



Red Hat Enterprise Linux 7 Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Ella Deon Ballard

Red Hat Enterprise Linux 7 Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Ella Deon Ballard
dlackey@redhat.com

Legal Notice

Copyright © 2014 Red Hat.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

Identity and policy management — for both users and machines — is a core function for almost any enterprise environment. Identity Management provides a way to create an identity domain that allows machines to enroll to a domain and immediately access identity information required for single sign-on and authentication services, as well as policy settings that govern authorization and access. This manual covers all aspects of installing, configuring, and managing Identity Management domains, including both servers and clients. This guide is intended for IT and systems administrators.

Table of Contents

Preface	3
1. Information for Managing Identity and Authentication Policies in Linux	3
2. Audience and Purpose	3
3. Giving Feedback	3
4. Document Change History	4
Chapter 1. Ways to Integrate Active Directory and Linux Environments	5
1.1. Defining Windows Integration	5
1.2. Small Environments: Using Windows as an Identity Source	6
1.3. Small Environments: Enrolling Individual Clients	6
1.4. Big Environments: Synchronizing Users	7
1.5. Big Environments: Trusted Realms	7
Part I. Adding a Single Linux System to an Active Directory Domain	9
Chapter 2. Using Active Directory as an Identity Provider for SSSD	10
2.1. About SSSD	10
2.2. Environments for SSSD	12
2.3. How SSSD Integrates with an Active Directory Environment	12
2.4. Configuring an Active Directory Domain with ID Mapping	16
2.5. Configuring an Active Directory Domain with POSIX Attributes	19
2.6. Configuring Active Directory as an LDAP Domain	23
2.7. Additional Configuration Examples	26
Chapter 3. Using realmd to Connect to an Active Directory Domain	31
3.1. About realmd	31
3.2. realmd Commands	31
3.3. Discovering and Joining Active Directory Domains	32
3.4. Managing User Logins from Active Directory	34
3.5. Adding Default User Configuration	34
3.6. Additional Configuration for the Active Directory Domain Entry	35
Chapter 4. Using Samba, Kerberos, and Winbind	37
4.1. About Samba and Active Directory Authentication	37
4.2. Summary of Configuration Files, Options, and Packages	40
4.3. Configuring a Domain Member Using authconfig	42
Part II. Integrating a Linux Domain with an Active Directory Domain	48
Chapter 5. Creating Cross-Realm Trusts with Active Directory and Identity Management	
5.1. The Meaning of "Trust"	49
5.2. Environment and Machine Requirements to Set up Trusts	59
5.3. Creating Trusts	62
5.4. Creating IdM Groups for Active Directory Users	84
5.5. Maintaining Trusts	86
5.6. Verifying That IdM Machines Have Resolvable Names	90
5.7. Setting PAC Types for Services	91
5.8. Using SSH from Active Directory Machines for IdM Resources	94
5.9. Using Trust with Kerberized Web Applications	95
Chapter 6. Setting up Kerberos Cross-Realm Authentication	97
6.1. A Trust Relationship	97
6.2. Setting up a Realm Trust	100
Chapter 7. Synchronizing Active Directory and Identity Management Users	101
7.1. Supported Windows Platforms	101
7.2. About Active Directory and Identity Management	101

7.3. About Synchronized Attributes	103
7.4. Setting up Active Directory for Synchronization	107
7.5. Managing Synchronization Agreements	107
7.6. Managing Password Synchronization	115
Index	120

Preface

Many IT environments are heterogeneous. In a mixed environment, there has to be some way to join systems to the larger domain, either directly as clients or by creating transparency between two peers domains.

This is especially important in environments where one domain (usually Active Directory) manages users, while another domain (such as a Linux domain through Identity Management) manages backend systems or a development or production environment.

This guide covers different default applications within Red Hat Enterprise Linux which can help a Linux system or an entire Linux domain integrate with an Active Directory environment.

1. Information for Managing Identity and Authentication Policies in Linux

Managing user and system identities, authentication settings, and application policies is one of the central responsibilities of system administration. Even if users are defined within an Active Directory environment, it is still critical that those users have the appropriate access controls and policies in place as they access Linux-based services and resources. Those policies are defined within the Linux environment.

There are two related guides for Red Hat Enterprise Linux 7 which deal with different scenarios related to identity, authentication, and policy management:

- ▶ For managing identity and authentication services at the system-level, see the *System-Level Authentication Guide*.
- ▶ For configuring and managing a Linux domain and centralizing system policies, which uses Red Hat Enterprise Linux Identity Management, see the *Linux Domain Administration Guide*.

2. Audience and Purpose

There are a number of different paths to integrate Linux systems within a Windows environment. With the different security and identity applications available within Red Hat Enterprise Linux, outlined both in this guide and in the *System-Level Authentication Guide* and the *Linux Domain Administration Guide*, the configuration options are almost limitless and depend on the needs of an individual system or environment.

This guide covers major applications and major integration options, which will be useful in many different environments. This is not a definitive or comprehensive source, and the true integration solution may be a more complex mix of different scenarios. Use the options here as guidelines to plan how to integrate your different environments.

This guide is written for systems administrators and IT staff with a working knowledge of Linux systems and applications, but the core audience is for Windows administrators who are planning integration.

3. Giving Feedback

If there is any error in this book or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for IdM through Bugzilla, <http://bugzilla.redhat.com/bugzilla>. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

1. Select the Red Hat group and the Red Hat Enterprise Linux 7 product.

2. Set the component to **doc-Enterprise_Identity_Management_Guide**.
3. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.

For enhancements, put in what information needs to be added and why.

4. Give a clear title for the bug. For example, "**Incorrect command example for setup script options**" is better than "**Bad example**".

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at docs@redhat.com.

1. Select the Community group and the freeIPA product.
2. Set the component to Documentation.
3. Set the version number to 3.2.
4. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.

For enhancements, put in what information needs to be added and why.

5. Give a clear title for the bug. For example, "**Incorrect command example for setup script options**" is better than "**Bad example**".

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact the Fedora docs team at docs@lists.fedoraproject.org.

