwd administrator@ad.example.com
administrator@ad.example.com:*:610600500:610600500:Administrator:/home/ad.example.com/administrator:
```

5.2.2.3. Verifying the ID Mapping

To verify the ID mapping:

1. Run the following command on a Windows Active Directory domain controller (DC) to list the highest ID:

```
C:\> dcdiag /v /test:ridmanager /s:ad.example.com
...
Available RID Pool for the Domain is 1600 to 1073741823
...
```

2. List the ID ranges on an IdM server:

```
[root@ipaserver ~]# ipa idrange-find
----------------
1 range matched
----------------
  Range name: AD.EXAMPLE.COM_id_range
  First Posix ID of the range: 610600000
  Number of IDs in the range: 200000
  First RID of the corresponding RID range: 0
  Domain SID of the trusted domain: S-1-5-21-796215754-1239681026-23416912
  Range type: Active Directory domain range
---------------------------
Number of entries returned 1
---------------------------
```

You require the first POSIX ID value in a later step.

3. On the Active Directory DC, display the security identifier (SID) or a user. For example, to display the SID of **administrator**:

```
C:\> wmic useraccount where name="administrator" get sid
S-1-5-21-796215754-1239681026-23416912-500
```

The last part of the SID is the relative identifier (RID). You require the user's RID in the next step.

> **NOTE**
>
> If the RID is higher than the default ID range (200000), use the **ipa idrange-mod** command to extend the range. For example:
>
> ```
> # ipa idrange-mod --range-size=1000000 AD.EXAMPLE.COM_id_range
> ```

4. Display the user ID of the same user on the IdM server:

```
[root@ipaserver ~]# id ad\\administrator
uid=610600500(administrator@ad.example.com)...
```

5. If you add the first POSIX ID value (610600000) to the RID (500), it must match the user ID displayed on the IdM server (610600500).

5.2.2.4. Creating a Trust on an Existing IdM Instance

When configuring a trust for an existing IdM instance, certain settings for the IdM server and entries within its domain are already configured. However, you must set the DNS configuration for the Active Directory domain and assign Active Directory SIDs to all existing IdM users and groups.

1. Prepare the IdM server for the trust, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust".

2. Create a trust agreement, as described in Section 5.2.2.1.2, "Creating a Trust Agreement" .

3. Generate SIDs for each IdM user.

> **NOTE**
>
> Do not perform this step if the SIDs were generated when the **ipa-adtrust-install** utility was used to establish the trust.

a. Add a new *ipaNTSecurityIdentifier* attribute, containing a SID, automatically for each entry by running the **ipa-sidgen-task** operation on the back-end LDAP directory.

```
[root@ipaserver ]# ldapmodify -x -H ldap://ipaserver.ipa.example.com:389 -D
"cn=directory manager" -w password

dn: cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
```

```
cn: sidgen
nsslapd-basedn: dc=ipadomain,dc=com
delay: 0

adding new entry "cn=sidgen,cn=ipa-sidgen-task,cn=tasks,cn=config"
```

b. After the task completes successfully, a message is recorded in the error logs that the SID generation task (**Sidgen task**) finished with a status of zero (0).

```
[root@ipaserver ]# grep "sidgen_task_thread" /var/log/dirsrv/slapd-IDM-EXAMPLE-
COM/errors
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c, line 191]:
Sidgen task starts ...
[20/Jul/2012:18:17:16 +051800] sidgen_task_thread - [file ipa_sidgen_task.c, line 196]:
Sidgen task finished [0].
```

4. Verify the Kerberos configuration, as described in Section 5.2.2.1.3, "Verifying the Kerberos Configuration".

5.2.2.5. Adding a Second Trust

When adding a trust on an IdM server that already has one or more trust agreements configured, certain general IdM trust settings, such as installing the trust-related packages or configuring SIDs, is no longer required. To add an additional trust, you only must configure DNS and establish a trust agreement.

1. Make sure that DNS is properly configured, as described in Section 5.2.1.2, "DNS and Realm Settings".

2. Create a trust agreement, as described in Section 5.2.2.1.2, "Creating a Trust Agreement" .

5.2.2.6. Creating a Trust in the Web UI

Before creating a trust in the web UI, prepare the IdM server for the trust. This trust configuration is easiest to perform from the command line, as described in Section 5.2.2.1.1, "Preparing the IdM Server for Trust".

Once the initial configuration is set, a trust agreement can be added in the IdM web UI:

1. Open the IdM web UI:

   ```
   https://ipaserver.example.com
   ```

2. Open the **IPA Server** main tab, and select the **Trusts** subtab.

3. In the **Trusts** subtab, click **Add** to open the new trust configuration window.

4. Fill in the required information about the trust:

   a. Provide the AD domain name in the **Domain** field.

   b. To set up the trust as two-way, select the **Two-way trust** check box. To set up the trust as one-way, leave **Two-way trust** unselected.

      For more information about one-way and two-way trusts, see Section 5.1.4, "One-Way and Two-Way Trusts".

c. To establish an external trust to a domain in another forest, select the **External Trust** check box.

For more information, see Section 5.1.5, "External Trusts to Active Directory" .

d. The **Establish using** section defines how the trust is to be established:

- To establish the trust using the AD administrator's user name and password, select **Administrative account** and provide the required credentials.

- Alternatively, to establish the trust with a shared password, select **Pre-shared password** and provide the trust password.

e. Define the ID configuration for the trust:

- The **Range type** option allows you to choose the ID range type. If you want IdM to automatically detect what kind of ID range to use, select **Detect**.

- To define the starting ID of the ID range, use the **Base ID** field. To define the size of the ID range, use the **Range size** field. If you want IdM to use default values for the ID range, do not specify these options.

For more information about ID ranges, see the section called "ID Ranges" .

Figure 5.5. Adding a Trust in the Web UI

5. Click **Add** to save the new trust.

After this, verify the Kerberos configuration, as described in Section 5.2.2.1.3, "Verifying the Kerberos Configuration".

5.2.3. Post-installation Considerations for Cross-forest Trusts

5.2.3.1. Potential Behavior Issues with Active Directory Trust

5.2.3.1.1. Active Directory Users and IdM Administration

Currently, Active Directory (AD) users and administrators can only see their self-service page after logging into the IdM Web UI. AD administrators cannot access the administrator's view of IdM Web UI. For details, see the Authenticating to the IdM Web UI as an AD User section in the *Linux Domain Identity, Authentication, and Policy Guide*.

Additionally, AD users currently cannot manage their own ID overrides. Only IdM users can add and manage ID overrides.

5.2.3.1.2. Authenticating Deleted Active Directory Users

By default, every IdM client uses the SSSD service to cache user identities and credentials. If the IdM or AD back-end provider is temporarily unavailable, SSSD enables the local system to reference identities for users who have already logged in successfully once.

Because SSSD maintains a list of users locally, changes that are made on the back end might not be immediately visible to clients that run SSSD offline. On such clients, users who have previously logged into IdM resources and whose hashed passwords are stored in the SSSD cache are able to log in again even if their user accounts have been deleted in AD.

If the above conditions are met, the user identity is cached in SSSD, and the AD user is able to log into IdM resources even if the user account is deleted AD. This problem will persist until SSSD becomes online and is able to verify AD user logon against AD domain controllers.

If the client system runs SSSD online, the password provided by the user is validated by an AD domain controller. This ensures that deleted AD users are not allowed to log in.

5.2.3.1.3. Credential Cache Collections and Selecting Active Directory Principals

The Kerberos credentials cache attempts to match a client principal to a server principal based on the following identifiers in this order:

1. service name

2. host name

3. realm name

When the client and server mapping is based on the host name or real name and credential cache collections are used, unexpected behavior can occur in binding as an AD user. This is because the realm name of the Active Directory user is different than the realm name of the IdM system.

If an AD user obtains a ticket using the **kinit** utility and then uses SSH to connect to an IdM resource, the principal is not selected for the resource ticket. an IdM principal is used because the IdM principal matches the realm name of the resource.

For example, if the AD user is **Administrator** and the domain is **ADEXAMPLE.ADREALM**, the principal is **Administrator@ADEXAMPLE.ADREALM**.

```
[root@server ~]# kinit Administrator@ADEXAMPLE.ADREALM
Password for Administrator@ADEXAMPLE.ADREALM:
[root@server ~]# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: Administrator@ADEXAMPLE.ADREALM

Valid starting      Expires            Service principal
27.11.2015 11:25:23  27.11.2015 21:25:23
krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
 renew until 28.11.2015 11:25:16
```

This is set as the default principal in the Active Directory ticket cache. However, if any IdM user also has a Kerberos ticket (such as **admin**), then there is a separate IdM credentials cache, with an IdM default

principal. That IdM default principal is selected for a host ticket if the Active Directory user uses SSH to connect to a resource.

```
[root@vm-197 ~]# ssh -l Administrator@adexample.adrealm ipaclient.example.com
Administrator@adexample.adrealm@ipaclient.example.com's password:

[root@vm-197 ~]# klist -A
Ticket cache: KEYRING:persistent:0:0
Default principal: Administrator@ADEXAMPLE.ADREALM

Valid starting      Expires            Service principal
27.11.2015 11:25:23  27.11.2015 21:25:23
krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
 renew until 28.11.2015 11:25:16

Ticket cache: KEYRING:persistent:0:0
Default principal: admin@EXAMPLE.COM >>>>> IdM user

Valid starting      Expires            Service principal
27.11.2015 11:25:18  28.11.2015 11:25:16  krbtgt/EXAMPLE.COM@EXAMPLE.COM
27.11.2015 11:25:48 28.11.2015 11:25:16 host/ipaclient.example.com@EXAMPLE.COM >>>>> host
principal
```

This is because the realm name of the IdM principal matches the realm of the IdM resource.

5.2.3.1.4. Resolving Group SIDs

Losing Kerberos Tickets
Running a command to obtain a SID from the Samba service, such as **net getlocalsid** or **net getdomainsid**, removes any existing admin ticket from the Kerberos cache.

> **NOTE**
>
> You are not required to run commands such as **net getlocalsid** or net **getdomainsid** in order to use Active Directory trusts.

Cannot Verify Group Membership for Users
It is not possible to verify that a specific trusted user is associated with a specific IdM group, external or POSIX.

Cannot Display Remote Active Directory Group Memberships for an Active Directory User
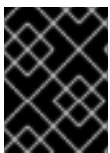
> **IMPORTANT**
>
> Note that this problem no longer occurs if the IdM server and client run on Red Hat Enterprise Linux 7.1 or later.

The **id** utility can be used to display local group associations for Linux system users. However, **id** does not display Active Directory group memberships for Active Directory users, even though Samba tools do display them.

To work around this, you can use the **ssh** utility to log into an IdM client machine as the given AD user. After the AD user logs in successfully for the first time, the **id** search detects and displays the AD group memberships:

```
[root@ipaserver ~]# id ADDOMAIN\user
uid=1921801107(user@ad.example.com) gid=1921801107(user@ad.example.com)
groups=1921801107(user@ad.example.com),129600004(ad_users),1921800513(domain
users@ad.example.com)
```

5.2.3.2. Configuring Trust Agents

After you set up a new replica in a trust environment, the replica does not automatically have the **AD trust agent** role installed. To configure the replica as a trust agent:

1. On an existing trust controller, run the **ipa-adtrust-install --add-agents** command:

   ```
   [root@existing_trust_controller]# ipa-adtrust-install --add-agents
   ```

   The command starts an interactive configuration session and prompts you for the information required to set up the agent.

   For further information about the **--add-agents** option, see the ipa-adtrust-install(1) man page.

2. On the new replica:

   a. Restart the IdM service:

      ```
      [root@new_trust_controller]# ipactl restart
      ```

   b. Remove all entries from the SSSD cache:

      ```
      [root@new_trust_controller]# sssctl cache-remove
      ```

      > **NOTE**
      >
      > To use the **sssctl** command, the sssd-tools package must be installed.

   c. Optionally, verify that the replica has the **AD trust agent** role installed:

      ```
      [root@new_trust_controller]# ipa server-show new_replica.idm.example.com
      ...
      Enabled server roles: CA server, NTP server, AD trust agent
      ```

5.3. MANAGING AND CONFIGURING A CROSS-FOREST TRUST ENVIRONMENT

5.3.1. User Principal Names in a Trusted Domains Environment

IdM supports the logging in using user principal names (UPN). A UPN is an alternative to the user name to authenticate with, and has the format **username@KERBEROS-REALM**. In an Active Directory forest it is possible to configure additional UPN suffixes. These enterprise principal names are used to provide

alternative logins to the default UPN.

For example, if a company uses the Kerberos realm **AD.EXAMPLE.COM**, the default UPN for a user is **user@ad.example.com**. However often a company want instead their users to be able to log in using their email addresses, like **user@example.com**. In this case the administrator adds an additional UPN suffix **example.com** to the Active Directory forest and sets the new suffix in the user's account properties.

UPN suffixes are only visible for IdM when defined in the AD forest root. As an AD administrator, you can define UPNs with the **Active Directory Domain and Trust** utility or the **PowerShell** command line tool.

> **NOTE**
>
> To configure UPN suffixes for users, Red Hat recommends to use tools that perform error validation, such as the **Active Directory Domain and Trust** utility.
>
> Red Hat recommends against configuring UPNs through low-level modifications, such as using **ldapmodify** commands to set the **userPrincipalName** attribute for users, because Active Directory does not validate those operations.

When you add or remove UPN suffixes in a trusted AD forest, you have to refresh the information for the trusted forest on the IdM master:

```
[root@ipaserver ~]# ipa trust-fetch-domains
Realm-Name: ad.example.com
-------------------------------
No new trust domains were found
-------------------------------
----------------------------
Number of entries returned 0
----------------------------
```

Verify that the alternative UPN was fetched, by running:

```
[root@ipaserver ~]# ipa trust-show
Realm-Name: ad.example.com
  Realm-Name: ad.example.com
  Domain NetBIOS name: AD
  Domain Security Identifier: S-1-5-21-796215754-1239681026-23416912
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  UPN suffixes: example.com
```

The UPN suffixes for a domain are stored in the multi-value attribute **ipaNTAdditionalSuffixes** in the **cn=*trusted_domain_name*,cn=ad,cn=trusts,dc=idm,dc=example,dc=com** subtree.

5.3.2. IdM Clients in an Active Directory DNS Domain

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.

> **IMPORTANT**
>
> This is not a recommended configuration and has some limitations. Red Hat recommends to always deploy IdM clients in a DNS zone different from the ones owned by Active Directory and access IdM clients through their IdM host names.

5.3.2.1. Kerberos Single Sign-on to the IdM Client is not Required

For IdM clients set up in the Active Directory DNS domain, only password authentication is available to access resources on this IdM host. To configure the client for this scenario:

1. To ensure that the System Security Service Daemon (SSSD) on the client can communicate with the IdM servers, install the IdM client with the **--domain=*IPA_DNS_Domain*** option:

   ```
   [root@idm-client.ad.example.com ~]# ipa-client-install --domain=idm.example.com
   ```

   This option disables the SRV record auto-detection for the Active Directory DNS domain.

2. Locate the existing mapping for the Active Directory domain in the **[domain_realm]** section of the **/etc/krb5.conf** configuration file:

   ```
   .ad.example.com = IDM.EXAMPLE.COM
   ad.example.com = IDM.EXAMPLE.COM
   ```

   Replace both lines with a mapping entry for the Linux clients fully qualified domain name (FQDN) in the Active Directory DNS zone to the IdM realm:

   ```
   idm-client.ad.example.com = IDM.EXAMPLE.COM
   ```

   Replacing the default mapping prevents Kerberos from sending its requests for the Active Directory domain to the IdM Kerberos Distribution Center (KDC). Instead Kerberos uses auto-discovery through SRV DNS records to locate the KDC. Only for the added host **idm-client.ad.example.com** the IdM KDC is set.

> **NOTE**
>
> Authenticating to resources on clients that are not within an IdM-owned DNS zone is only possible by using user name and password.

Handling of SSL certificates

SSL-based services require a certificate with dNSName extension records that cover all system host names, because both original (A/AAAA) and CNAME records must be in the certificate. Currently, IdM only issues certificates to host objects in the IdM database.

In the described setup without single sign-on available, IdM already has a host object for the FQDN in the database, and **certmonger** can request a certificate for this name:

```
[root@idm-client.ad.example.com ~]# ipa-getcert request -r \
    -f /etc/httpd/alias/server.crt \
    -k /etc/httpd/alias/server.key \
    -N CN=ipa-client.ad.example.com \
```

```
-D ipa-client.ad.example.com \
-K host/idm-client.ad.example.com@IDM.EXAMPLE.COM \
-U id-kp-serverAuth
```

The **certmonger** service uses the default host key stored in the **/etc/krb5.keytab** file to authenticate to the IdM Certificate Authority (CA).

5.3.2.2. Kerberos Single Sign-on to the IdM Client is Required

If you require Kerberos single sign-on to access resources on the IdM client, the client must be within the IdM DNS domain, for example **idm-client.idm.example.com**. You must create a CNAME record **idm-client.ad.example.com** in the Active Directory DNS domain pointing to the A/AAAA record of the IdM client.

For Kerberos-based application servers, MIT Kerberos supports a method to allow the acceptance of any host-based principal available in the application's keytab. To disable the strict checks on what Kerberos principal was used to target the Kerberos server, set the following option in the **[libdefaults]** section of the **/etc/krb5.conf** configuration file:

```
ignore_acceptor_hostname = true
```

Handling of SSL certificates

SSL-based services require a certificate with dNSName extension records that cover all system host names, because both original (A/AAAA) and CNAME records must be in the certificate. Currently, IdM only issues certificates to host objects in the IdM database.

In the described setup without single sign-on available, IdM already has a host object for the FQDN in the database, and **certmonger** can request a certificate for this name:

1. Create a new host object:

   ```
   [root@idm-server.idm.example.com ~]# ipa host-add idm-client.ad.example.com --force
   ```

   Use the **--force** option, because the host name is a CNAME and not an A/AAAA record.

2. Allow the IdM DNS host name to manage the Active Directory host entry in the IdM database:

   ```
   [root@idm-server.idm.example.com ~]# ipa host-add-managedby idm-client.ad.example.com \
        --hosts=idm-client.idm.example.com
   ```

With this setup, the IdM client can request an SSL certificate with dNSName extension record for its host name within the Active Directory DNS domain:

```
[root@idm-client.idm.example.com ~]# ipa-getcert request -r \
    -f /etc/httpd/alias/server.crt \
    -k /etc/httpd/alias/server.key \
    -N CN=`hostname --fqdn` \
    -D `hostname --fqdn` \
    -D idm-client.ad.example.com \
    -K host/idm-client.idm.example.com@IDM.EXAMPLE.COM \
    -U id-kp-serverAuth
```

5.3.3. Creating IdM Groups for Active Directory Users

User groups are required to set access permissions, host-based access control, sudo rules, and other controls on IdM users. These groups are what grant access to IdM domain resources, as well as restricting access.

Both AD users and AD groups can be added directly to IdM user groups. To do that, first add the AD users or groups to a non-POSIX IdM external group and then to a local IdM POSIX group. The POSIX group can then be used for user and role management of the AD users. The principles of handling non-POSIX groups in IdM are described in Section 5.1.3.2, "Active Directory Users and Identity Management Groups".

> **NOTE**
>
> It is also possible to add AD user groups as members to IdM external groups. This might make it easier to define policies for Windows users, by keeping the user and group management within the single AD realm.

1. *Optional.* Create or select the group in the AD domain to use to manage AD users in the IdM realm. Multiple groups can be used and added to different groups on the IdM side.

2. Create an external group in the IdM domain for the Active Directory users by adding the **--external** option to the **ipa group-add** command. The **--external** option indicates that this group is intended to contain members from outside the IdM domain. For example:

```
[root@ipaserver ~]# ipa group-add --desc='AD users external map' ad_users_external --external
-------------------------------
Added group "ad_users_external"
-------------------------------
  Group name: ad_users_external
  Description: AD users external map
```

> **NOTE**
>
> The external group must be linked to a additional group of a user and not to the user's primary group. Active Directory stores group members in ***member*** attributes of a group, and IdM uses this attribute to resolve the members. However, Active Directory stores the primary group of users in the ***primaryGroupID*** attribute in the user's entry, which is not resolved.

3. Create a new IdM POSIX group or select an existing one for administering the IdM policies. For example, to create a new group:

```
[root@ipaserver ~]# ipa group-add --desc='AD users' ad_users
---------------------
Added group "ad_users"
---------------------
  Group name: ad_users
  Description: AD users
  GID: 129600004
```

4. Add the AD users or groups to the IdM external group as an external member. The AD member is identified by its fully-qualified name, such as **DOMAIN\group_name** or **DOMAIN\username**. The AD identity is then mapped to the Active Directory SID for the user or group.

   For example, for an AD group:

   ```
   [root@ipaserver ~]# ipa group-add-member ad_users_external --external "AD\Domain
   Users"
    [member user]:
    [member group]:
     Group name: ad_users_external
     Description: AD users external map
     External member: S-1-5-21-3655990580-1375374850-1633065477-513
   SID_DOM_GROUP (2)
   ------------------------
   Number of members added 1
   ------------------------
   ```

5. Add the external IdM group to the POSIX IdM group as a member. For example:

   ```
   [root@ipaserver ~]# ipa group-add-member ad_users --groups ad_users_external
     Group name: ad_users
     Description: AD users
     GID: 129600004
     Member groups: ad_users_external
   ------------------------
   Number of members added 1
   ------------------------
   ```

5.3.4. Maintaining Trusts

Trust management involves several areas, such as global trust configuration, Kerberos trust configuration, DNS realm configuration, or ID ranges assignment to Active Directory users.

5.3.4.1. Editing the Global Trust Configuration

The **ipa-adtrust-install** utility automatically automatically configures background information for the IdM domain which is required to create a trust with the Active Directory domain.

The global trust configuration contains five attributes:

- A Windows-style security ID (SID); this attribute is autogenerated and cannot be modified

- A domain GUID; this attribute is autogenerated and cannot be modified

- A Kerberos domain name; this attribute comes from the IdM configuration and cannot be modified

- The default group to which to add IdM users; this attribute can be modified

- The NetBIOS name; it is not recommended to modify this attribute

The trust configuration is stored in the **cn=*domain*,cn=ad,cn=etc,dc=example,dc=com** subtree.

5.3.4.1.1. Changing the NetBIOS Name

> **IMPORTANT**
>
> Changing the NetBIOS name in most cases requires to re-establish all existing trusts.
> Therefore, Red Hat recommends not to change the attribute.

A NetBIOS name compatible within an Active Directory topology is configured for the IdM server when running the **ipa-adtrust-install** utility. To change it later, run **ipa-adtrust-install** again and specify the new NetBIOS name using the **--netbios-name** option:

```
[root@ipaserver ]# ipa-adtrust-install --netbios-name=NEWBIOSNAME
```

5.3.4.1.2. Changing the Default Group for Windows Users

When Identity Management is configured to trust an Active Directory forest, an MS-PAC record is added to the Kerberos tickets of IdM users. An MS-PAC record contains security identifiers (SIDs) of the groups to which an IdM user belongs. If the primary group of the IdM user has no SID assigned, the value of the security identifier defined for the *Default SMB Group* will be used. The same logic is applied by the Samba suite when the AD domain controller requests user information from the IdM trust controller.

The Default SMB Group is a fallback group created automatically by the **ipa-adtrust-install** utility. The default group cannot be deleted, but you can use the global trust configuration to specify another IdM group to be used as a fallback for the primary group of the IdM users.

To set the default group from the command line, use the **ipa trustconfig-mod** command:

```
[root@server ~]# kinit admin
[root@server ~]# ipa trustconfig-mod --fallback-primary-group="Example Windows Group"
```

To set the default group from the IdM web UI:

1. Open the IdM web UI.

   ```
   https://ipaserver.example.com
   ```

2. Under the **IPA Server** main tab, select the **Trusts** subtab, and then open the **Global Configuration** section.

3. Select a new group from all of the IdM groups in the **Fallback primary group** drop-down list.
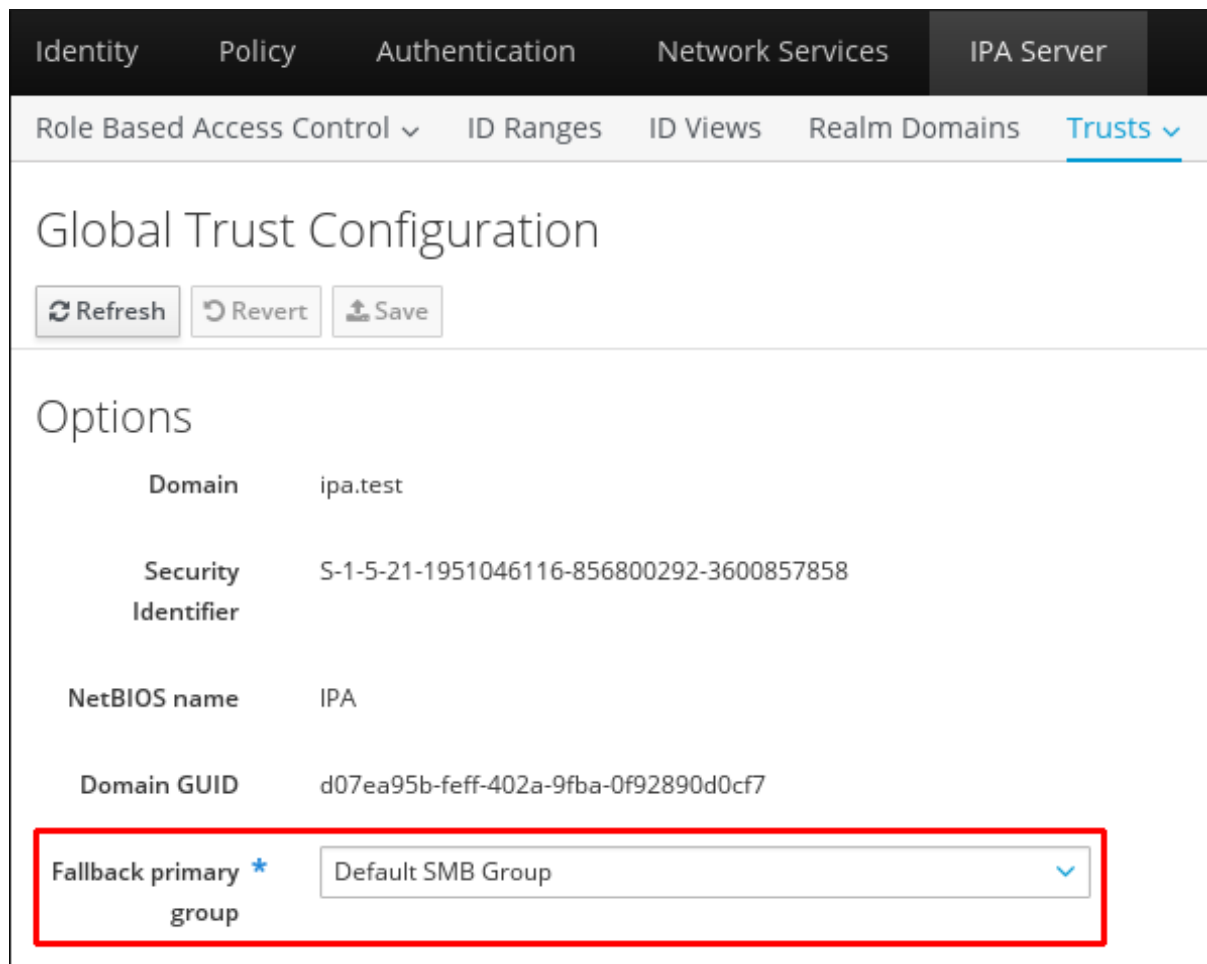
Figure 5.6. Configuring the Default Group for Windows Users

4. Click **Save** to save the new configuration.

5.3.4.2. Discovering, Enabling, and Disabling Trust Domains

A transitive trust means that the trust path can follow a chain of domains. It is described in more detail in Section 5.1.1, "The Architecture of a Trust Relationship".

IdM has a trust with the root domain in a forest and, due to transitivity, all of its child domains and other domains from the same forest are implicitly included in that trust. IdM follows that topology as Windows users from anywhere in the forest attempt to access IdM resources. Each domain and child domain is a *trust domain* in the IdM trust configuration. Each domain is stored in its own entry, **cn=*subdomain*,cn=*trust_name*,cn=ad,cn=trusts,dc=example,dc=com** in the trusts subtree.

IdM attempts to discover and map the full Active Directory topology when the trust is first configured, although in some cases it is required or beneficial to retrieve that topology manually. That is done with the **trust-fetch-domains** command:

```
[root@ipaserver ~]# kinit admin
[root@ipaserver ~]# ipa trust-fetch-domains ad.example.com
---------------------------------------------
List of trust domains successfully refreshed
---------------------------------------------
  Realm name: test.ad.example.com
  Domain NetBIOS name: TEST
  Domain Security Identifier: S-1-5-21-87535643-5658642561-5780864324
```

```
Realm name: users.ad.example.com
Domain NetBIOS name: USERS
Domain Security Identifier: S-1-5-21-91314187-2404433721-1858927112

Realm name: prod.ad.example.com
Domain NetBIOS name: PROD
Domain Security Identifier: S-1-5-21-46580863-3346886432-4578854233
---------------------------
Number of entries returned 3
---------------------------
```

NOTE

When adding a trust with a shared secret, you need to manually retrieve topology of the AD forest. After running the **ipa trust-add ad.domain --trust-secret** command, validate incoming trust at AD side using forest trust properties in the AD Domains and Trusts tool. Then, run the **ipa trust-fetch-domains ad.domain** command. IdM will receive information about the trust, which will then be usable.

Once the topology is retrieved (through automatic or manual discovery), individual domains and child domains in that topology can be enabled, disabled, or removed entirely within the IdM trust configuration.

For example, to disallow users from a specific child domain from using IdM resources, disable that trust domain:

```
[root@ipaserver ~]# kinit admin
[root@ipaserver ~]# ipa trustdomain-disable test.ad.example.com
-----------------------------------------
Disabled trust domain "test.ad.example.com"
-----------------------------------------
```

That trust domain can be re-enabled using the **trustdomain-enable** command.

If a domain should be permanently removed from the topology, than it can be deleted from the IdM trust configuration.