4. Document Change History

Revision 7.0-5.405	Thu Jul 7 2014	Rüdiger Landmann
Add html-single and epub formats		
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.		

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 ultimate desired interaction between the Linux environment and the Windows environment. It could mean that individual Linux systems are enrolled into a Windows domain, or it could mean that a Linux domain is configured to be a peer of the Windows domain, or it could simply mean that information is copied between environments.

There are several major potential points of contact between a Windows domain and Linux systems, and 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, Windows only or both Linux and Windows?
- ▶ How are users authenticated on a Linux system — locally or through Windows?
- ▶ How is group membership configured for groups? How is that group membership determined?
- ▶ Will users authenticate using a simple username/password, Kerberos tickets, certificates, or a combination of methods?
- ▶ How are user attributes managed? Specifically, for Linux-required POSIX attributes, are those attributes set in the Windows domain, configured locally on the Linux system, or (for UID/GID numbers and Windows SIDs) dynamically mapped?
- ▶ What users will be accessing what resources? Will Windows-defined users access Linux resources? Will Linux-defined users

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, with user identities, is *how* to obtain that 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 (e.g., some 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

- ▶ Is there a single DNS domain? Are there subdomains?
- ▶ How will system hostnames be resolved?

- ▶ How will service discovery be configured?

Security Policies

- ▶ Where are access control instructions set?
- ▶ What 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 (e.g., the DNS service is changed), 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 what elements in the domains are integrated, is how that integration is maintained. If a particular means of integration is heavily manual, yet the environment has a large number of systems which are frequently updated, then that one means may not work for that environment from a maintenance standpoint.

1.2. Small Environments: Using Windows as an Identity Source

Probably the lightest touch for Windows integration is to have the Linux system use Windows as an identity store, but to otherwise maintain all service, security, and other configuration within the local system.

There are two services available to configure the local system:

- ▶ System Security Services Daemon (SSSD), using Active Directory as an identity provider
- ▶ **realmd**, to configure SSSD (or, more rarely Winbind) as Active Directory as an identity provider

Both SSSD and **realmd** use Windows for pass-through authentication. The user identities reside in the Windows side, and there can be some limited configuration for groups or authorization, but most configuration — including security policies like SELinux — remain within the ownership of the local system.

There is also a lot of latitude in how user attributes are defined and used. For example, user IDs can be created locally on the Linux system, mapped to Windows SIDs, or taken directly from the Windows configuration. This is also true for login shells, group membership, home directories, and other user settings.

Both SSSD and **realmd** are local services, with local configuration files. Provisioning systems (such as Foreman or Puppet) can be used to maintain these files to try to lower the administrative overhead of making changes. Ultimately, though, each system has to be updated individually to change or add integration settings. This means that using SSSD or **realmd** alone are best in IT environments with a small number of Linux systems.

1.3. Small Environments: Enrolling Individual Clients

An alternative to using Active Directory as an external identity store is to simply enroll a system within a Windows domain.

There are several different paths to enroll a Linux system in a Windows domain:

- ▶ Winbind and Samba, to enroll a system directly in a Windows domain

- ▶ Local PAM and Kerberos configuration, to enroll a system directly in a Windows domain

As with using SSSD and realmd, using either Winbind or PAM/Kerberos configuration requires local changes to the system. These can be managed through a provisioning system, but there is no central authority defining the configuration. Additionally, it requires external servers (either a Samba server or Kerberos KDC) within the Linux environment to integrate with the Windows environment. The Linux environment is more constrained, as well, with less use for or flexibility in managing user attributes.

1.4. Big Environments: Synchronizing Users

Red Hat Enterprise Linux has a Linux domain tool included by default, Identity Management. This creates a Linux domain and centralizes the maintenance of Linux systems policies (SELinux, password policy, sudo, automount, host-based access controls, and others). Identity Management also creates and maintains a central identity store, which is used by all Linux clients enrolled in the domain.

If there are a large number of configured Linux users in Identity Management (meaning, Windows is not the only user directory), then it is possible to simply copy users between Windows and Linux directories.

Synchronization has some benefits:

- ▶ The Linux-based users exist within the Windows domain and can be configured to access Windows resources.
- ▶ The sync configuration is relatively simple.
- ▶ Windows users can be Identity Management users and administrators.

However, it also has some limitations:

- ▶ It requires an additional Password Sync service to be installed on a Windows domain controller.
- ▶ There is a limited set of user attributes available for synchronization.
- ▶ Groups are not synchronized in Identity Management ^[1].
- ▶ One of the directories must still be the master directory or there can be conflicts in adding and removing users and in the attributes in entries.
- ▶ There can be an overall performance hit to the IdM domain or to the IdM server with synchronization.
- ▶ Synchronization is limited to a single IdM server and a single Active Directory domain.

1.5. Big Environments: Trusted Realms

Kerberos has a concept called *trust* where specific agreements can be put in place so that the member of one Kerberos realm are trusted as if they belong to another Kerberos realm, and this grants transparent access to resources.

Identity Management in Red Hat Enterprise Linux expands on a basic Kerberos v5 trust to include a mutual configuration with a DNS domain. This is the closest configuration to a full domain-level integration with an Active Directory environment.

Linux systems and identities retain all of the flexibility of configuration: maintenance for the Linux domain, including security policies, remains within the Linux domain. It is not dictated by the Windows domain (as with Winbind/Samba) or limited as with synchronization.

The Identity Management domain can use Windows group assignments to create its own, local security policies, which gives administrative control to both the Windows and the Linux environments for managing users and authorization.

Additionally, Identity Management supports multiple domains, which can be useful if there are different Windows environments, subdomains within the Active Directory forest, or transitive trusts with between Windows domains.

The main drawback to using trusts with Identity Management is the complexity of the DNS configuration. Identity Management manages its own DNS domain, either as a separate domain or as a subdomain of the Active Directory DNS domain. If DNS is not properly configured in both Identity Management and Active Directory, there can be problems resolving hostnames, handling queries, and even with running some services.

[1] Synchronization is also available in Red Hat Directory Server, and groups are synchronized there.

Part I. Adding a Single Linux System to an Active Directory Domain

Chapter 2. Using Active Directory as an Identity Provider for SSSD

The System Security Services Daemon (SSSD) provides access to different identity and authentication providers. This service ties a local system to a larger backend system. That can be a simple LDAP directory, domains for Active Directory or IdM in Red Hat Enterprise Linux, or Kerberos realms.

SSSD configures a way to connect to an identity store to retrieve authentication information and then uses that to create a local cache of users and credentials. With some types of identity providers — including Active Directory — SSSD also pulls in group and authorization information.

2.1. About SSSD

SSSD is an intermediary between local clients and any configured data store. This relationship brings a number of benefits for administrators:

- ▶ *Reducing the load on identification/authentication servers.* Rather than having every client service attempt to contact the identification server directly, all of the local clients can contact SSSD which can connect to the identification server or check its cache.
- ▶ *Permitting offline authentication.* SSSD can optionally keep a cache of user identities and credentials that it retrieves from remote services. This allows users to authenticate to resources successfully, even if the remote identification server is offline or the local machine is offline.
- ▶ *Using a single user account.* Remote users frequently have two (or even more) user accounts, such as one for their local system and one for the organizational system. This is necessary to connect to a virtual private network (VPN). Because SSSD supports caching and offline authentication, remote users can connect to network resources simply by authenticating to their local machine and then SSSD maintains their network credentials.

SSSD caches those users and credentials, so if the local system or the identity provider go offline, the user credentials are still available to services to verify.

2.1.1. SSSD Configuration

SSSD is a local service which connects a system to a larger, external identity service. This is done by configuring *domains* in the SSSD configuration file. Each domain represents a different, external data source. Domains always represent an *identity provider* which supplies user information and, optionally, define other providers for different kinds of operations, such as authentication or password changes. (The identity provider can also be used for all operations, if all operations are performed within a single domain or server.)



NOTE

SSSD allows all user identities to be created and maintained in a separate, external identity source. For Windows integration, then the Active Directory domain can be used to manage user accounts (as it is with most environments). Local system users do not need to be created or synced with user accounts in Active Directory — SSSD uses those Windows identities and lets those Windows users access the local system and local services.

SSSD also defines which services on the system use SSSD for credentials caching and user accounts. These relate to foundational security services such as the Name Switch Service (NSS) and pluggable authentication modules (PAM), which are then used by higher-level applications.

Example 2.1. Simple sssd.conf File

```
[sssd]
domains = LOCAL
services = nss
config_file_version = 2

[nss]
filter_groups = root
filter_users = root

[domain/WINDOWS]
id_provider = ad
auth_provider = ad
access_provider = ad
```

2.1.2. Active Directory Domain Configuration

The most basic type of domain is an LDAP domain. Any LDAPv3 directory server can be configured as an LDAP identity provider for an SSSD domain. Some specialty LDAP services have additional, specific configuration, which can either simplify service-specific configuration or supply service-specific functionality. One of those identity provider types is for Active Directory.

As shown in [Example 2.1, “Simple sssd.conf File”](#), the SSSD configuration file has three major sections: the first configures the SSSD service (`[sssd]`), the second configures system services which will use SSSD as an identity cache (such as `[nss]` and `[pam]`), and the third section configures the identity domains (`[domain/NAME]`).

By default, only an identity provider really needs to be configured — the identity provider is used for the authentication, access (authorization), and password providers if no other types or servers are identified. Active Directory can be configured as any kind of provider using the `ad` option.

```
[domain/ADEXAMPLE]
id_provider = ad
auth_provider = ad
access_provider = ad
chpass_provider = ad

ad_server = dc1.example.com
ad_hostname = client.example.com
ad_domain = example.com
```

The connection information is required to identify what Active Directory server to use.

Past that basic configuration, the Active Directory identity provider can be configured specifically for the Active Directory environment and specific features, such as how to use POSIX attributes or mapping for Windows SIDs on the local system, failover servers, and account information such as home directories.

All of the LDAP domain parameters are available to the Active Directory provider, as well as Active Directory-specific configuration parameters. The complete lists are available in the [sssd-ldap](#) and [sssd-ad](#) man pages.

There are a number of options in the generic LDAP provider configuration which can be used to configure an Active Directory provider. Using the `ad` value is a short-cut which automatically pulls in the parameters and values to configure a given provider for Active Directory.

For example, the shortcut for an access provider is:

```
access_provider = ad
```

Using generic LDAP parameters, that configuration expands to:

```
access_provider = ldap
ldap_access_order = expire
ldap_account_expire_policy = ad
```

Those settings are all set implicitly by using the **ad** provider type.

2.2. Environments for SSSD

A number of Linux system services can leverage SSSD for caching user identities and configuring backend identity stores. Additionally, applications can be written or configured to use SSSD or services (like NSS or PAM) managed by SSSD for identities. This means that SSSD is ideal for identity integration for environments using NSS, PAM, automount, SSH, and sudo or for applications which require access to an externally-managed identity store.

SSSD replaces older identity management services which were used for Windows integration, including NIS and Winbind.

SSSD is a local system service, so configuring it manually is only feasible for environments with a small number of systems.

There are some tools which can do the initial configuration for the SSSD Active Directory domain. **realm** edits all of the underlying configuration files automatically, but this is still a system-level tool. It simplifies editing the configuration, but must be run separately on each system. The **ipa-client-install** tool also configures SSSD appropriately and an IdM server can configure a client to work with an Active Directory-IdM trust, but that requires a configured and functioning IdM Linux domain and an already-configured trust environment.

2.3. How SSSD Integrates with an Active Directory Environment

2.3.1. About Active Directory Identities on the Local System

Active Directory can replicate user entries and attributes from its local directory into a *global catalog*, which makes the information available to other domains within the forest. SSSD checks this global catalog for information about users and groups, so information is not limited to a single Active Directory domain or subdomain — SSSD, too, has access to all user data for all domains within the topology.

SSSD, then, can be used by applications which need to query the Active Directory global catalog for user or group information.