```
[root@ipaserver ~]# kinit admin
[root@ipaserver ~]# ipa trustdomain-del prod.ad.example.com
------------------------------------------------------------------
Removed information about the trusted domain " "prod.ad.example.com"
------------------------------------------------------------------
```

5.3.4.3. Viewing and managing domains associated with IdM Kerberos realm

Domains that are associated with the IdM Kerberos realm are stored in the **cn=Realm Domains,cn=ipa,cn=etc,dc=example,dc=com** subtree in the IdM directory. The list of domains is used by IdM when it establishes a trust with Active Directory. Knowing the full list of domains managed by IdM allows the AD domain controller to know which authentication requests to route to the IdM KDC. The list of configured domains associated with IdM realm can be displayed using the **realmdomains-show** command:

```
[root@ipaserver ~]# kinit admin
[root@ipaserver ~]# ipa realmdomains-show
Domain: ipa.example.org, ipa.example.com, example.com
```

In an IdM setup with integrated DNS:

- A domain is automatically added to the domains list after a new DNS zone is added to IdM using the **ipa dnszone-add** command. Running **ipa realmdomains-show** shows the new domain in the list of domains controlled by the IdM KDC:

  ```
  # kinit admin
  # ipa dnszone-add ipa2.example.com
  # ipa realmdomains-show
  Domain: ipa.example.org, ipa.example.com, example.com, ipa2.example.com
  ```

  Deletion and other types of modification of domains associated with the IdM Kerberos realm are also taken care of automatically.

In an IdM setup without integrated DNS:

- If a DNS zone has been added that is part of the IdM Kerberos realm, the new domain has to be added manually to the IdM list of domains under the control of the IdM KDC. Add the new domain using the **ipa realmdomains-mod** command with the **--add-domain** option:

  ```
  [root@ipaserver ~]# kinit admin
  [root@ipaserver ~]# ipa realmdomains-mod --add-domain=ipa2.example.com
  Domain: ipa.example.org, ipa.example.com, example.com, ipa2.example.com
  ```

  If a DNS zone has been deleted, you need to delete the domain associated with the IdM Kerberos realm manually, too:

  ```
  [root@ipaserver ~]# kinit admin
  [root@ipaserver ~]# ipa realmdomains-mod --del-domain=ipa2.example.com
  Domain: ipa.example.org, ipa.example.com, example.com
  ```

  If there are multiple changes to be made to the list of domains, the list itself can be modified and replaced using the **--domain** option.

  ```
  [root@ipaserver ~]# ipa realmdomains-mod --domain={ipa.example.org,ipa2.example.com}
  ```

5.3.4.4. Adding Ranges for UID and GID Numbers in a Transitive Trust

Creating ID ranges at the time when a trust is originally configured is described in the section called "ID Ranges". To add an ID range later, use the **ipa idrange-add** command with the following options:

- the **--base-id** option sets the base ID for the POSIX range, which is the starting number

- The **--range-size** option sets the size of the POSIX ID range IdM uses. IdM maps the RID of users and groups in a trusted AD domain to POSIX IDs. The **--range-size** option defines the maximum number of IDs IdM creates. AD uses a new RID for each user and group you create. If you delete a user or group, AD will not re-use the RID for future AD entries. Therefore, the range must be large enough for IdM to assign an ID to each existing AD user and group as well as the ones you create in the future. For example, if an administrator deletes 20000 of 50000 AD users and will, during the time, create 10000 new accounts, the range must be at least set to

60000. However, it is important that the range also contains enough reserves. In large environments, in which you expect that the default (200000) range size is not sufficient, set **--range-size** to a higher value.

- the **--rid-base** option sets the starting number of the RID, which is the far-right number in the SID; the value represents the range to add to the base ID to prevent conflicts

- the **--dom-sid** option sets the domain SID, because there can be multiple domains configured for trusts

In the following example, the base ID is 1,200,000 and the RID is 1,000. The resulting ID number is then 1,201,000.

```
[root@server ~]$ kinit admin
[root@server ~]$ ipa idrange-add --base-id=1200000 --range-size=200000 --rid-base=0 --dom-sid=S-1-5-21-123-456-789 trusted_dom_range
```

> **IMPORTANT**
>
> Make sure that the manually defined ID range does not overlap with the ID range used by IdM.

5.3.4.5. Adjusting DNA ID ranges manually

In some cases, you may need to manually adjust Distributed Numeric Assignment (DNA) ID ranges for existing replicas, for example to recover a DNA ID range assigned to a non-functioning replica or to extend a range that has run out of IDs.

When adjusting a DNA ID range manually, make sure that the newly adjusted range is included in the IdM ID range. You can check this using the **ipa idrange-find** command. If the newly adjusted range is not included in the IdM ID range, the command fails.

To recover a DNA ID range from a non-functioning replica, use the **ipa-replica-manage dnarange-show** command to see currently assigned DNA ranges. To see the currently assigned on-deck DNA ranges, use the **ipa-replica-manage dnanextrange-show** command.

> **IMPORTANT**
>
> Do not create overlapping ID ranges. If any of the ID ranges you assign to servers or replicas overlap, it could result in two different servers assigning the same ID value to different entries.

To define the current DNA ID range for a specified server, use the **ipa-replica-manage dnarange-set** command:

```
# ipa-replica-manage dnarange-set masterA.example.com 1250-1499
```

To define the next DNA ID range for a specified server, use the **ipa-replica-manage dnanextrange-set** command:

```
# ipa-replica-manage dnanextrange-set masterB.example.com 1500-5000
```

5.3.4.6. Kerberos Flags for Services and Hosts

Accessing services or hosts in a trusted domain can require special flags for the Kerberos ticket-granting ticket (TGT). For example, if you want to log in using single sign-on to an IdM client with an Active Directory (AD) account from an AD client, the Kerberos TGT flag **OK_AS_DELEGATE** is required.

For more information and how to set Kerberos flags, see Kerberos Flags for Services and Hosts in the *Linux Domain Identity, Authentication, and Policy Guide* .

5.3.5. Setting PAC Types for Services

On IdM resources, if an Active Directory user requests a ticket for a service, then IdM forwards the request to Active Directory to retrieve the user information. Access data, associated with the Active Directory group assignments for the user, is sent back by Active Directory and embedded in the Kerberos ticket.

Group information in Active Directory is stored in a list of identifiers in each Kerberos ticket for Active Directory users in a special data set called *privileged access certificates* or MS-PAC. The group information in the PAC has to be mapped to the Active Directory groups and then to the corresponding IdM groups to help determine access.

IdM services can be configured to generate PACs for each authentication request when a user first attempts to authenticate to a domain service.

5.3.5.1. Setting Default PAC Types

The IdM server configuration defines which PAC types are generated by default for a service. The global settings can be overridden by changing the local settings on a specific service.

1. Open the **IPA Server** tab.

2. Select the **Configuration** subtab.

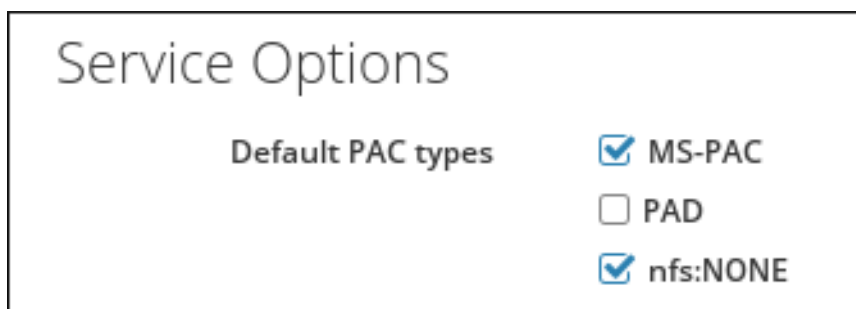3. Scroll to the **Service Options** area.

Figure 5.7. The **Service Options** Area

4. To use PAC, select the **MS-PAC** check box, which adds a certificate that can be used by AD services. If no check box is selected, then no PAC is added to Kerberos tickets.

   If you select the **nfs:NONE** check box, the MS-PAC record will not be added to the service tickets issued against NFS servers.

> **NOTE**
>
> You can ignore the **PAD** check box. This functionality is not yet available in IdM.

5. Click the **Update** link at the top of the page to save the changes.

5.3.5.2. Setting PAC Types for a Service

The global policy sets what PAC types to use for a service if nothing is set explicitly for that service. However, the global settings can be overridden on the local service configuration.

To change the PAC setting from the command line, use the **ipa service-mod** command with the **--pac-type** option. For information on how to use the command, run it with the **--help** option added:

```
$ ipa service-mod --help
Usage: ipa [global-options] service-mod PRINCIPAL [options]

Modify an existing IPA service.
Options:
-h, --help          show this help message and exit
...
```

To change the PAC setting in the web UI:

1. Open the **Identity** tab, and select the **Services** subtab.

2. Click the name of the service to edit.

3. In the **Service Settings** area, check the **Override inherited settings** option and then select the **MS-PAC** check box to add a certificate that can be used by AD services.
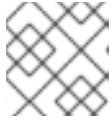

Figure 5.8. The Service Settings Area

If no check box is selected, then no PACs are added to Kerberos tickets.

> **NOTE**
>
> You can ignore the **PAD** check box. This functionality is not yet available in IdM.

4. Click the **Update** link at the top of the page to save the changes.

5.3.6. Using POSIX Attributes Defined in Active Directory

5.3.6.1. Defining UID and GID Attributes for Active Directory Users

If the Windows administrator manually defines POSIX UID and GID attributes for a user, create a matching group on the IdM server with the same GID for the user.

Creating the group ensures that the user is associated with a primary user group. If such group does not exist, the IdM server is unable to look up all groups to which the user belongs.

5.3.6.2. Transferring Login Shell and Home Directory Attributes

> **IMPORTANT**
>
> The client must be enrolled with an IdM server based on Red Hat Enterprise Linux 7.1 or later to benefit from this functionality.

SSSD is able to read the following attribute values from an Active Directory server in a trust relationship with IdM:

- the **loginShell** attribute, which specifies the AD user's shell
- the **unixHomeDirectory** attribute, which specifies the AD user's home directory

When a custom shell or home directory value is defined on the AD server using these attributes, the custom value is then displayed to the IdM client for the AD user. Therefore, the same user shell is displayed for the AD user both on the AD side and on the IdM side.

Note that to display the AD user's home directory to the IdM client, the ***subdomain_homedir*** option in the **[domain]** section of the **/etc/sssd/sssd.conf** file on the IdM server must be set to **%o**. The **%o** value represents the home directory retrieved from the identity provider. For example:

```
[domain/example.com]
subdomain_homedir = %o
```

If the AD administrator modifies **loginShell** or **unixHomeDirectory** on the AD side, the change is automatically reflected on the IdM side as well. If the attributes are not defined on the AD server, SSSD uses a template default value. This default value is then displayed to the IdM client.

5.3.7. Using SSH from Active Directory Machines for IdM Resources

When a trust is configured, Active Directory users can access machines, services, and files on IdM hosts using SSH and their AD credentials.

5.3.7.1. Caching Considerations

IdM clients do not connect to Active Directory domain controllers (DC) directly to retrieve user

attributes. Instead, a client connects to an IdM server who caches this information. For this reason, if you disable a user in Active Directory, the user can still authenticate to IdM clients using SSH key authentication until the record of the user expires in the IdM database.

IdM updates a record of a user in the following situations:

- The entry has expired automatically.

- You manually expire the entry of the user in the cache using the **sss_cache** utility:

  ```
  # sss_cache --user user_name
  ```

- The user authenticates to an IdM server using the **kinit** utility or the web UI.

5.3.7.2. Using SSH Without Passwords

The **localauth** Kerberos plug-in for local authorization ensures that Kerberos principals are automatically mapped to local SSSD user names. With **localauth**, Windows users from a trusted AD domain are not prompted for a password when logging in using Kerberos and can therefore use SSH without passwords.

The plug-in provides a reliable mapping mechanism across multiple realms and trusts: when **sssd** connects to the Kerberos library to map the principal to a local POSIX identity, the SSSD plug-in maps them according to the trust agreements defined in IdM.

In certain situations, users use an SSH bastion host to access other Red Hat Enterprise Linux machines. By default, if you use Kerberos to authenticate to SSH on the bastion host, the Kerberos ticket cannot be forwarded to authenticate using Kerberos to other Red Hat Enterprise Linux hosts. To enable such forward authentication, add the **OK_AS_DELEGATE** Kerberos flag to the bastions host principal:

```
# ipa host-mod bastion_host.idm.example.com --ok-as-delegate=true
```

Kerberos Authentication for AD Users on Red Hat Enterprise Linux 7.1 and newer Systems
In Red Hat Enterprise Linux 7.1 and newer systems, SSSD automatically configures the **localauth** Kerberos plug-in.

SSSD allows user names in the format **user@AD.DOMAIN**, **ad.domain\user** and **AD\user**.

> **NOTE**
>
> On systems with **localauth**, it is not required to set the *auth_to_local* option in the **/etc/krb5.conf** file or list Kerberos principals in the **.k5login** file. The **localauth** plug-in makes this previously used configuration for logins without passwords obsolete.

Manual Configuration of Kerberos Authentication for AD Users
On systems where the **localauth** plug-in is not present, SSH prompts for a user password for Active Directory domain users even if the user obtains a proper Kerberos ticket.

To enable Active Directory users to use Kerberos for authentication in this situation, configure the *auth_to_local* option in the **/etc/krb5.conf** file or list the user Kerberos principals in the **.k5login** file in the home directory of the user.

Configuring /etc/krb5.conf
The following procedure describes how to configure realm mapping in the Kerberos configuration.

1. Open the **/etc/krb5.conf** file.

2. In the **[realms]** section, identify the IdM realm by name, and then add two *auth_to_local* lines to define the Kerberos principal name mapping:

   - In one rule, include a rule to map different Active Directory user name formats and the specific Active Directory domain.

   - In the other rule, set the value of **DEFAULT**, for standard Unix user names.

   For example:

   ```
   [realms]
   IDM = {

   ....
   auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
   auth_to_local = DEFAULT
   }
   ```

3. Restart the KDC service.

   ```
   [root@server ~]# systemctl restart krb5kdc.service
   ```

Note that if you configure Kerberos authentication using the *auth_to_local* option, the user name used for SSH access must meet the following criteria:

- The user name must have the format **ad_user@ad_domain**.

- The domain name must be lowercase.

- The case of the user name must match the case of the user name in Active Directory. For example, **user** and **User** are considered different users because of the different cases.

For more information about setting *auth_to_local*, see the krb5.conf(5) man page.

Configuring **.k5login**

The following procedure configures the system to find the Kerberos principal name for a local user name.

1. Create the **.k5login** file in the user's home directory.

2. List the Kerberos principals used by the user in the file.

If the authenticating user matches the principal in an existing Kerberos ticket, the user is allowed to log in using the ticket and is not prompted for a password.

Note that if you configure Kerberos authentication using the **.k5login** configuration, the user name used for SSH access must have the format **ad_user@ad_domain**.

For more information about configuring the **.k5login** file, see the .k5login(5) man page.

Either one of these configuration procedures results in AD users being able to log in using Kerberos.

5.3.8. Using a Trust with Kerberos-enabled Web Applications

Any existing web application can be configured to use Kerberos authentication, which references the trusted Active Directory and IdM Kerberos realms. For the full Kerberos configuration directives, see the Configuration page for the mod_auth_kerb module .

> **NOTE**
>
> After changing the Apache application configuration, restart the Apache service:
>
> ```
> [root@ipaserver ~]# systemctl restart httpd.service
> ```

For example, for an Apache server, there are several options that define how the Apache server connects to the IdM Kerberos realm:

KrbAuthRealms

The **KrbAuthRealms** option gives the application location to the name of the IdM domain. This is required.

Krb5Keytab

The **Krb5Keytab** option gives the location for the IdM server keytab. This is required.

KrbServiceName

The **KrbServiceName** option sets the Kerberos service name used for the keytab (HTTP). This is recommended.

KrbMethodK5Passwd and **KrbMethodNegotiate**

The **KrbMethodK5Passwd** Kerberos method option enables password-based authentication for valid users. The **KrbMethodNegotiate** option enables single sign-on (SSO) if a valid Kerberos ticket is available.

These options are recommended for ease of use for many users.

KrbLocalUserMapping

The **KrbLocalUserMapping** option enables normal web logins (which are usually the UID or common name of the account) to be mapped to the fully-qualified user name (which has a format of *user@REALM.COM*).

This option is strongly recommended. Without the domain name/login name mapping, the web login appears to be a different user account than the domain user. This means that users cannot see their expected data.

For information on supported user name formats, see Section 5.2.1.9, "Supported User Name Formats".

Example 5.1. Kerberos Configuration in an Apache Web Application

```
<Location "/mywebapp">
    AuthType Kerberos
    AuthName "IPA Kerberos authentication"
    KrbMethodNegotiate on
    KrbMethodK5Passwd on
```

```
    KrbServiceName HTTP
    KrbAuthRealms IDM_DOMAIN
    Krb5Keytab /etc/httpd/conf/ipa.keytab
    KrbLocalUserMapping on
    KrbSaveCredentials off
    Require valid-user
</Location>
```

5.3.9. Configuring an IdM server as a Kerberos Distribution Center Proxy for Active Directory Kerberos communication

In certain situations, network restrictions or firewall rules prevent Identity Management (IdM) clients from sending Kerberos traffic to port 88 on Active Directory (AD) domain controllers. The solution is to set up a Kerberos proxy, for instance on an Identity Management server, to relay traffic from IdM clients to AD.

1. On IdM clients, add the Active Directory realm to the [realms] section of the **/etc/krb5.conf** file. Set the **kdc** and **kpasswd_server** parameters to point to the IdM server's fully qualified domain name followed by /**KdcProxy**':

   ```
   AD.EXAMPLE.COM = {
       kdc = https://server.idm.example.com/KdcProxy
       kpasswd_server = https://server.idm.example.com/KdcProxy
     }
   ```

2. On IdM clients, disable the creation of **/var/lib/sss/pubconf/kdcinfo.*** files which could override the /**etc/krb5.conf** specifications from the previous step. Edit the /**etc/sssd/sssd.conf** file, setting the **krb5_use_kdcinfo** to **False**:

   ```
   [domain/example.com]
   krb5_use_kdcinfo = False
   ```

3. On IdM servers, set the **use_dns** option to **true** in the /**etc/ipa/kdcproxy/kdcproxy.conf** file to utilize DNS service (SRV) records to find AD servers to communicate with:

   ```
   use_dns = true
   ```

   Alternatively, if you do not want to use DNS SRV records, add explicit AD servers to the [realms] section of the /**etc/krb5.conf** file:

   ```
   AD.EXAMPLE.COM = {
       kdc = ad-server.ad.example.com
       kpasswd_server = ad-server.ad.example.com
     }
   ```

   NOTE

   You can perform steps 2 and 3 of the procedure by running a script, for example an Ansible script. This is especially useful when making changes on multiple systems.

4. On IdM servers, restart IPA services:

```
# ipactl restart
```

5. To verify that the procedure has been successful, run the following on an IdM client:

```
# rm /var/lib/sss/pubconf/kdcinfo*
# kinit ad_user@AD.EXAMPLE.COM
Password for ad_user@AD.EXAMPLE.COM:
# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: ad_user@AD.EXAMPLE.COM


Valid starting     Expires          Service principal
[... output truncated ...]
```

5.4. CHANGING THE LDAP SEARCH BASE FOR USERS AND GROUPS IN A TRUSTED ACTIVE DIRECTORY DOMAIN

As an administrator, you can set a different search base for users and groups in the trusted Active Directory domain. For example, this enables you to filter out users from inactive organizational units so that only active Active Directory users and groups are visible to the SSSD client system.

5.4.1. Prerequisites

- To ensure that SSSD does not resolve all groups the users belongs to, consider disabling the support for the **tokenGroups** attribute on the Active Directory side.

  With **tokenGroups** enabled, SSSD resolves all groups the user belongs to because the attribute contains a flat list of SIDs. See Token-Groups attribute on Microsoft Developer Network for details about the attribute.

5.4.2. Configuring the LDAP Search Base to Restrict Searches

This procedure describes restricting searches in SSSD to a specific subtree by editing the /**etc**/**sssd**/**sssd.conf** file.

Considerations

- If your SSSD clients are directly joined to an Active Directory domain, perform this procedure on all the clients.

- If your SSSD clients are in an Identity Management domain that is in a trust with Active Directory, perform this procedure only on the Identity Management server.

Procedure

1. Make sure the trusted domain has a separate **[domain]** section in **sssd.conf**. The headings of trusted domain sections follow this template:

```
[domain/main_domain/trusted_domain]
```

For example:

[domain/*idm.example.com*/*ad.example.com*]

2. Edit the **sssd.conf** file to restrict the search base to a specific organizational unit (OU). For example, the *ldap_search_base* option changes the search base for all types of objects.

[domain/*idm.example.com*/*ad.example.com*]
ldap_search_base = ou=*finance*,dc=*ad*,dc=*example*,dc=*com*

You can also use the *ldap_user_search_base*, *ldap_group_search_base*, *ldap_netgroup_search_base*, and *ldap_service_search_base* options. For more details on these options, see the sssd-ldap(5) man page.

3. Restart SSSD.

systemctl restart sssd.service

4. To verify, resolve a few Active Directory users on the SSSD client. For example, to test a change to the user search base and group search base:

getent passwd *ad_user@ad.example.com*
getent group *ad_group@ad.example.com*

If SSSD is configured correctly, you are able to resolve only objects from the configured search base.

If you are able to resolve users from other search domains, troubleshoot the problem by inspecting the SSSD logs:

1. Expire the SSSD caches.

sss_cache --everything

2. In the general **[domain]** section of **sssd.conf**, set the *debug_level* option to **9**.

3. Repeat the command for resolving a user.

4. In the SSSD logs at **/var/log/sssd**/, look for messages from the **sdap_get_generic_*** functions. The functions log the filter and search base used in user searches.

Additional Resources

- For a list of options you can use in trusted domain sections of **sssd.conf**, see **TRUSTED DOMAIN SECTION** in the sssd.conf(5) man page.

5.5. CHANGING THE FORMAT OF USER NAMES DISPLAYED BY SSSD

By default, SSSD uses the **user_name@domain_name** format when displaying user names. Before you change the format, see Section 5.2.1.9, "Supported User Name Formats" to learn about the reason of this default value.

To configure that SSSD displays only the user name without domain:

1. Add the following entry to the domain's section in the **/etc/sssd/sssd.conf** file:

```
full_name_format = %1$s
```

2. Restart SSSD:

```
# systemctl restart sssd
```

5.6. RESTRICTING IDENTITY MANAGEMENT OR SSSD TO SELECTED ACTIVE DIRECTORY SERVERS OR SITES IN A TRUSTED ACTIVE DIRECTORY DOMAIN

As an administrator, you can disable autodiscovery of Active Directory servers and sites in the trusted Active Directory domain and instead list servers, sites, or both manually, so that you can limit the list of Active Directory servers that SSSD communicates with. For example, this enables you to avoid contacting sites that are not accessible.

5.6.1. Configuring SSSD to Contact a Specific Active Directory Server

This procedure describes manually setting Active Directory servers that SSSD connects to by editing the **/etc/sssd/sssd.conf** file.

Considerations

- If your SSSD clients are directly joined to an Active Directory domain, perform this procedure on all the clients.

  In this setup, restricting the Active Directory domain controllers (DCs) or sites also configures the SSSD clients to connect to a particular server or site for authentication.

- If your SSSD clients are in an Identity Management domain that is in a trust with Active Directory, perform this procedure only on the Identity Management server.

  In this setup, restricting the Active Directory DCs or sites does not configure the Identity Management clients to connect to a particular server or site for authentication. Although trusted Active Directory users and groups are resolved through Identity Management servers, authentication is performed directly against the Active Directory DCs. Starting with Red Hat Enterprise Linux 7.6 and sssd-1.16.2-5.el7, you can configure SSSD on IdM clients to use a specific AD server or site using the *ad_server* and *ad_site* options. In prior versions of Red Hat Enterprise Linux 7, restrict authentication by defining the required Active Directory DCs in the **/etc/krb5.conf** file on the clients.

Procedure

1. Make sure the trusted domain has a separate **[domain]** section in **sssd.conf**. The headings of trusted domain sections follow this template:

   ```
   [domain/main_domain/trusted_domain]
   ```

   For example:

   ```
   [domain/idm.example.com/ad.example.com]
   ```

2. Edit the **sssd.conf** file to list the host names of the Active Directory servers or sites to which you want SSSD to connect.

Use the **ad_server** and, optionally, **ad_backup_server** options for Active Directory servers. Use the **ad_site** option for Active Directory sites. For more details on these options, see the sssd-ad(5) man page.

For example:

> [domain/*idm.example.com*/*ad.example.com*]
> ad_server = *dc1.ad.example.com*

3. Restart SSSD.

> # systemctl restart sssd.service

4. To verify, on the SSSD client, resolve or authenticate as an Active Directory user from the configured server or site. For example:

> # id *ad_user@ad.example.com*

If you are unable to resolve the user or authenticate, use these steps to troubleshoot the problem:

1. In the general **[domain]** section of **sssd.conf**, set the **debug_level** option to **9**.

2. Inspect the SSSD logs at **/var/log/sssd/** to see which servers SSSD contacted.

Additional Resources

- For a list of options you can use in trusted domain sections of **sssd.conf**, see **TRUSTED DOMAIN SECTION** in the sssd.conf(5) man page.

5.7. ACTIVE DIRECTORY TRUST FOR LEGACY LINUX CLIENTS

Linux clients running Red Hat Enterprise Linux with SSSD version 1.8 or earlier (*legacy clients*) do not provide native support for IdM cross-forest trusts with Active Directory. Therefore, for AD users to be able to access services provided by the IdM server, the legacy Linux clients and the IdM server have to be properly configured.

Instead of using SSSD version 1.9 or later to communicate with the IdM server to obtain LDAP information, legacy clients use other utilities for this purpose, for example **nss_ldap**, **nss-pam-ldapd**, or SSSD version 1.8 or earlier. Clients running the following versions of Red Hat Enterprise Linux do not use SSSD 1.9 and are therefore considered to be legacy clients:

- Red Hat Enterprise Linux 5.7 or later

- Red Hat Enterprise Linux 6.0 – 6.3

IMPORTANT

Do not use the configuration described in this section for non-legacy clients, that is, clients running SSSD version 1.9 or later. SSSD 1.9 or later provides native support for IdM cross-forest trusts with AD, meaning AD users can properly access services on IdM clients without any additional configuration.

When a legacy client joins the domain of an IdM server in a trust relationship with AD, a *compat LDAP tree* provides the required user and group data to AD users. However, the compat tree enables the AD users to access only a limited number of IdM services.

Legacy clients *do not* provide access to the following services:

- Kerberos authentication

- host-based access control (HBAC)

- SELinux user mapping

- **sudo** rules

Access to the following services *is provided* even in case of legacy clients:

- information look-up

- password authentication

5.7.1. Server-side Configuration for AD Trust for Legacy Clients

Make sure the IdM server meets the following configuration requirements:

- The ipa-server package for IdM and the ipa-server-trust-ad package for the IdM trust add-on have been installed.

- The **ipa-server-install** utility has been run to set up the IdM server.

- The **ipa-adtrust-install --enable-compat** command has been run, which ensures that the IdM server supports trusts with AD domains and that the compat LDAP tree is available.

  If you have already run **ipa-adtrust-install** without the **--enable-compat** option in the past, run it again, this time adding **--enable-compat**.

- The **ipa trust-add** *ad.example.org* command has been run to establish the AD trust.

If the host-based access control (HBAC) **allow_all** rule is disabled, enable the **system-auth** service on the IdM server, which allows authentication of the AD users.

You can determine the current status of **allow_all** directly from the command line using the **ipa hbacrule-show** command. If the rule is disabled, **Enabled: FALSE** is displayed in the output:

```
[user@server ~]$ kinit admin
[user@server ~]$ ipa hbacrule-show allow_all
  Rule name: allow_all
  User category: all
  Host category: all
  Service category: all
  Description: Allow all users to access any host from any host
  Enabled: FALSE
```

> **NOTE**
>
> For information on disabling and enabling HBAC rules, see Configuring Host-Based Access Control in the *Linux Domain Identity, Authentication, and Policy Guide* .

To enable **system-auth** on the IdM server, create an HBAC service named **system-auth** and add an HBAC rule using this service to grant access to IdM masters. Adding HBAC services and rules is described in Configuring Host-Based Access Control section in the *Linux Domain Identity, Authentication, and Policy Guide*. Note that HBAC services are PAM service names; if you add a new PAM service, make sure to create an HBAC service with the same name and then grant access to this service through HBAC rules.

5.7.2. Client-side Configuration Using the ipa-advise Utility

The **ipa-advise** utility provides the configuration instructions to set up a legacy client for an AD trust.

To display the complete list of scenarios for which **ipa-advise** can provide configuration instructions, run **ipa-advise** without any options. Running **ipa-advise** prints the names of all available sets of configuration instructions along with the descriptions of what each set does and when it is recommended to be used.

```
[root@server ~]# ipa-advise
config-redhat-nss-ldap  : Instructions for configuring a system
      with nss-ldap as a IPA client.
      This set of instructions is targeted
      for platforms that include the
      authconfig utility, which are all
      Red Hat based platforms.
config-redhat-nss-pam-ldapd : Instructions for configuring a system
(...)
```

To display a set of instructions, run the **ipa-advise** utility with an instruction set as a parameter:

```
[root@server ~]# ipa-advise config-redhat-nss-ldap
#!/bin/sh
# ---------------------------------------------------------------------
# Instructions for configuring a system with nss-ldap as a IPA client.
# This set of instructions is targeted for platforms that include the
# authconfig utility, which are all Red Hat based platforms.
# ---------------------------------------------------------------------
# Schema Compatibility plugin has not been configured on this server. To
# configure it, run "ipa-adtrust-install --enable-compat"
# Install required packages via yum
yum install -y wget openssl nss_ldap authconfig

# NOTE: IPA certificate uses the SHA-256 hash function. SHA-256 was
# introduced in RHEL5.2. Therefore, clients older than RHEL5.2 will not
# be able to interoperate with IPA server 3.x.
# Please note that this script assumes /etc/openldap/cacerts as the
# default CA certificate location. If this value is different on your
# system the script needs to be modified accordingly.
# Download the CA certificate of the IPA server
mkdir -p -m 755 /etc/openldap/cacerts
wget http://idm.example.com/ipa/config/ca.crt -O /etc/openldap/cacerts/ca.crt
(...)
```

You can configure a Linux client using the **ipa-advise** utility by running the displayed instructions as a shell script or by executing the instructions manually.

To run the instructions as a shell script:

1. Create the script file.

   [root@server ~]# ipa-advise *config-redhat-nss-ldap* > *setup_script.sh*

2. Add execute permissions to the file using the **chmod** utility.

   [root@server ~]# chmod +x *setup_script.sh*

3. Copy the script to the client using the **scp** utility.

   [root@server ~]# scp *setup_script.sh* root@*client*

4. Run the script on the client.

   [root@client ~]# ./*setup_script.sh*

> **IMPORTANT**
>
> Always read and review the script file carefully before you run it on the client.

To configure the client manually, follow and execute the instructions displayed by **ipa-advise** from the command line.

5.8. TROUBLESHOOTING CROSS-FOREST TRUSTS

This section provides information about possible problems in an cross-forest trust environment and ways to solve them.

5.8.1. Troubleshooting the ipa-extdom Plug-in

IdM clients in an IdM domain with a trust to Active Directory (AD) cannot receive information about users and groups from AD directly. Additionally, IdM does not store information about AD users in Directory Server running on IdM masters. Instead, IdM servers use the **ipa-extdom** to receive information about AD users and groups and forwards them to the requesting client.

Setting the Config Timeout of the ipa-extdom Plug-in

The **ipa-extdom** plug-in sends a request to SSSD for the data about AD users. However, not all requested data might be already in the cache of SSSD. In this case, SSSD requests the data from the AD domain controller (DC). This can be time-consuming for certain operations. The config timeout value defines the time in milliseconds of how long the **ipa-extdom** plug-in waits for a reply of SSSD before the plug-in cancels the connection and returns a timeout error to the caller.

By default, the config timeout is **10000** milliseconds (10 seconds).

- If you set a too small value, such as **500** milliseconds, SSSD might not have enough time to reply and requests will always return a timeout.

- If the value is too large, such as **30000** milliseconds (30 seconds), a single request might block the connection to SSSD for this amount of time. Since only one thread can connect to SSSD at a time, all other requests from the plug-in have to wait.

- If there are many requests sent by IdM clients, they can block all available workers configured for Directory Server and, as a consequence, the server might not be able to reply to any kind of request for some time.

Change the config timeout in the following situations:

- If IdM clients frequently receive timeout errors before their own search timeout is reached when requesting information about AD users and groups, the config timeout value is too small.

- If the Directory Server on the IdM server is often locked and the **pstack** utility reports that many or all worker threads are handling **ipa-extdom** requests at this time, the value is too large.

For example, to set the config value to **20000** milliseconds (20 seconds), enter:

```
# ldapmodify -D "cn=directory manager" -W
dn: cn=ipa_extdom_extop,cn=plugins,cn=config

changetype: modify
replace: ipaExtdomMaxNssTimeout
ipaExtdomMaxNssTimeout: 20000
```

Setting the Maximum Size of the ipa-extdom Plug-in Buffer Used for NSS Calls

The **ipa-extdom** plug-in uses calls which use the same API as typical name service switch (NSS) calls to request data from SSSD. Those calls use a buffer where SSSD can store the requested data. If the buffer is too small, SSSD returns an **ERANGE** error and the plug-in retries the request with a larger buffer. The *ipaExtdomMaxNssBufSize* attribute in the **cn=ipa_extdom_extop,cn=plugins,cn=config** entry of Directory Server on the IdM master defines the maximum size of the buffer in bytes.

By default, the buffer is **134217728** bytes (128 MB). Only increase the value if, for example, a group has so many members that all names do not fit into the buffer and the IPA client cannot resolve the group.

For example, to set the buffer to **268435456** bytes (256 MB), enter:

```
# ldapmodify -D "cn=directory manager" -W

dn: cn=ipa_extdom_extop,cn=plugins,cn=config
changetype: modify
replace: ipaExtdomMaxNssBufSize
ipaExtdomMaxNssBufSize: 268435456
```

PART III. INTEGRATING A LINUX DOMAIN WITH AN ACTIVE DIRECTORY DOMAIN: SYNCHRONIZATION

This part provides instruction on how to synchronize **Active Directory** and **Identity Management** users, how to migrate existing environments from synchronization to trust, and how to use **ID Views** in **Active Directory** environments.

CHAPTER 6. SYNCHRONIZING ACTIVE DIRECTORY AND IDENTITY MANAGEMENT USERS

This chapter describes synchronization between Active Directory and Red Hat Enterprise Linux Identity Management. Synchronization is one of the two methods for indirect integration of the two environments. For details on the cross-forest trust, which is the other, recommended method, see Chapter 5, *Creating Cross-forest Trusts with Active Directory and Identity Management* . If you are unsure which method to choose for your environment, read Section 1.3, "Indirect Integration".

Identity Management uses *synchronization* to combine the user data stored in an Active Directory domain and the user data stored in the IdM domain. Critical user attributes, including passwords, are copied and synchronized between the services.

Entry synchronization is performed through a process similar to replication, which uses hooks to connect to and retrieve directory data from the Windows server.

Password synchronization is performed through a Windows service which is installed on the Windows server and then communicates to the Identity Management server.

6.1. SUPPORTED WINDOWS PLATFORMS

Synchronization is supported with Active Directory forests that use the following forest and domain functional levels:

- Forest functional level range: Windows Server 2008 – Windows Server 2012 R2

- Domain functional level range: Windows Server 2008 – Windows Server 2012 R2

The following operating systems are explicitly supported and tested for synchronization using the mentioned functional levels:

- Windows Server 2012 R2

- Windows Server 2016

PassSync 1.1.5 or later is compatible with all supported Windows Server versions.

6.2. ABOUT ACTIVE DIRECTORY AND IDENTITY MANAGEMENT

Within the IdM domain, information is shared among servers and replicas by copying that information, reliably and predictably, between data masters (servers and replicas). This process is *replication*.

A similar process can be used to share data between the IdM domain and a Microsoft Active Directory domain. This is *synchronization*.

Synchronization is the process of copying user data back and forth between Active Directory and Identity Management. When users are synchronized between Active Directory and Identity Management, the directory synchronization (DirSync) LDAP server extension control is used to search a directory for objects that have changed.

AD-IDM SYNC PROCESS
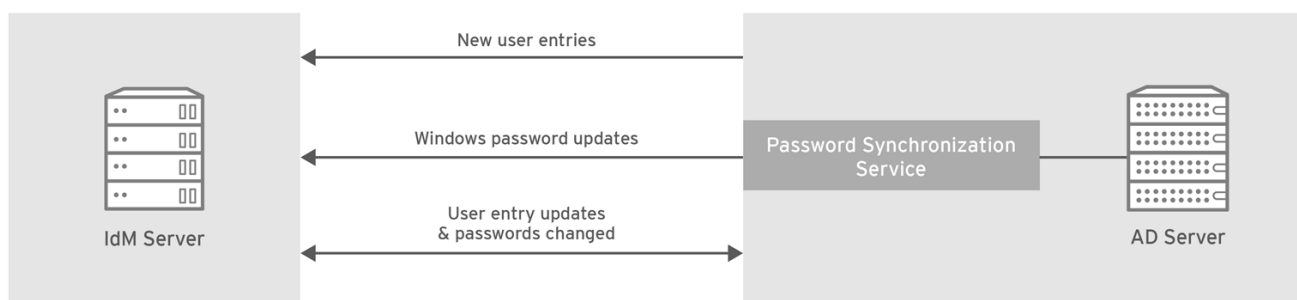
RHEL_404973_0516

Figure 6.1. Active Directory and IdM Synchronization

Synchronization is defined in an *agreement* between an IdM server and an Active Directory domain controller. The agreement defines all of the information required to identify user entries that can be synchronized, such as the subtree to synchronize, as well as defining how account attributes are handled. The synchronization agreements are created with default values which can be tweaked to meet the needs of a specific domain. When two servers are involved in synchronization, they are called *peers*.

Table 6.1. Information in a Synchronization Agreement

Windows Information	IdM Information
- User subtree (**cn=Users,$SUFFIX**) - Connection information - Active Directory administrator user name and password - Password Synchronization Service password - CA certificate	User subtree (**ou=People,$SUFFIX**)

Synchronization is most commonly *bidirectional*. Information is sent back and forth between the IdM and the Windows domain in a process that is very similar to how IdM servers and replicas share information among themselves. An exception are new user entries, which are only added from the Windows domain to the IdM domain. It is possible to configure synchronization to only synchronize one way. That is *unidirectional* synchronization.

To prevent the risk of data conflicts, only one directory should originate or remove user entries. This is typically the Windows directory, which is the primary identity store in the IT environment, and then new accounts or account deletions are synchronized to the Identity Management peer. Either directory can modify entries.

Synchronization, then, is configured between one Identity Management server and one Active Directory domain controller. The Identity Management server propagates throughout to the IdM domain, while the domain controller propagates changes throughout the Windows domain.
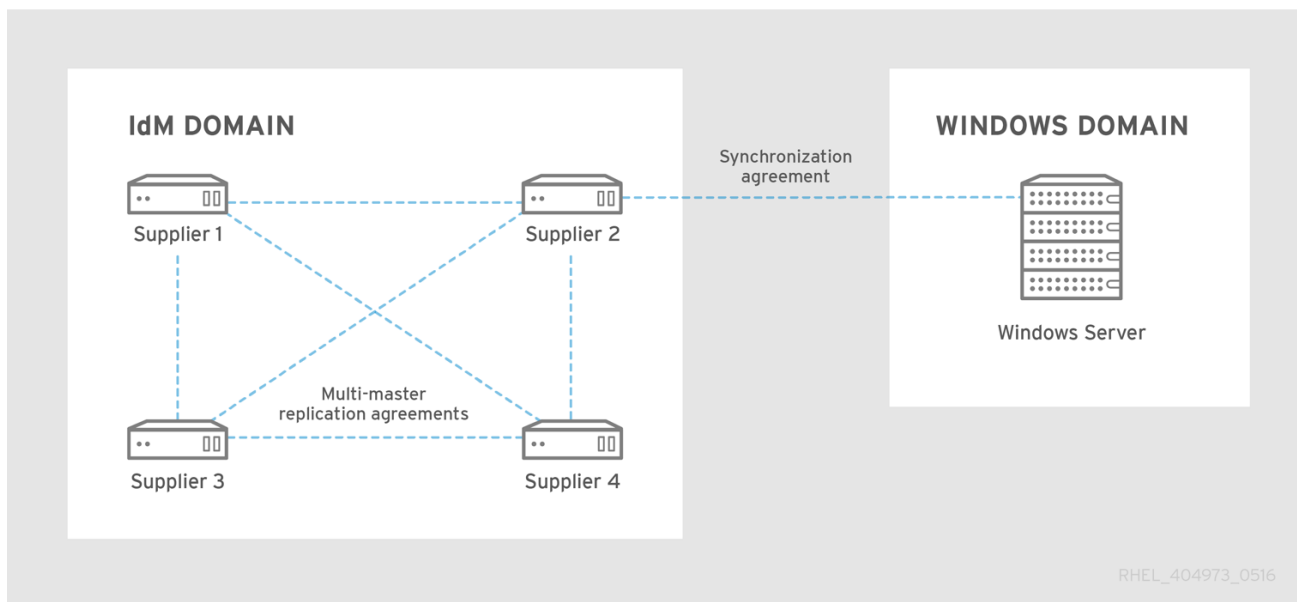
Figure 6.2. Synchronization Topology

There are some key features to IdM synchronization:

- A synchronization operation runs every five minutes. To modify the frequency, set the **winSyncInterval** attribute in the Active Directory peers DN:

  > cn=meTo*winserver.ad.example.com*,cn=replica,cn=*dc\3Didm*,*dc\3Dexample*,*dc\3Dcom*,cn=
  > mapping tree,cn=config

- Synchronization can only be configured with one Active Directory domain.

- Synchronization can only be configured with *one* Active Directory domain controller.

- Only user information is synchronized; group information is not.

- Both user attributes and passwords can be synchronized.

- While modifications are bidirectional (going both from Active Directory to IdM and from IdM to Active Directory), creating accounts is only unidirectional, from Active Directory to Identity Management. New accounts created in Active Directory are synchronized over to IdM automatically. However, user accounts created in IdM must also be created in Active Directory before they will be synchronized. In this situation, the synchronization process tries to find a matching account with the same value for the **uid** attribute in IdM than for the **sAMAccountName** attribute in Active Directory. If a match is found, the IdM **ntUserDomainId** attribute is set to the Active Directory **objectGUID** value. These attributes are globally unique and immutable, and entries stay synchronized, even if they are moved or renamed.

- Account lock information is synchronized by default, so a user account which is disabled in one domain is disabled in the other.

- Password synchronization changes take effect immediately. If a user password is added or changed on one peer, that change is immediately propagated to the other peer server.

  The Password Synchronization client synchronizes new passwords or password updates.

  Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing

passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

- While there can only be one agreement, the PassSync service must be installed on every Active Directory server.

When Active Directory users are synchronized over to IdM, certain attributes (including Kerberos and POSIX attributes) will have IPA attributes automatically added to the user entries. These attributes are used by IdM within its domain. They are not synchronized back over the corresponding Active Directory user entry.

Some of the data in synchronization can be modified as part of the synchronization process. For examples, certain attributes can be automatically added to Active Directory user accounts when they are synced over to the IdM domain. These attribute changes are defined as part of the synchronization agreement and are described in Section 6.5.2, "Changing the Behavior for Synchronizing User Account Attributes".

6.3. ABOUT SYNCHRONIZED ATTRIBUTES

Identity Management synchronizes a subset of user attributes between IdM and Active Directory user entries. Any other attributes present in the entry, either in Identity Management or in Active Directory, are ignored by synchronization.
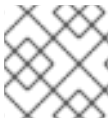
> **NOTE**
>
> Most POSIX attributes are not synchronized.

Although there are significant schema differences between the Active Directory LDAP schema and the 389 Directory Server LDAP schema used by Identity Management, there are many attributes that are the same. These attributes are simply synchronized between the Active Directory and IdM user entries, with no changes to the attribute name or value format.

User Schema That Are the Same in Identity Management and Windows Servers

- cn[2]

- physicalDeliveryOfficeName

- description

- postOfficeBox

- destinationIndicator

- postalAddress

- facsimileTelephoneNumber

- postalCode

- givenname

- registeredAddress

- homePhone

- sn

- homePostalAddress

- st

- initials

- street

- l

- telephoneNumber

- mail

- teletexTerminalIdentifier

- mobile

- telexNumber

- o

- title

- ou

- userCertificate

- pager

- x121Address

Some attributes have different names but still have direct parity between IdM (which uses 389 Directory Server) and Active Directory. These attributes are *mapped* by the synchronization process.

Table 6.2. User Schema Mapped between Identity Management and Active Directory

Identity Management	Active Directory
cn[a]	name
nsAccountLock	userAccountControl
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon

Identity Management	Active Directory
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

[a] The *cn* is mapped directly (*cn* to *cn*) when synchronizing from Identity Management to Active Directory. When synchronizing from Active Directory *cn* is mapped from the *name* attribute in Active Directory to the *cn* attribute in Identity Management.

6.3.1. User Schema Differences between Identity Management and Active Directory

Even though attributes may be successfully synchronized between Active Directory and IdM, there may still be differences in how Active Directory and Identity Management define the underlying X.500 object classes. This could lead to differences in how the data are handled in the different LDAP services.

This section describes the differences in how Active Directory and Identity Management handle some of the attributes which can be synchronized between the two domains.

6.3.1.1. Values for cn Attributes

In 389 Directory Server, the *cn* attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the Identity Management *cn* attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that, potentially, if a *cn* value is added to an Active Directory entry and that value is not one of the values for *cn* in Identity Management, then all of the Identity Management *cn* values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the *cn* attribute as its naming attribute, where Identity Management uses *uid*. This means that there is the potential to rename the entry entirely (and accidentally) if the *cn* attribute is edited in the Identity Management.

6.3.1.2. Values for street and streetAddress

Active Directory uses the attribute *streetAddress* for a user's postal address; this is the way that 389 Directory Server uses the *street* attribute. There are two important differences in the way that Active Directory and Identity Management use the *streetAddress* and *street* attributes, respectively:

- In 389 Directory Server, *streetAddress* is an alias for *street*. Active Directory also has the *street* attribute, but it is a separate attribute that can hold an independent value, not an alias for *streetAddress*.

- Active Directory defines both *streetAddress* and *street* as single-valued attributes, while 389 Directory Server defines *street* as a multi-valued attribute, as specified in RFC 4519.

Because of the different ways that 389 Directory Server and Active Directory handle *streetAddress* and *street* attributes, there are two rules to follow when setting address attributes in Active Directory and Identity Management:

- The synchronization process maps *streetAddress* in the Active Directory entry to *street* in Identity Management. To avoid conflicts, the *street* attribute should not be used in Active Directory.

- Only one Identity Management *street* attribute value is synchronized to Active Directory. If the *streetAddress* attribute is changed in Active Directory and the new value does not already exist in Identity Management, then all *street* attribute values in Identity Management are replaced with the new, single Active Directory value.

6.3.1.3. Constraints on the initials Attribute

For the *initials* attribute, Active Directory imposes a maximum length constraint of six characters, but 389 Directory Server does not have a length limit. If an *initials* attribute longer than six characters is added to Identity Management, the value is trimmed when it is synchronized with the Active Directory entry.

6.3.1.4. Requiring the surname (sn) Attribute

Active Directory allows **person** entries to be created without a surname attribute. However, RFC 4519 defines the **person** object class as requiring a surname attribute, and this is the definition used in Directory Server.

If an Active Directory **person** entry is created without a surname attribute, that entry will not be synchronized over to IdM since it fails with an object class violation.

6.3.2. Active Directory Entries and POSIX Attributes

When a Windows user account contains values for the *uidNumber* and *gidNumber* attributes, WinSync does not synchronize these values over to Identity Management. Instead, it creates new UID and GID values in Identity Management.

As a result, the values for *uidNumber* and *gidNumber* are different in Active Directory and in Identity Management.

6.4. SETTING UP ACTIVE DIRECTORY FOR SYNCHRONIZATION

Synchronizing user accounts is enabled within IdM. It is only necessary to set up a synchronization agreement (Section 6.5.1, "Creating Synchronization Agreements"). However, the Active Directory does need to be configured in a way that allows the Identity Management server to connect to it.

6.4.1. Creating an Active Directory User for Synchronization

On the Windows server, it is necessary to create the user that the IdM server will use to connect to the Active Directory domain.

The process for creating a user in Active Directory is covered in the Windows server documentation at http://technet.microsoft.com/en-us/library/cc732336.aspx. The new user account must have the proper permissions:

- Grant the synchronization user account **Replicating directory changes** rights to the synchronized Active Directory subtree. Replicator rights are required for the synchronization user to perform synchronization operations.

  Replicator rights are described in http://support.microsoft.com/kb/303972.

- Add the synchronization user as a member of the **Account Operators** and **Enterprise Read-only Domain Controllers** groups. It is not necessary for the user to belong to the **Domain Admins** group.

6.4.2. Setting up an Active Directory Certificate Authority

The Identity Management server connects to the Active Directory server using a secure connection. This requires that the Active Directory server have an available CA certificate or CA certificate chain available, which can be imported into the Identity Management security databases, so that the Windows server is a trusted peer.

While this could technically be done with an external (to Active Directory) CA, most deployments should use the Certificate Services available with Active Directory.

The procedure for setting up and configuring certificate services on Active Directory is covered in the Microsoft documentation at http://technet.microsoft.com/en-us/library/cc772393(v=WS.10).aspx.

6.5. MANAGING SYNCHRONIZATION AGREEMENTS

6.5.1. Creating Synchronization Agreements

Synchronization agreements are created on the IdM server using the **ipa-replica-manage connect** command because it creates a *connection* to the Active Directory domain. To establish an encrypted connection to Active Directory, IdM must to trust the Windows CA certificate.

1. Copy the root certificate authority (CA) certificate to the IdM server:

    a. If your Active Directory CA certificate is self-signed:

      i. Export the Active Directory CA certificate on the Windows server.

          A. Press the **Super key**+**R** combination to open the **Run** dialog.

          B. Enter **certsrv.msc** and click **OK**.

          C. Right-click on the name of the local Certificate Authority and choose **Properties**.

          D. On the **General** tab, select the certificate to export in the **CA certificates** field and click **View Certificate**.

          E. On the **Details** tab, click **Copy to File** to start the **Certificate Export Wizard**.

          F. Click **Next**, and then select **Base-64 encoded X.509 (.CER)**.
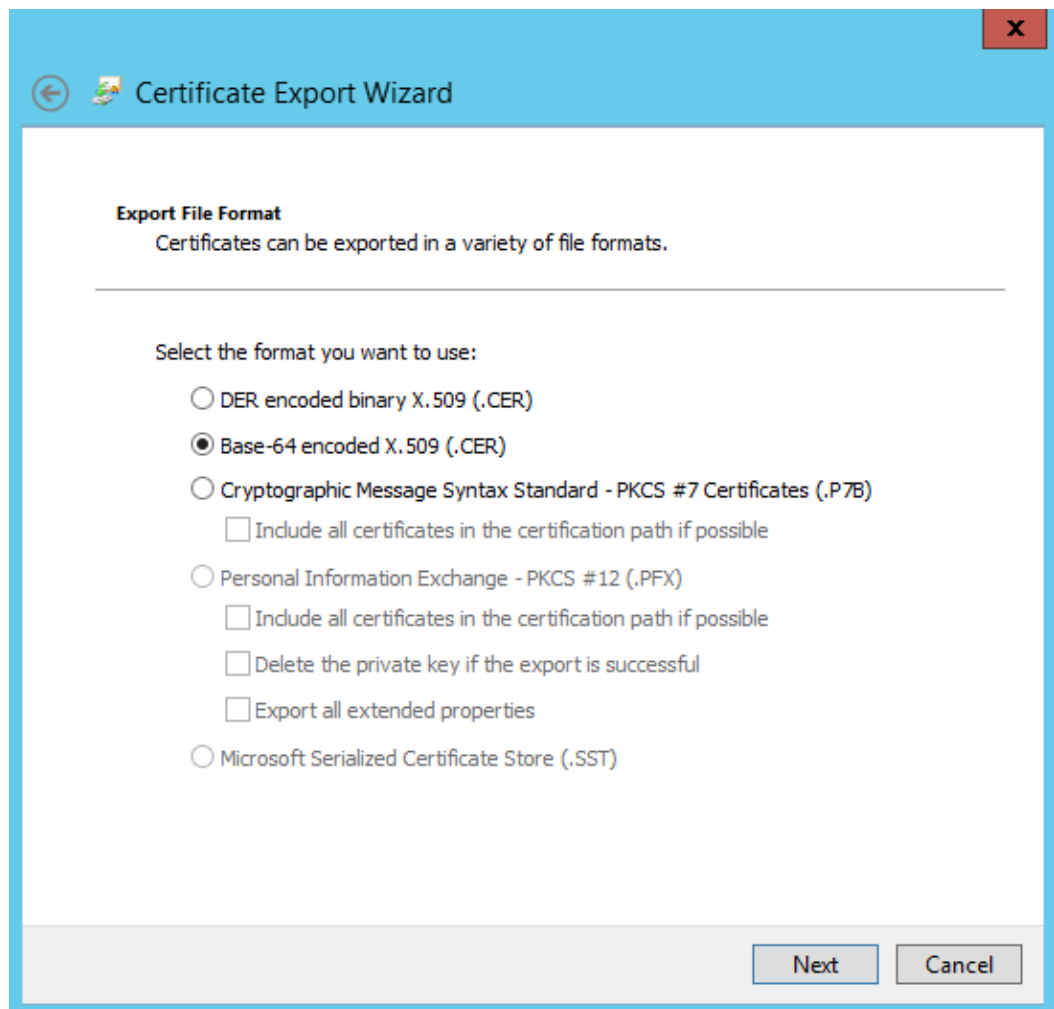
G. Specify a suitable directory and file name for the exported file. Click **Next** to export the certificate, and then click **Finish**.

H. Copy the exported certificate to the IdM server machine.

b. If your Active Directory CA certificate is signed by an external CA:

   i. To find out what certificate is the CA root certificate, display the certificate chain:

```
# openssl s_client -connect adserver.example.com:636
CONNECTED(00000003)
depth=1 C = US, O = Demo Company, OU = IT, CN = Demo CA-28
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/O=Demo Company/OU=IT/CN=adserver.example.com
   i:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
 1 s:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
   i:/C=US/O=Demo Company/OU=IT/CN=Demo Root CA 2
```

   The previous example shows that the Active Directory server's CA certificate is signed by **CN=Demo CA-1**, which is signed by **CN=Demo Root CA 2**. This means that **CN=Demo Root CA 2** is the root CA.

   ii. Copy the CA certificate to the IdM server.

2. Remove any existing Kerberos credentials on the IdM server.

   ```
   $ kdestroy
   ```

3. Use the **ipa-replica-manage** command to create a Windows synchronization agreement. This requires the **--winsync** option. If passwords will be synchronized as well as user accounts, then also use the **--passsync** option and set a password to use for Password Synchronization.

   The **--binddn** and **--bindpw** options give the user name and password of the system account on the Active Directory server that IdM will use to connect to the Active Directory server.

   ```
   $ ipa-replica-manage connect --winsync \
     --binddn cn=administrator,cn=users,dc=example,dc=com \
     --bindpw Windows-secret \
     --passsync secretpwd \
     --cacert /etc/openldap/cacerts/windows.cer \
     adserver.example.com -v
   ```

   - **--winsync**: Identifies this as a Windows synchronization agreement.

   - **--binddn**: IdM uses this DN of an Active Directory account to bind to the remote directory and synchronize attributes.

   - **--bindpw**: Password for the synchronization account.

   - **--cacert**: Full path and file name to the:

     - Active Directory CA certificate, if the CA was self–signed.

     - external CA certificate, if the Active Directory CA was signed by an external CA.

   - **--win-subtree**: DN of the Windows directory subtree containing the users to synchronize. The default value is **cn=Users,$SUFFIX**.

   - **AD_server_name**: Fully qualified domain name (FQDN) of the Active Directory domain controller.

4. When prompted, enter the Directory Manager password.

5. *Optional.* Configure Password Synchronization, as in Section 6.6.2, "Setting up Password Synchronization". Without the Password Synchronization client, user attributes are synchronized between the peer servers, but passwords are not.

   > **NOTE**
   >
   > The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.
   >
   > Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

6.5.2. Changing the Behavior for Synchronizing User Account Attributes

When the synchronization agreement is created, it has certain default behaviors defined for how the synchronization process handles the user account attributes during synchronization. The types of behaviors are things like how to handle lockout attributes or how to handle different DN formats. This behavior can be changed by editing the synchronization agreement.

The synchronization agreement exists as a special plug-in entry in the LDAP server and each attribute behavior is set through an LDAP attribute. To change the synchronization behavior, use the **ldapmodify** command to modify the LDAP server entry directly.

For example, account lockout attributes are synchronized between IdM and Active Directory by default, but this can be disabled by editing the *ipaWinSyncAcctDisable* attribute. (Changing this means that if an account is disabled in Active Directory, it is still active in IdM and vice versa.)

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password

dn: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
replace: ipaWinSyncAcctDisable
ipaWinSyncAcctDisable: none

modifying entry "cn=ipa-winsync,cn=plugins,cn=config"
```

The following is an overview of synchronization settings attributes:

General User Account Parameters

- **ipaWinSyncNewEntryFilter**: Sets the search filter to use to find the entry which contains the list of object classes to add to new user entries.

  Default value: **(cn=ipaConfig)**

- **ipaWinSyncNewUserOCAttr**: Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.

  Default value: **ipauserobjectclasses**

- **ipaWinSyncHomeDirAttr**: Identifies which attribute in the entry contains the default location of the POSIX home directory.

  Default value: **ipaHomesRootDir**

- **ipaWinSyncUserAttr**: Sets an additional attribute with a specific value to add to Active Directory users when they are synchronized over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and the synchronization process adds all of the values to the entry.

  Example: **ipaWinSyncUserAttr: attributeName attributeValue**

  > **NOTE**
  >
  > This only sets the attribute value if the entry does not already have that attribute present. If the attribute is present, then the entry's value is used when the Active Directory entry is synchronized over.

- **ipaWinSyncForceSync**: Sets whether existing IdM users that match existing AD users should be forced to be synchronized. When set to **true**, such IdM users are automatically edited so that they are synchronized.

  Possible values: **true** | **false**

  If an IdM user account has a **uid** parameter which is identical to the **sAMAccountName** in an existing Active Directory user, then that account is *not* synchronized by default. This attribute tells the synchronization service to add the **ntUser** and **ntUserDomainId** to the IdM user entries automatically, which allows them to be synchronized.

User Account Lock Parameters

- **ipaWinSyncAcctDisable**: Sets which way to synchronize account lockout attributes. It is possible to control which account lockout settings are in effect. For example, **to_ad** means that when account lockout attribute is set in IdM, its value is synchronized over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synchronized from both domains.

  Possible values: **both** (default), **to_ad**, **to_ds**, **none**

- **ipaWinSyncInactivatedFilter**: Sets the search filter to use to find the DN of the group used to hold inactivated (disabled) users. This does not need to be changed in most deployments.

  Default value: **(&(cn=inactivated)(objectclass=groupOfNames))**

Group Parameters

- **ipaWinSyncDefaultGroupAttr**: Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the **gidNumber** for the user account.

  Default value: **ipaDefaultPrimaryGroup**

- **ipaWinSyncDefaultGroupFilter**: Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the **gidNumber** for the user account.

  Default value: **ipaDefaultPrimaryGroup**

Realm Parameters

- **ipaWinSyncRealmAttr**: Sets the attribute which contains the realm name in the realm entry.

  Default value: **cn**

- **ipaWinSyncRealmFilter**: Sets the search filter to use to find the entry which contains the IdM realm name.

  Default value: **(objectclass=krbRealmContainer)**

6.5.3. Changing the Synchronized Windows Subtree

Creating a synchronization agreement automatically sets the two subtrees to use as the synchronized user database. In IdM, the default is **cn=users,cn=accounts,$SUFFIX**, and for Active Directory, the default is **CN=Users,$SUFFIX**.

The value for the Active Directory subtree can be set to a non-default value when the synchronization agreement is created by using the **--win-subtree** option. After the agreement is created, the Active Directory subtree can be changed by using the **ldapmodify** command to edit the **nsds7WindowsReplicaSubtree** value in the synchronization agreement entry.

1. Get the name of the synchronization agreement, using **ldapsearch**. This search returns only the values for the **dn** and **nsds7WindowsReplicaSubtree** attributes instead of the entire entry.

   ```
   [jsmith@ipaserver ~]$ ldapsearch -xLLL -D "cn=directory manager" -w password -p 389 -h
   ipaserver.example.com -b cn=config objectclass=nsdswindowsreplicationagreement dn
   nsds7WindowsReplicaSubtree

   dn:
   cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
   tree,cn=config
   nsds7WindowsReplicaSubtree: cn=users,dc=example,dc=com

   ... 8< ...
   ```

2. Modify the synchronization agreement

   ```
   [jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -W -p 389 -h
   ipaserver.example.com <<EOF
    dn:
   cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
   tree,cn=config
    changetype: modify
    replace: nsds7WindowsReplicaSubtree
    nsds7WindowsReplicaSubtree: cn=alternateusers,dc=example,dc=com
    EOF

    modifying entry
   "cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
   tree,cn=config"
   ```

The new subtree setting takes effect immediately. If a synchronization operation is currently running, then it takes effect as soon as the current operation completes.

6.5.4. Configuring Uni-directional Synchronization

By default, all modifications and deletions are bidirectional. A change in Active Directory is synchronized over to Identity Management, and a change to an entry in Identity Management is synchronized over to Active Directory. This is essentially an equitable, multi-master relationship, where both Active Directory and Identity Management are equal peers in synchronization and are both data masters.

However, there can be some data structure or IT designs where only one domain should be a data master and the other domain should accept updates. This changes the synchronization relationship from a multi-master relationship (where the peer servers are equal) to a master consumer relationship.

This is done by setting the **oneWaySync** parameter on the synchronization agreement. The possible values are **fromWindows** (for Active Directory to Identity Management synchronization) and **toWindows** (for Identity Management to Active Directory synchronization).

For example, to synchronize changes from Active Directory to Identity Management:

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password -p 389 -h
ipaserver.example.com

dn: cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
tree,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```
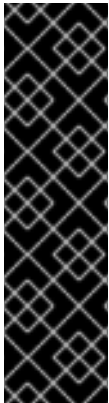
> **IMPORTANT**
>
> Enabling unidirectional synchronization does *not* automatically prevent changes on the
> unsynchronized server, and this can lead to inconsistencies between the synchronization
> peers between synchronization updates. For example, unidirectional synchronization is
> configured to go from Active Directory to Identity Management, so Active Directory is (in
> essence) the data master. If an entry is modified or even deleted on the
> Identity Management, then the Identity Management information is different then the
> information and those changes are never carried over to Active Directory. During the next
> synchronization update, the edits are overwritten on the Directory Server and the deleted
> entry is re-added.