There are inherent structural differences between how Windows and Linux handle system users and in the user schemas used in Active Directory and standard LDAPv3 directory services. When using an Active Directory identity provider with SSSD to manage system users, it is necessary to reconcile the Active Directory-style user to the new SSSD user. There are two ways to do this:

- Using ID mapping on SSSD to create a map between Active Directory security IDs (SIDs) and the generated UIDs on Linux.

ID mapping is the simplest option for most environments because it requires no additional packages or configuration on Active Directory.

- Using Services for Unix to insert POSIX attributes on Windows user and group entries, and then having those attributes pulled into PAM/NSS.

This requires more configuration and information within the Active Directory environment, but it gives more administrative control over the specific UID/GID values (and other POSIX attributes).

2.3.1.1. About Security ID Mapping

2.3.1.1.1. The Mechanism of ID Mapping

Linux/Unix systems use a local user ID number and group ID number to identify users on the system. These UID:GID numbers are a simple integer, such as `501:501`. These numbers are simple because they are always created and administered locally, even for systems which are part of a larger Linux/Unix domain.

Microsoft Windows and Active Directory use a different user ID structure to identify users, groups, and machines. Each ID is constructed of different segments that identify the security version, the issuing authority type, the machine, and the identity itself. For example:

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

The third through sixth blocks are the machine identifier:

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

The last block is the *relative identifier* (RID) which identifies the specific entity:

```
S-1-5-21-3623811015-3361044348-30300820-1013
```

A range of possible ID numbers are always assigned to SSSD. (This is a local range, so it is the same for every machine.)

```
|_____|
|
| minimum ID                max ID
```

This range is divided into 10,000 sections (by default), with each section allocated 200,000 IDs.

```
| slice 1 | slice 2 | ... |
|_____|_____|_____|
|
| minimum ID                max ID
```

When a new Active Directory domain is detected, the SID is hashed. Then, SSSD takes the modulus of the hash and the number of available sections to determine which ID section to assign to the Active Directory domain. This is a reliably consistent means of assigning ID sections, so the same ID range is assigned to the same Active Directory domain on most client machines.

```
| Active | Active | |
|Directory|Directory|
|domain 1|domain 2| ... |
|
| slice 1 | slice 2 | ... |
|_____|_____|_____|
|
| minimum ID                max ID
```

**Note**

While the method of assigning ID sections is consistent, **ID mapping is based on the order that an Active Directory domain is encountered on a client machine** — so it may not result in consistent ID range assignments on all Linux client machines. If consistency is required, then consider disabling ID mapping and using explicit POSIX attributes.

2.3.1.1.2. ID Mapping Parameters

ID mapping is enabled in two parameters, one to enable the mapping and one to load the appropriate Active Directory user schema:

```
Idap_id_mapping = True
Idap_schema = ad
```

**Note**

When ID mapping is enabled, the *uidNumber* and *gidNumber* attributes are ignored. This prevents any manually-assigned values. If *any* values must be manually assigned, then *all* values must be manually assigned, and ID mapping should be disabled.

2.3.1.1.3. Mapping Users

When an Active Directory user attempts to log into a local system service for the first time, an entry for that user is created in the SSSD cache. The remote user is set up much like a system user:

- ▶ A system UID is created for the user based on his SID and the ID range for that domain.
- ▶ A GID is created for the user, which is identical to the UID.
- ▶ A private group is created for the user.
- ▶ A home directory is created, based on the home directory format in the `sssd.conf` file.
- ▶ A shell is created, according to the system defaults or the setting in the `sssd.conf` file.
- ▶ If the user belongs to any groups in the Active Directory domain, then, using the SID, SSSD adds the user to those groups on the Linux system.

2.3.1.2. About SSSD and POSIX Attributes

Active Directory can be configured to create and store POSIX attributes such as *uidNumber*, *gidNumber*, *unixHomeDirectory*, and *loginShell*. As with all user attributes, these are originally stored in the local domain, but they can be replicated to the global catalog — and once they are in the global catalog, they are available to SSSD and any application which uses SSSD for its identity information.

**IMPORTANT**

When SSSD uses the POSIX attributes directly, they must be published to the Active Directory global catalog. SSSD queries the global catalog for user information.

When POSIX attributes are already defined in Active Directory, then it is not acceptable to use the SID/UID

mapping as described in [Section 2.3.1.1, “About Security ID Mapping”](#). The UID and GID numbers are already defined, and mapping creates new, different numbers. The best solution in that situation is to use the UID and GID numbers as defined in Active Directory and then apply that to the local Linux accounts managed by SSSD.

To use existing POSIX attributes, two things must be configured:

- ▶ The POSIX attributes must be published to Active Directory's global catalog.
- ▶ ID mapping (**ldap_id_mapping** in the Active Directory domain entry) must be disabled in SSSD.

```
ldap_id_mapping = False
```

2.3.1.3. Active Directory Users and Range Retrieval Searches

Microsoft Active Directory has an attribute, **MaxValueRange**, which sets a limit on how many values for a multi-valued attribute will be returned. This is the *range retrieval* search extension. Essentially, this runs multiple mini-searches, each returning a subset of the results within a given range, until all matches are returned.

For example, when doing a search for the **member** attribute, each entry could have multiple values, and there can be multiple entries with that attribute. If there are 2000 matching results (or more), then **MaxValueRange** limits how many are displayed at once; this is the value range. The given attribute then has an additional flag set, showing which range in the set the result is in:

```
attribute:range=low-high:value
```

For example, results 100 to 500 in a search:

```
member;range=99-499: cn=John Smith...
```

This is described in the Microsoft documentation at <http://msdn.microsoft.com/en-us/library/cc223242.aspx>.

SSSD supports range retrievals with Active Directory providers as part of user and group management, without any additional configuration.

However, some LDAP provider attributes which are available to configure searches — such as **ldap_user_search_base** — are not performant with range retrievals. Be cautious when configuring search bases in the Active Directory provider domain and consider what searches may trigger a range retrieval.

2.3.2. Linux Clients and Active Directory DNS Sites

SSSD connects a local Linux system to a larger Active Directory environment. This requires that SSSD have an awareness of possible configurations within the Active Directory forest and work with them so that the Linux client is cleanly integrated.

Active Directory forests can be very large, with numerous different domain controllers, domains and subdomains, and physical sites. To increase client performance, Active Directory uses a special kind of DNS record to identify domain controllers within the same domain but at different physical locations. Clients connect to the closest domain controller.

**NOTE**

Microsoft has a tech brief at <http://technet.microsoft.com/es-es/library/cc759550%28v=ws.10%29.aspx> which describes how DNS and Active Directory work together.

Active Directory extends normal DNS SRV records to identify a specific physical location or site for its domain controllers. Clients (such as SSSD) can determine which domain controllers to use based on their own site configuration.

SSSD can determine which domain controller to use by querying the Active Directory domain first for its site configuration, and then for the domain controller DNS records.

1. SSSD attempts to connect to the Active Directory domain and looks up any available domain controller through normal DNS discovery.
2. It retrieves a list of primary and fallback servers.
3. SSSD sends a special CLDAP ping to any domain controller. The ping is really an LDAP search which looks for the DNS domain, domain SID, and version:

```
(&(&(DnsDomain=ad.domain)(DomainSid=S-1-5-21-1111-2222-3333))
(NtVer=0x01000016))
```

This is used to retrieve the information about the client's site (if one is configured).

4. If a site is configured for the client, then the reply contains extended DNS SRV records for the primary server, containing the site name (*site-name._sites.*):

```
_service._protocol.site-name._sites.domain.name
```

The backup server record is also sent, as a standard SRV record:

```
_service._protocol.domain.name
```

If no site is configured, then a standard SRV record is sent for all primary and backup servers.

2.4. Configuring an Active Directory Domain with ID Mapping

When configuring an Active Directory domain, the simplest configuration is to use the **ad** value for all providers (identity, access, password). Also, load the native Active Directory schema for user and group entries, rather than using the default RFC 2307.

Other configuration is available in the general LDAP provider configuration ([sssd-ldap](#)) and Active Directory-specific configuration ([sssd-ad](#)). This includes setting LDAP filters for a specific user or group subtree, filters for authentication, and values for some account settings. Some additional configuration is covered in [Section 2.7, "Additional Configuration Examples"](#).

1. Make sure that both the Active Directory and Linux systems have a properly configured environment.
 - ▶ Name resolution must be properly configured, particularly if service discovery is used with SSSD.
 - ▶ The clocks on both systems must be in sync for Kerberos to work properly.

List the keys for the system and check that the host principal is there.

```
[root@server ~]# klist -k
```

3. If necessary, install the **oddjob-mkhomedir** package to allow SSSD to create home directories for Active Directory users.

```
[root@server ~]# yum install oddjob-mkhomedir
```

4. Use **authconfig** to enable SSSD for system authentication. Use the **--enablemkhomedir** to enable SSSD to create home directories.

```
[root@server ~]# authconfig --update --enablesssd --enablesssdauth --enablemkhomedir
```

5. Open the SSSD configuration file.

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

6. Configure the Active Directory domain.

- a. In the **[sssd]** section, add the Active Directory domain to the list of active domains. This is the name of the domain entry that is set in *[domain/NAME]* in the SSSD configuration file.

Also, add **pac** to the list of services; this enables SSSD to set and use MS-PAC information on tickets used to communicate with the Active Directory domain.

```
[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam, pac
```

- b. Create a new domain section at the bottom of the file for the Active Directory domain. This section has the format *domain/NAME*, such as **domain/ad.example.com**. For each provider, set the value to **ad**, and give the connection information for the specific Active Directory instance to connect to.

```
[domain/ad.example.com]
id_provider = ad
ad_server = ipaserver.example.com
ad_hostname = ipa1.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad
```

- c. Make sure that the Active Directory schema is enabled (this is recommended for all deployments and required when using ID mapping), and enable ID mapping.

```
# defines user/group schema type
ldap_schema = ad

# enabling ID mapping
ldap_id_mapping = True
```

- d. Enable credentials caching; this allows users to log into the local system using cached information, even if the Active Directory domain is unavailable.

```
cache_credentials = true
```

- e. Configure access controls.

```
ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true
```

7. Restart the SSH service to load the new PAM configuration.

```
[root@server ~]# systemctl restart sshd.service
```

8. Restart SSSD after changing the configuration file.

```
[root@rhel-server ~]# systemctl restart sssd.service
```

2.5. Configuring an Active Directory Domain with POSIX Attributes

To use Active Directory-defined POSIX attributes in SSSD, those attributes must be replicated to the global catalog. That requires additional configuration on the Active Directory domain. Additionally, ID mapping must be **disabled** in SSSD, so the POSIX attributes are used from Active Directory rather than creating new settings locally.

Other configuration is available in the general LDAP provider configuration ([sssd-ldap](#)) and Active Directory-specific configuration ([sssd-ad](#)). This includes setting LDAP filters for a specific user or group subtree, filters for authentication, and values for some account settings. Some additional configuration is covered in [Section 2.7, "Additional Configuration Examples"](#).

1. Make sure that both the Active Directory and Linux systems have a properly configured environment.
 - ▶ Name resolution must be properly configured, particularly if service discovery is used with SSSD.
 - ▶ The clocks on both systems must be in sync for Kerberos to work properly.
2. In the Active Directory domain, set the POSIX attributes to be replicated to the global catalog.
 - a. Install *Identity Management for UNIX Components* on all primary and child domain controllers. Full details are available in the Microsoft documentation at <http://technet.microsoft.com/en-us/library/cc731178.aspx>.

This allows the POSIX attributes and related schema to be available to user accounts. These attributes are available in the **UNIX Attributes** tab in the entry's **Properties** menu.

- b. Install the Active Directory Schema Snap-in to add attributes to be replicated to the global catalog. This is described in the Microsoft documentation at <http://technet.microsoft.com/en-us/library/cc755885%28v=ws.10%29.aspx>.
- c. The full details for replicating schema are in the Microsoft documentation at <http://support.microsoft.com/kb/248717>.