6.5.5. Deleting Synchronization Agreements

Synchronization can be stopped by deleting the synchronization agreement which *disconnects* the IdM
and Active Directory servers. In the inverse of creating a synchronization agreement, deleting a
synchronization agreement uses the **ipa-replica-manage disconnect** command and then the host
name of the Active Directory server.

1. Delete the synchronization agreement.

   ```
   # ipa-replica-manage disconnect adserver.ad.example.com
   ```

2. List the certificates in the IdM directory certificate database:

   ```
   # certutil -L -d /etc/dirsrv/slapd-IDM-EXAMPLE-COM/
   Certificate Nickname            Trust Attributes
                                   SSL,S/MIME,JAR/XPI

   IDM.EXAMPLE.COM IPA CA              CT,C,C
   CN=adserver,DC=ad,DC=example,DC=com    C,,
   Server-Cert                     u,u,u
   ```

3. Remove the Active Directory CA certificate from the IdM server database:

   ```
   # certutil -D -d /etc/dirsrv/slapd-IDM-EXAMPLE-COM/ -n
   "CN=adserver,DC=ad,DC=example,DC=com"
   ```

6.5.6. Winsync Agreement Failures

**Creating the synchronization agreement fails because it cannot connect to the Active Directory
server.**

One of the most common synchronization agreement failures is that the IdM server cannot connect to the Active Directory server:

> "Update failed! Status: [81 - LDAP error: Can't contact LDAP server]

This can occur if the wrong Active Directory CA certificate was specified when the agreement was created. This creates duplicate certificates in the IdM LDAP database (in the **/etc/dirsrv/slapd-DOMAIN/** directory) with the name *Imported CA*. This can be checked using **certutil**:

```
$ certutil -L -d /etc/dirsrv/slapd-DOMAIN/

Certificate Nickname                    Trust Attributes
SSL,S/MIME,JAR/XPI

CA certificate                          CTu,u,Cu
Imported CA                              CT,,C
Server-Cert                             u,u,u
Imported CA                              CT,,C
```

To resolve this issue, remove the CA certificate from the certificate database:

> # certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"

There are errors saying passwords are not being synchronized because it says the entry exists

For some entries in the user database, there may be an informational error message that the password is not being reset because the entry already exists:

> "Windows PassSync entry exists, not resetting password"

This is not an error. This message occurs when an exempt user, the Password Synchronization user, is not being changed. The Password Synchronization user is the operational user which is used by the service to change the passwords in IdM.

6.6. MANAGING PASSWORD SYNCHRONIZATION

Synchronizing user entries is configured with the synchronization agreement. However, passwords in both Active Directory and Identity Management are not part of the normal user synchronization process. A separate client must be installed on the Active Directory servers to capture passwords as user accounts are created or passwords are changed, and then to forward that password information with the synchronized updates.

NOTE

The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.

Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

6.6.1. Setting up the Windows Server for Password Synchronization

Synchronizing passwords requires these things:

- Active Directory must be running in SSL.
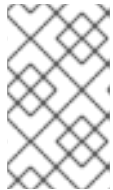
> **NOTE**
>
> Install the Microsoft Certificate System in Enterprise Root Mode.
> Active Directory will then automatically enroll to retrieve its SSL server
> certificate.

- The Password Synchronization Service must be installed on *each* Active Directory domain controller. To synchronize a password from Windows, the PassSync service requires access to the unencrypted password to synchronize it over a secure connection to IdM. Because users can change their passwords on every domain controller, the installation of the PassSync service on each domain controller is necessary.

- The password policy must be set similar on IdM and Active Directory side. When the synchronization destination receives an updated password, it was only validated to match the policy on the source. It is not re-validated on the synchronization destination.

To verify that the Active Directory password complexity policy is enabled, run on an Active Directory domain controller:

```
> dsquery * -scope base -attr pwdProperties
  pwdProperties
  1
```

If the value of the attribute **pwdProperties** is set to **1**, the password complexity policy is enabled for the domain.

> **NOTE**
>
> If you are unsure if group policies define deviating password settings for Organizational
> Units (ou), ask your group policy administrator.

To enable the Active Directory password complexity setting for the whole domain:

1. Run **gpmc.msc** from the command line.

2. Select **Group Policy Management**.

3. Open **Forest:** *ad.example.com* → **Domains** → *ad.example.com*.

4. Right-click the **Default Domain Policy** entry and select **Edit**.
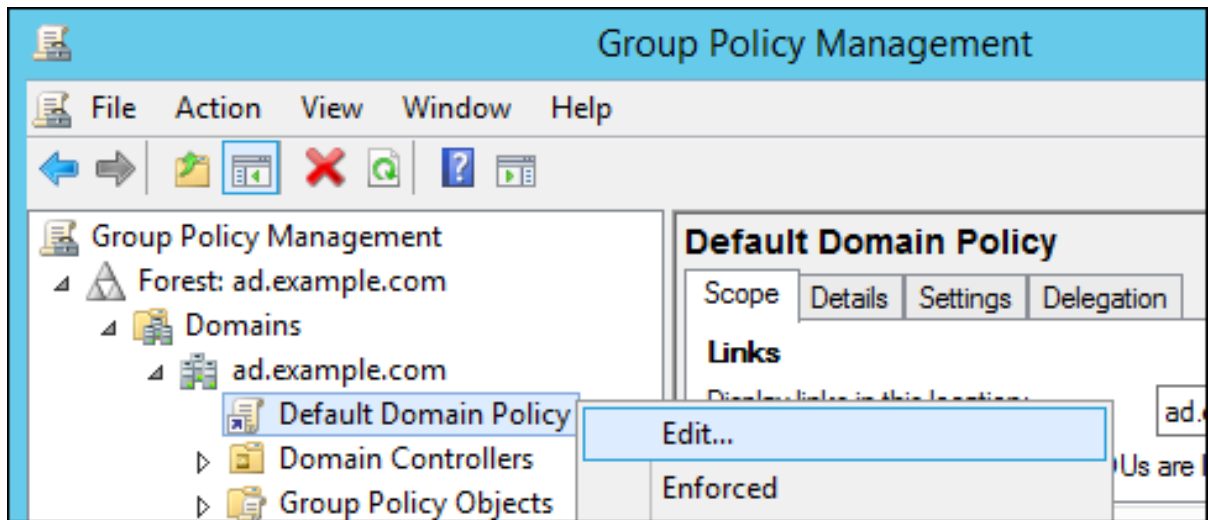
5. The **Group Policy Management Editor** opens automatically.

6. Open **Computer Configuration → Policies → Windows Settings → Security Settings → Account Policies → Password Policy**.

7. Enable the **Password must meet complexity requirements** option and save.
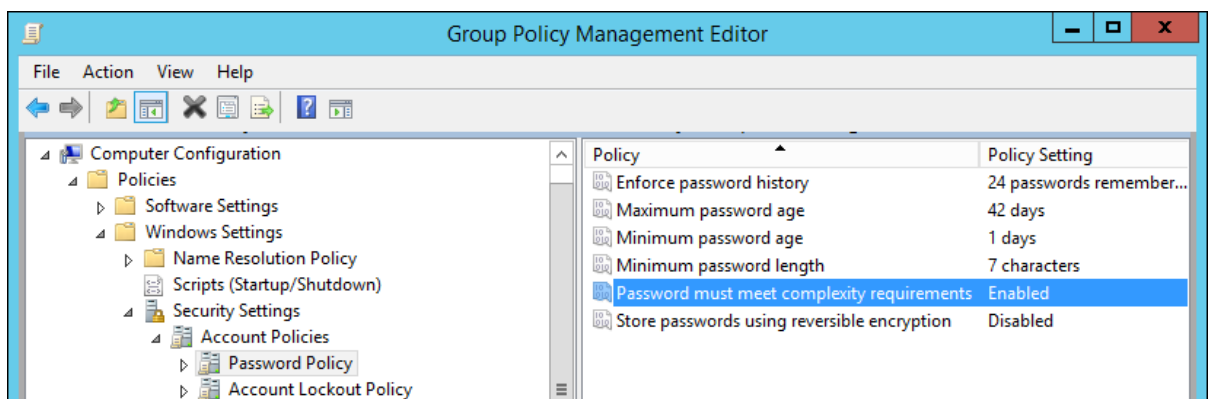
6.6.2. Setting up Password Synchronization

Install the Password Synchronization Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the **RedHat-PassSync-*.msi** file to the Active Directory domain controller:

   1. Log in to the Customer Portal.

   2. Click **Downloads** at the top of the page.

   3. Select **Red Hat Enterprise Linux** from the product list.

   4. Select the most recent version of Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7 and architecture.

   5. Download **WinSync Installer** for the architecture of the Active Directory domain controller by clicking the **Download Now** button.

2. Double-click the **MSI** file to install it.

3. The **Password Synchronrization Setup** window appears. Hit **Next** to begin installing.

4. Fill in the information to establish the connection to the IdM server.

    o   The IdM server connection information, including the host name and secure port number.

    o   The user name of the system user which Active Directory uses to connect to the IdM machine. This account is configured automatically when synchronization is configured on the IdM server. The default account is **uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com**.

    o   The password set in the **--passsync** option when the synchronization agreement was created.

    o   The search base for the people subtree on the IdM server. The Active Directory server connects to the IdM server similar to an **ldapsearch** or replication operation, so it has to know where in the IdM subtree to look for user accounts. The user subtree is **cn=users,cn=accounts,dc=example,dc=com**.

    o   The certificate token is not used at this time, so that field should be left blank.

Hit **Next**, then **Finish** to install Password Synchronization.

5. Import the IdM server's CA certificate into the PassSync certificate store.

    1. Download the IdM server's CA certificate from **http://ipa.example.com/ipa/config/ca.crt**.

    2. Copy the IdM CA certificate to the Active Directory server.

    3. Install the IdM CA certificate in the Password Synchronization database. For example:

```
cd "C:\Program Files\Red Hat Directory Password Synchronization"

certutil.exe -d . -A -n "IPASERVER.EXAMPLE.COM IPA CA" -t CT,, -a -i ipaca.crt
```

6. Reboot the Windows machine to start Password Synchronization.

> **NOTE**
>
> The Windows machine must be rebooted. Without the rebooting, **PasswordHook.dll** is not enabled, and password synchronization will not function.

7. If passwords for existing accounts should be synchronized, reset the user passwords.

> **NOTE**
>
> The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.
>
> Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

The first attempt to synchronize passwords, which happened when the Password Synchronization application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory synchronization peers. The tools to create the certificate and key databases is installed with the **.msi**.

The password synchronization client cannot synchronize passwords for members of the IdM **admin** group. This is an intended behavior to prevent, for example, password synchronization agents or low level user administrators to change passwords of top level administrators.

> **NOTE**
>
> Passwords are only validated on the synchronization source to match the password policies. To verify and enable the Active Directory password complexity policy, see Section 6.6.1, "Setting up the Windows Server for Password Synchronization" .

---

[2] The **cn** is treated differently than other synchronized attributes. It is mapped directly (**cn** to **cn**) when synchronizing from Identity Management to Active Directory. When synchronizing from Active Directory to Identity Management, however, **cn** is mapped from the **name** attribute on Windows to the **cn** attribute in Identity Management.

CHAPTER 7. MIGRATING EXISTING ENVIRONMENTS FROM SYNCHRONIZATION TO TRUST

Synchronization and *trust* are two possible approaches to indirect integration. Synchronization is generally discouraged, and Red Hat recommends to use the approach based on Active Directory (AD) trust instead. See Section 1.3, "Indirect Integration" for details.

This chapter describes how to migrate an existing synchronization-based setup to AD trust. The following migrating options are available in IdM:

- Section 7.1, "Migrate from Synchronization to Trust Automatically Using **ipa-winsync-migrate**"

- Section 7.2, "Migrate from Synchronization to Trust Manually Using ID Views"

7.1. MIGRATE FROM SYNCHRONIZATION TO TRUST AUTOMATICALLY USING IPA-WINSYNC-MIGRATE

> **IMPORTANT**
>
> The **ipa-winsync-migrate** utility is only available on systems running Red Hat Enterprise Linux 7.2 or later.

7.1.1. How Migration Using ipa-winsync-migrate Works

The **ipa-winsync-migrate** utility migrates all synchronized users from an AD forest, while preserving the existing configuration in the Winsync environment and transferring it into the AD trust. For each AD user created by the Winsync agreement, **ipa-winsync-migrate** creates an ID override in the Default Trust View (see Section 8.1, "Active Directory Default Trust View").

After the migration completes:

- The ID overrides for the AD users have the following attributes copied from the original entry in Winsync:

  - Login name (**uid**)

  - UID number (**uidnumber**)

  - GID number (**gidnumber**)

  - Home directory (**homedirectory**)

  - GECOS entry (**gecos**)

- The user accounts in the AD trust keep their original configuration in IdM, which includes:

  - POSIX attributes

  - User groups

  - Role-based access control rules

  - Host-based access control rules

- ○ SELinux membership

- ○ **sudo** rules

- The new AD users are added as members of an external IdM group.

- The original Winsync replication agreement, the original synchronized user accounts, and all local copies of the user accounts are removed.

7.1.2. How to Migrate Using ipa-winsync-migrate

Before you begin:

- Back up your IdM setup using the **ipa-backup** utility. See Backing Up and Restoring Identity Management in the *Linux Domain Identity, Authentication, and Policy Guide* .

  Reason: The migration affects a significant part of the IdM configuration and many user accounts. Creating a backup enables you to restore your original setup if necessary.

To migrate:

1. Create a trust with the synchronized domain. See Chapter 5, *Creating Cross-forest Trusts with Active Directory and Identity Management*.

2. Run **ipa-winsync-migrate** and specify the AD realm and the host name of the AD domain controller:

   ```
   # ipa-winsync-migrate --realm example.com --server ad.example.com
   ```

   If a conflict occurs in the overrides created by **ipa-winsync-migrate**, information about the conflict is displayed, but the migration continues.

3. Uninstall the Password Sync service from the AD server. This removes the synchronization agreement from the AD domain controllers.

See the ipa-winsync-migrate(1) man page for more details about the utility.

7.2. MIGRATE FROM SYNCHRONIZATION TO TRUST MANUALLY USING ID VIEWS

You can use ID views to manually change the POSIX attributes that AD previously generated for AD users.

1. Create a backup of the original synchronized user or group entries.

2. Create a trust with the synchronized domain. For information about creating trusts, see Chapter 5, *Creating Cross-forest Trusts with Active Directory and Identity Management* .

3. For every synchronized user or group, preserve the UID and GIDs generated by IdM by doing one of the following:

   - ○ Individually create an ID view applied to the specific host and add user ID overrides to the view.

   - ○ Create user ID overrides in the Default Trust View.

For details, see Defining a Different Attribute Value for a User Account on Different Hosts .

> **NOTE**
>
> Only IdM users can manage ID views. AD users cannot.

4. Delete the original synchronized user or group entries.

For general information on using ID views in Active Directory environments, see Chapter 8, *Using ID Views in Active Directory Environments*.

CHAPTER 8. USING ID VIEWS IN ACTIVE DIRECTORY ENVIRONMENTS

ID views enable you to specify new values for POSIX user or group attributes, as well as to define on which client host or hosts the new values will apply.

Integration systems other than Identity Management (IdM) sometimes generate UID and GID values based on an algorithm different than the algorithm used in IdM. By overriding the previously generated values to make them compliant with the values used in IdM, a client that used to be a member of another integration system can be fully integrated with IdM.

> **NOTE**
>
> This chapter only describes ID views functionality related to Active Directory (AD). For general information about ID views, see the Linux Domain Identity, Authentication, and Policy Guide.