For the relevant POSIX attributes (*uidNumber*, *gidNumber*, *unixHomeDirectory*, and *loginShell*), open the **Properties** menu, select the **Replicate this attribute to the Global Catalog** checkbox, and then click **OK**.

3. On the Linux client, add the Active Directory domain to the client's DNS configuration so that it can resolve the domain's SRV records.

```
search ipaserver.example.com adserver.example.com
nameserver 198.68.72.1
```

4. Set up the Linux system as an Active Directory client and enroll it within the Active Directory domain. This is done by configuring the Kerberos and Samba services on the Linux system.

- a. Set up Kerberos to use the Active Directory Kerberos realm.

- a. Open the Kerberos client configuration file.

```
[root@server ~]# vim /etc/krb5.conf
```

- b. Configure the **[logging]** and **[libdefaults]** sections so that they connect to the Active Directory realm.

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = yes
```

If autodiscovery is not used with SSSD, then also configure the **[realms]** and **[domain_realm]** sections to explicitly define the Active Directory server.

- b. Configure the Samba server to connect to the Active directory server.

- a. Open the Samba configuration file.

```
[root@server ~]# vim /etc/samba/smb.conf
```

- b. Set the Active Directory domain information in the **[global]** section.

```
[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads
```

- c. Add the Linux machine to the Active Directory domain.

- a. Obtain Kerberos credentials for a Windows administrative user.

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

- b. Add the machine to the domain using the **net** command.


```
[root@server ~]# net ads join -k
Joined 'server' to dns domain 'example.com'
```

This creates a new keytab file, `/etc/krb5.keytab`.

- c. List the keys for the system and check that the host principal is there.

```
[root@server ~]# klist -ke
```

- d. Test that users can search the global catalog, using an `ldapsearch`.

```
[root@server ~]# ldapsearch -H ldap://server.ad.example.com:3268 -Y
GSSAPI -N -b "dc=ad,dc=example,dc=com" "(&(objectClass=user)
(SAMAccountName=aduser))"
```

5. Install the `sssd-ad` package.

```
[root@server ~]# yum install sssd-ad
```

6. Start the SSSD service.

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

7. Open the SSSD configuration file.

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

8. Configure the Active Directory domain.

- a. In the `[sssd]` section, add the Active Directory domain to the list of active domains. This is the name of the domain entry that is set in `[domain/NAME]` in the SSSD configuration file.

Also, add `pac` to the list of services; this enables SSSD to set and use MS-PAC information on tickets used to communicate with the Active Directory domain.

```
[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam, pac
```

- b. Create a new domain section at the bottom of the file for the Active Directory domain. This section has the format `domain/NAME`, such as `domain/ad.example.com`. For each provider, set the value to `ad`, and give the connection information for the specific Active Directory instance to connect to.

```
[domain/ad.example.com]
id_provider = ad
ad_server = ipaserver.example.com
ad_hostname = ipa1.example.com
auth_provider = ad
chpass_provider = ad
access_provider = ad
```

- c. Enable the Active Directory schema. This is recommended for all deployments.

```
# defines user/group schema type
ldap_schema = ad
```

- d. **Disable** ID mapping. This tells SSSD to search the global catalog for POSIX attributes, rather than creating UID:GID numbers based on the Windows SID.

```
# disabling ID mapping
ldap_id_mapping = False
```

- e. If home directory and a login shell are set in the user accounts, then comment out these lines to configure SSSD to use the POSIX attributes rather than creating the attributes based on the template.

```
# Comment out if the users have the shell and home dir set on the AD side
#default_shell = /bin/bash
#fallback_homedir = /home/%d/%u
```

- f. Microsoft Active Directory allows each account to have two Kerberos principals. If the host principal for the domain (such as *client.ad.example.com@AD.EXAMPLE.COM*) is not available, then uncomment the **ldap_sasl_authid** line and set the host principal to use.

```
# Uncomment and adjust if the default principal SHORTNAME$@REALM is not
available
# ldap_sasl_authid = host/client.ad.example.com@AD.EXAMPLE.COM
```

- g. Set whether to use short names or fully-qualified user names for Active Directory users. In complex topologies, using fully-qualified names may be necessary for disambiguation.

```
# Comment out if you prefer to use shortnames.
use_fully_qualified_names = True
```

- h. Enable credentials caching; this allows users to log into the local system using cached information, even if the Active Directory domain is unavailable.

```
cache_credentials = true
```

- i. Configure access controls.

```
ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true
```

9. Set the file permissions and owner for the SSSD configuration file.

```
[root@server ~]# chown root:root /etc/sss/sss.conf
[root@server ~]# chmod 0600 /etc/sss/sss.conf
[root@server ~]# restorecon /etc/sss/sss.conf
```

10. If necessary, install the **oddjob-mkhomedir** package to allow SSSD to create home directories for Active Directory users.

```
[root@server ~]# yum install oddjob-mkhomedir
```

11. Use **authconfig** to enable SSSD for system authentication. Use the **--enablemkhomedir** to enable SSSD to create home directories.

```
[root@server ~]# authconfig --update --enablesssd --enablesssdauth --
enablemkhomedir
```

- Restart the SSH service to load the new PAM configuration.

```
[root@server ~]# systemctl restart sshd.service
```

- Restart SSSD after changing the configuration file.

```
[root@rhel-server ~]# systemctl restart sssd.service
```

Using **authconfig** automatically configured the NSS and PAM configuration files to use SSSD as their identity source.

For example, the **nsswitch.conf** file has SSSD (**sss**) added as a source for user, group, and service information.

```
passwd:          files sss
shadow:         files sss
group:          files sss
...
services:      files sss
...
netgroup:      files sss
```

The different **pam.d** files add a line for the **pam_sss.so** module beneath every **pam_unix.so** line in the **/etc/pam.d/system-auth** and **/etc/pam.d/password-auth** files.

```
auth            sufficient    pam_sss.so use_first_pass
...
account        [default=bad success=ok user_unknown=ignore] pam_sss.so
...
password       sufficient    pam_sss.so use_authtok
...
session        optional      pam_mkhomedir.so
session        optional      pam_sss.so
```

2.6. Configuring Active Directory as an LDAP Domain

While Active Directory can be configured as a type-specific identity provider, it can also be configured as a pure LDAP identity provider with a Kerberos authentication provider.

- It is recommended that SSSD connect to the Active Directory server using SASL, which means that the local host must have a service keytab *for the Windows domain* on the Linux host.

This keytab can be created using Samba.

- Configure the **/etc/krb5.conf** file to use the Active Directory realm.

```
[logging]
default = FILE:/var/log/krb5libs.log

[libdefaults]
default_realm = AD.EXAMPLE.COM
dns_lookup_realm = true
dns_lookup_kdc = true
```

```

ticket_lifetime = 24h
renew_lifetime = 7d
rdns = false
forwardable = yes

[realms]
# Define only if DNS lookups are not working
# AD.EXAMPLE.COM = {
#   kdc = server.ad.example.com
#   admin_server = server.ad.example.com
# }

[domain_realm]
# Define only if DNS lookups are not working
# .ad.example.com = AD.EXAMPLE.COM
# ad.example.com = AD.EXAMPLE.COM

```

- b. Set the Samba configuration file, `/etc/samba/smb.conf`, to point to the Windows Kerberos realm.

```

[global]
workgroup = EXAMPLE
client signing = yes
client use spnego = yes
kerberos method = secrets and keytab
log file = /var/log/samba/%m.log
password server = AD.EXAMPLE.COM
realm = EXAMPLE.COM
security = ads

```

- c. Then, run the `net ads` command to log in as an administrator principal. This administrator account must have sufficient rights to add a machine to the Windows domain, but it does not require domain administrator privileges.

```
[root@server ~]# net ads join -U Administrator
```

- d. Run `net ads` again to add the host machine to the domain. This can be done with the host principal (`host/FQDN`) or, optionally, with the NFS service (`nfs/FQDN`).

```
[root@server ~]# net ads join createupn="host/rhel-
server.example.com@AD.EXAMPLE.COM" -U Administrator
```

2. Make sure that the Services for Unix package is installed on the Windows server.
3. Set up the Windows domain which will be used with SSSD.
 - a. On the Windows machine, open **Server Manager**.
 - b. Create the Active Directory Domain Services role.
 - c. Create a new domain, such as **ad.example.com**.
 - d. Add the Identity Management for UNIX service to the Active Directory Domain Services role. Use the Unix NIS domain as the domain name in the configuration.
4. On the Active Directory server, create a group for the Linux users.
 - a. Open **Administrative Tools** and select **Active Directory Users and Computers**.

- b. Select the Active Directory domain, **ad.example.com**.
 - c. In the **Users** tab, right-click and select **Create a New Group**.
 - d. Name the new group **unixusers**, and save.
 - e. Double-click the **unixusers** group entry, and open the **Users** tab.
 - f. Open the **Unix Attributes** tab.
 - g. Set the NIS domain to the NIS domain that was configured for **ad.example.com** and, optionally, set a group ID (GID) number.
5. Configure a user to be part of the Unix group.
- a. Open **Administrative Tools** and select **Active Directory Users and Computers**.
 - b. Select the Active Directory domain, **ad.example.com**.
 - c. In the **Users** tab, right-click and select **Create a New User**.
 - d. Name the new user **aduser**, and make sure that the **User must change password at next logon** and **Lock account** checkboxes are *not* selected.
- Then save the user.
- e. Double-click the **aduser** user entry, and open the **Unix Attributes** tab. Make sure that the Unix configuration matches that of the Active Directory domain and the **unixgroup** group:
 - ▶ The NIS domain, as created for the Active Directory domain
 - ▶ The UID
 - ▶ The login shell, to **/bin/bash**
 - ▶ The home directory, to **/home/aduser**
 - ▶ The primary group name, to **unixusers**



TIP

Password lookups on large directories can take several seconds per request. The initial user lookup is a call to the LDAP server. Unindexed searches are much more resource-intensive, and therefore take longer, than indexed searches because the server checks every entry in the directory for a match. To speed up user lookups, index the attributes that are searched for by SSSD:

- ▶ uid
- ▶ uidNumber
- ▶ gidNumber
- ▶ gecos

6. On the Linux system, configure the SSSD domain.

```
[root@rhel-server ~]# vim /etc/sss/sss.conf
```

For a complete list of LDAP provider parameters, see the **sssd-ldap(5)** man pages.

Example 2.2. An Active Directory 2008 R2 Domain with Services for Unix

```
[sssd]
config_file_version = 2
domains = ad.example.com
services = nss, pam

...

[domain/ad.example.com]
cache_credentials = true

# for performance
ldap_referrals = false

id_provider = ldap
auth_provider = krb5
chpass_provider = krb5
access_provider = ldap

ldap_schema = rfc2307bis

ldap_sasl_mech = GSSAPI
ldap_sasl_authid = host/rhel-server.example.com@AD.EXAMPLE.COM

#provide the schema for services for unix
ldap_schema = rfc2307bis

ldap_user_search_base = ou=user accounts,dc=ad,dc=example,dc=com
ldap_user_object_class = user
ldap_user_home_directory = unixHomeDirectory
ldap_user_principal = userPrincipalName

# optional - set schema mapping
# parameters are listed in sssd-ldap
ldap_user_object_class = user
ldap_user_name = sAMAccountName

ldap_group_search_base = ou=groups,dc=ad,dc=example,dc=com
ldap_group_object_class = group

ldap_access_order = expire
ldap_account_expire_policy = ad
ldap_force_upper_case_realm = true
ldap_disable_referrals = true

krb5_realm = AD-REALM.EXAMPLE.COM
# required
krb5_canonicalize = false
```

7. Restart SSSD.