You can use ID views in AD environments for the following purposes:

Overriding AD User Attributes, such as POSIX Attributes or SSH Login Details

See Section 8.3, "Using ID Views to Define AD User Attributes" for details.

Migrating from synchronization-based to trust-based integration

See Section 7.2, "Migrate from Synchronization to Trust Manually Using ID Views" for details.

Performing per-host group override of the IdM user attributes

See Section 8.4, "Migrating NIS Domains to IdM" for details.

8.1. ACTIVE DIRECTORY DEFAULT TRUST VIEW

8.1.1. What Is the Default Trust View

The Default Trust View is the default ID view always applied to AD users and groups in trust-based setups. It is created automatically when you establish the trust using **ipa-adtrust-install** and cannot be deleted.

Using the Default Trust View, you can define custom POSIX attributes for AD users and groups, thus overriding the values defined in AD.

Table 8.1. Applying the Default Trust View

	Values in AD	Default Trust View		Result
Login	ad_user	ad_user	→	ad_user
UID	111	222	→	222
GID	111	(no value)	→	111

NOTE

The Default Trust View only accepts overrides for AD users and groups, not for IdM users and groups. It is applied on the IdM server and clients and therefore only need to provide overrides for Active Directory users and groups.

8.1.2. Overriding the Default Trust View with Other ID Views

If another ID view applied to the host overrides the attribute values in the Default Trust View, IdM applies the values from the host-specific ID view on top of the Default Trust View.

- If an attribute is defined in the host-specific ID view, IdM applies the value from this view.

- If an attribute is not defined in the host-specific ID view, IdM applies the value from the Default Trust View.

The Default Trust View is always applied to IdM servers and replicas as well as to AD users and groups. You cannot assign a different ID view to them: they always apply the values from the Default Trust View.

Table 8.2. Applying a Host-Specific ID View on Top of the Default Trust View

	Values in AD	Default Trust View	Host-Specific View		Result
Login	ad_user	ad_user	(no value)	→	ad_user
UID	111	222	333	→	333
GID	111	(no value)	333	→	333

8.1.3. ID Overrides on Clients Based on the Client Version

The IdM masters always apply ID overrides from the Default Trust View, regardless of how IdM clients retrieve the values: using SSSD or using Schema Compatibility tree requests.

However, the availability of ID overrides from host-specific ID views is limited:

Legacy clients: RHEL 6.3 and earlier (SSSD 1.8 and earlier)

The clients can request a specific ID view to be applied.

To use a host-specific ID view on a legacy client, change the base DN on the client to: **cn=*id_view_name*,cn=views,cn=compat,dc=*example*,dc=*com***.

RHEL 6.4 to 7.0 (SSSD 1.9 to 1.11)

Host-specific ID views on the clients are not supported.

RHEL 7.1 and later (SSSD 1.12 and later)

Full support.

8.2. FIXING ID CONFLICTS

IdM uses *ID ranges* to avoid collisions of POSIX IDs from different domains. For details on ID ranges, see ID Ranges in the *Linux Domain Identity, Authentication, and Policy Guide* .

POSIX IDs in ID views do not use a special range type, because IdM must allow overlaps with other kinds of ID ranges. For example, AD users created through synchronization have POSIX IDs from the same ID range as IdM users.

POSIX IDs are managed manually in ID views on the IdM side. Therefore, if an ID collision occurs, fix it by changing the conflicting IDs.

8.3. USING ID VIEWS TO DEFINE AD USER ATTRIBUTES

With ID views, you can change the user attribute values defined in AD. For a complete list of the attributes, see Attributes an ID View Can Override .

For example: If you are managing a mixed Linux-Windows environment and want to manually define POSIX attributes or SSH login attributes for an AD user, but the AD policy does not allow it, you can use ID views to override the attribute values. When the AD user authenticates to clients running SSSD or authenticates using a compat LDAP tree, the new values are used in the authentication process.
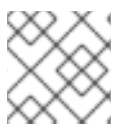
> **NOTE**
>
> Only IdM users can manage ID views. AD users cannot.

The process for overriding the attribute values follows these steps:

1. Create a new ID view.

2. Add a user ID override in the ID view, and specify the require attribute value.

3. Apply the ID view to a specific host.

For details on how to perform these steps, see Defining a Different Attribute Value for a User Account on Different Hosts in the *Linux Domain Identity, Authentication, and Policy Guide* .

8.4. MIGRATING NIS DOMAINS TO IDM

If you are managing a Linux environment and want to migrate disparate NIS domains with different UIDs and GIDs into a modern identity management solution, you can use ID views to set host specific UIDs and GIDs for existing hosts to prevent changing the permissions on existing files and directories.

The process for the migration follows these steps:

1. Create the users and groups in the IdM domain. For details, see

   - Adding Stage or Active Users

   - Adding and Removing User Groups

2. Use ID views for existing hosts to override the IDs IdM generated during the user creation:

   1. Create an individual ID view.

   2. Add ID overrides for the users and groups to the ID view.

3. Assign the ID view to the specific hosts.

   For details, see Defining a Different Attribute Value for a User Account on Different Hosts .

3. Installing and Uninstalling Identity Management Clients in the *Linux Domain Identity, Authentication, and Policy Guide*.

4. Decommission the NIS domains.

8.5. CONFIGURATION OPTIONS FOR USING SHORT NAMES TO RESOLVE AND AUTHENTICATE USERS AND GROUPS

This section describes configuration options enabling you to use short user or group names instead of the **user_name@domain** or **domain\user_name** fully qualified names format to resolve and authenticate users and groups in an Active Directory (AD) environment. You can configure this:

- in Identity Management (IdM) that trusts AD

- on Red Hat Enterprise Linux joined to an AD using SSSD

8.5.1. How Domain Resolution Works

You can use the **domain resolution order** option to specify the order in which a list of domains is searched to return a match for a given user name. You can set the option:

- on the server. See:

    - Section 8.5.2.1, "Setting the Domain Resolution Order Globally"

    - Section 8.5.2.2, "Setting the Domain Resolution Order for an ID view"

- on the client. See Section 8.5.3, "Configuring the Domain Resolution Order on an IdM Client"

In environments with an Active Directory trust, applying one or both of the server-based options is recommended.

From the perspective of a particular client, the **domain resolution order** option can be set in more than one of the three locations above. The order in which a client consults the three locations is:

1. the local **sssd.conf** configuration

2. the id view configuration

3. the global IdM configuration

Only the domain resolution order setting found first will be used.

In environments in which Red Hat Enterprise Linux is directly integrated into an AD, you can only set the domain resolution order on the client.

> **NOTE**
>
> You must use qualified names if:
>
> - A user name exists in multiple domains
>
> - The SSSD configuration includes the **default_domain_suffix** option and you want to make a request towards a domain not specified with that option

8.5.2. Configuring the Domain Resolution Order on an Identity Management Server

Select the server-based configuration if a large number of clients in a domain or subdomain should use an identical domain resolution order.

8.5.2.1. Setting the Domain Resolution Order Globally

Select this option for setting the domain resolution order to all the clients in the trust. In order to do this, use the **ipa config-mod** command. For example, in an IdM domain that trusts an AD forest with multiple child domains:

```
$ ipa config-mod --domain-resolution-
order='idm.example.com:ad.example.com:subdomain1.ad.example.com:subdomain2.ad.example.com
'
Maximum username length: 32
Home directory base: /home
...
Domain Resolution Order:
idm.example.com:ad.example.com:subdomain1.ad.example.com:subdomain2.ad.example.com
...
```

With the domain resolution order set in this way, users from both the IdM domain and from the trusted AD forest can log in using short names only.

8.5.2.2. Setting the Domain Resolution Order for an ID view

Select this option to apply the setting to the clients in a specific domain.

For example, on your subdomain server, *server.idm.example.com*, you observe many more logins from the *subdomain2.ad.example.com* subdomain than from *subdomain1.ad.example.com*. The global resolution order states, however, that the *subdomain1.ad.example.com* subdomain user database is tried out before *subdomain2.ad.example.com* when resolving user names. To set a different order for certain servers, set up a domain resolution order for a specific view:

1. Create an ID view with the **domain resolution order** option set:

   ```
   $ ipa idview-add example_view --desc "ID view for custom shortname resolution on
   server.idm.example.com" --domain-resolution-order
   subdomain2.ad.example.com:subdomain1.ad.example.com
   --------------------------------
   Added ID View "example_view"
   --------------------------------
   ID View Name: example_view
   Description: ID view for custom shortname resolution on server.idm.example.com
   Domain Resolution Order: subdomain2.ad.example.com:subdomain1.ad.example.com
   ```

2. Apply the view on the clients. For example:

```
$ ipa idview-apply example_view --hosts server.idm.example.com
-----------------------------------
Applied ID View "example_view"
-----------------------------------
hosts: server.idm.example.com
---------------------------------------------
Number of hosts the ID View was applied to: 1
---------------------------------------------
```

For further information on ID views, see Chapter 8, *Using ID Views in Active Directory Environments* .

8.5.3. Configuring the Domain Resolution Order on an IdM Client

Set the domain resolution order on the client if you want to set it on a low number of clients or if the clients are directly connected to AD.

Set the **domain_resolution_order** option, in the [sssd] section, in the **/etc/sssd/sssd.conf** file, for example:

```
domain_resolution_order = subdomain1.ad.example.com, subdomain2.ad.example.com
```

For further information on configuring the **domain_resolution_order** option, see the sssd.conf(5) man page.

APPENDIX A. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Enterprise Linux.

Revision 7.0-51   **Thu Mar 4 2021**   **Florian Delehaye**
7.9 GA version of the guide. Added a new section about adjusting DNA ID ranges manually.

Revision 7.0-50   **Wed May 27 2020**   **Florian Delehaye**
Several fixes and updates.

Revision 7.0-49   **Tue Aug 06 2019**   **Marc Muehlfeld**
Document version for 7.7 GA publication.

Revision 7.0-48   **Wed Jun 05 2019**   **Marc Muehlfeld**
Updated *Configuring Trust Agents*, added *How the AD Provider Handles Trusted Domains* and *Changing the Format of User Names Displayed by SSSD*.

Revision 7.0-47   **Tue Apr 08 2019**   **Marc Muehlfeld**
Several minor fixes and updates.

Revision 7.0-46   **Mon Oct 29 2018**   **Filip Hanzelka**
Preparing document for 7.6 GA publication.

Revision 7.0-45   **Mon Jun 25 2018**   **Filip Hanzelka**
Added *Switching Between SSSD and Winbind for SMB Share Access*.

Revision 7.0-44   **Thu Apr 5 2018**   **Filip Hanzelka**
Preparing document for 7.5 GA publication.

Revision 7.0-43   **Wed Feb 28 2018**   **Filip Hanzelka**
Updated *GPO Settings Supported by SSSD*.

Revision 7.0-42   **Mon Feb 12 2018**   **Aneta Šteflová Petrová**
Updated *Creating a Two-Way Trust with a Shared Secret*.

Revision 7.0-41   **Mon Jan 29 2018**   **Aneta Šteflová Petrová**
Minor fixes.

Revision 7.0-40   **Fri Dec 15 2017**   **Aneta Šteflová Petrová**
Minor fixes.

Revision 7.0-39   **Mon Dec 6 2017**   **Aneta Šteflová Petrová**
Updated *Using Samba for Active Directory Integration*.

Revision 7.0-38   **Mon Dec 4 2017**   **Aneta Šteflová Petrová**
Updated *DNS and Realm Settings* for trusts.

Revision 7.0-37   **Mon Nov 20 2017**   **Aneta Šteflová Petrová**
Updated *Creating a Two-Way Trust with a Shared Secret*.

Revision 7.0-36   **Mon Nov 6 2017**   **Aneta Šteflová Petrová**
Minor fixes.

Revision 7.0-35   **Mon Oct 23 2017**   **Aneta Šteflová Petrová**
Updated *Active Directory Entries and POSIX Attributes* and *Configuring an AD Domain with ID Mapping as a Provider for SSSD*.

Revision 7.0-34	**Mon Oct 9 2017**	**Aneta Šteflová Petrová**

Added *Configuration Options for Using Short Names*. Updated *Trust Controllers and Trust Agents*.

Revision 7.0-33	**Tue Sep 26 2017**	**Aneta Šteflová Petrová**

Updated the autodiscovery section in the SSSD chapter. Added two sections on configuring trusted domains.

Revision 7.0-32	**Tue Jul 18 2017**	**Aneta Šteflová Petrová**

Document version for 7.4 GA publication.

Revision 7.0-31	**Tue May 23 2017**	**Aneta Šteflová Petrová**

A minor fix for About Security ID Mapping.

Revision 7.0-30	**Mon Apr 24 2017**	**Aneta Šteflová Petrová**

Minor fixes for Defining Windows Integration.

Revision 7.0-29	**Mon Apr 10 2017**	**Aneta Šteflová Petrová**

Updated Direct Integration.

Revision 7.0-28	**Mon Mar 27 2017**	**Aneta Šteflová Petrová**

Moved Allowing Users to Change Other Users' Passwords Cleanly to the Linux Domain Identity guide as Enabling Password Reset. Updated Supported Windows Platforms for trusts. Fixed broken links. Other minor updates.

Revision 7.0-27	**Mon Feb 27 2017**	**Aneta Šteflová Petrová**

Updated port requirements for trusts. Minor restructuring for trust and sync. Other minor updates.

Revision 7.0-26	**Wed Nov 23 2016**	**Aneta Šteflová Petrová**

Added ipa-winsync-migrate. Minor fixes for the trust, SSSD, and synchronization chapters.

Revision 7.0-25	**Tue Oct 18 2016**	**Aneta Šteflová Petrová**

Version for 7.3 GA publication.

Revision 7.0-24	**Thu Jul 28 2016**	**Marc Muehlfeld**

Updated diagrams, added Kerberos flags for services and hosts, other minor fixes.

Revision 7.0-23	**Thu Jun 09 2016**	**Marc Muehlfeld**

Updated the synchronization chapter. Removed the Kerberos chapter. Other minor fixes.

Revision 7.0-22	**Tue Feb 09 2016**	**Aneta Petrová**

Updated realmd, removed index, moved a part of ID views to the Linux Domain Identity guide, other minor updates.

Revision 7.0-21	**Fri Nov 13 2015**	**Aneta Petrová**

Version for 7.2 GA release with minor updates.

Revision 7.0-20	**Thu Nov 12 2015**	**Aneta Petrová**

Version for 7.2 GA release.

Revision 7.0-19	**Fri Sep 18 2015**	**Tomáš Čapek**

Updated the splash page sort order.

Revision 7.0-18	**Thu Sep 10 2015**	**Aneta Petrová**

Updated the output format.

Revision 7.0-17	**Mon Jul 27 2015**	**Aneta Petrová**

Added GPO-based access control, a number of other minor changes.

Revision 7.0-16	**Thu Apr 02 2015**	**Tomáš Čapek**

Added ipa-advise, extended CIFS share with SSSD, admonition for the Identity Management for UNIX extension.

Revision 7.0-15   Fri Mar 13 2015   **Tomáš Čapek**
Async update with last-minute edits for 7.1.

Revision 7.0-13   Wed Feb 25 2015   **Tomáš Čapek**
Version for 7.1 GA release.

Revision 7.0-11   Fri Dec 05 2014   **Tomáš Čapek**
Rebuild to update the sort order on the splash page.

Revision 7.0-7   Mon Sep 15 2014   **Tomáš Čapek**
Section 5.3 Creating Trusts temporarily removed for content updates.

Revision 7.0-5   June 27, 2014   **Ella Deon Ballard**
Improving Samba+Kerberos+Winbind chapters.

Revision 7.0-4   June 13, 2014   **Ella Deon Ballard**
Adding Kerberos realm chapter.

Revision 7.0-3   June 11, 2014   **Ella Deon Ballard**
Initial release.