```
[root@rhel-server ~]# systemctl restart sssd.service
```

2.7. Additional Configuration Examples

2.7.1. Account Settings

With Linux users, certain system preferences are set by default for new users. For example, the `pam_oddjob_mkhomedir.so` library automatically creates home directories in a defined location.

These system preferences either may not be set in the Windows user accounts or may be set to something incompatible with a Linux system. There are two such areas:

- ▶ The user home directory
- ▶ A default user shell

2.7.1.1. Setting a User Home Directory

Red Hat Enterprise Linux has a PAM library (`pam_oddjob_mkhomedir.so`) which automatically creates user directories when a user first logs in. This includes Active Directory users, when they first log into a Linux system.

With SSSD, the format of the user directory is retrieved from the identity provider. If the identity provider has a home directory format that is different than the format for the Linux system or if it does not supply a value, then SSSD can be configured to create a home directory using a template specified in its configuration. The template can be set globally in the NSS service section or per domain.

There are two possible parameters:

- ▶ `fallback_homedir`, which supplies a template if the identity provider does not supply one
- ▶ `override_homedir`, which sets a template to use regardless of what information is set in the identity provider

Both can use variables within the template, such a `%u` for the login name and `%d` for the domain name.

```
[nss]
fallback_homedir = /home/%u

...

[domain/AEXAMPLE]
id_provider = ad
ad_server = ipaserver.example.com
ad_hostname = ipa1.example.com
auth_provider = ad
...
override_homedir = /home/%d/%u
```

2.7.1.2. Setting a User Shell

By default, SSSD attempts to retrieve information about user shells from the identity provider. In both Active Directory and LDAPv3 schema, this is defined in the `loginShell` attribute. However, this is an optional attribute, so it may not be defined for every user. For Active Directory users, the defined login shell may not be allowed on the Linux system.

There are a number of ways to handle shells in the SSSD configuration:

- ▶ Setting a fallback value if no shells are supplied (`shell_fallback`)
- ▶ Setting lists of allowed or blacklisted shells (`allowed_shells` and `vetoed_shells`)
- ▶ Setting a default value (`default_shell`)

- ▶ Setting a value to use, even if another value is given in the identity provider (**override_shell**)



NOTE

allowed_shells, **vetoed_shells**, and **shell_fallback** can only be set as global settings, not per domain. However, these parameters do not affect local system users, only external users retrieved through SSSD identity providers. Using a general setting, such as **/bin/rbash**, is good for most external users.

Default values can be set per domain, while some values (such as the white and blacklists for shells) must be set globally, in the NSS service configuration. For example:

```
[nss]
shell_fallback = /bin/sh
allowed_shells = /bin/sh,/bin/rbash,/bin/bash
vetoed_shells = /bin/ksh
...

[domain/AEXAMPLE]
id_provider = ad
ad_server = ipaserver.example.com
ad_hostname = ipa1.example.com
auth_provider = ad
...
default_shell = /bin/rbash
```

2.7.2. Enabling Dynamic DNS Updates (Active Directory Only)

Active Directory allows its clients to refresh their DNS records automatically. Active Directory also actively maintains DNS records to make sure they are updated, including timing out (aging) and removing (scavenging) inactive records.

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, then the client's DNS record is refreshed at several times:

- ▶ When the identity provider comes online (always)
- ▶ When the Linux system reboots (always)
- ▶ At a specified interval (optional configuration)



TIP

This can be set to the same interval as the DHCP lease, which means that the Linux client is renewed after the lease is renewed.

DNS updates are sent to the Active Directory 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
ad_server = ipaserver.example.com
ad_hostname = ipa1.example.com
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
```

Table 2.1. Options for Dynamic DNS Updates

Option	Description	Format
dyndns_update	Sets whether to update the DNS server dynamically with the client IP address. This requires secure updates. This must be set to true for any other dynamic DNS setting to be enabled. The default is true.	Boolean
dyndns_ttl	Sets a time-to-live for the client's DNS record. The default is 3600 seconds.	Integer
dyndns_refresh_interval	Sets a frequency to perform an automatic DNS update, in addition to the update when the provider comes online. The default is 86400 seconds (24 hours).	Integer
dyndns_update_ptr	Sets whether to update the PTR record when the client updates its DNS records. The default is true.	Boolean

2.7.3. Using a Filter with Access Controls

There is an Active Directory access provider, which means that Active Directory is used as the source for authorization information. This is actually a shortcut, that combines several generic LDAP parameters into a single configuration parameter. Setting the Active Directory provider:

```
access_provider = ad
```

This is the same as setting several different LDAP parameters, including setting the access order to check for account expirations.

```
access_provider = ldap
ldap_access_order = expire
ldap_account_expire_policy = ad
```

There is an additional option to identify which user accounts to grant access to based on an LDAP filter. First, accounts must match the filter, and then they must pass the expiration check (implicit in the **access_provider = ad** setting).

For example, this sets that only users which belong to the administrators group and have a **unixHomeDirectory** attribute match the access control check:

```
access_provider = ad
ad_access_filter = (&(memberOf=cn=admin,ou=groups,dc=example,dc=com)
(unixHomeDirectory=*))
```

The access control check requires a secure connection (SASL with a GSS-API mechanism). Configuring the same functionality using the generic LDAP parameters requires defining that SASL/GSS-API connection, the filter, and the expiration checks.

```
access_provider = ldap
ldap_access_order = filter, expire
ldap_account_expire_policy = ad
ldap_access_filter = (&(memberOf=cn=admin,ou=groups,dc=example,dc=com)
(unixHomeDirectory=*))
ldap_sasl_mech = GSSAPI
ldap_sasl_authid = CLIENT_SHORTNAME$@EXAMPLE.COM
ldap_schema = ad
```

Chapter 3. Using realmd to Connect to an Active Directory Domain

realmd provides a clear and simple way to discover and join identity domains. It does not connect to the domain itself; rather, it configures underlying Linux system services (SSSD or Winbind) to connect to the domain.

3.1. About realmd

[Chapter 2, Using Active Directory as an Identity Provider for SSSD](#) describes using the System Security Daemon on a local system and Active Directory as a backend identity provider. There are a number of different configuration parameters, some required and some optional, for each possible identity provider and for SSSD itself.

All of the domain information must be available in advance and then properly formatted in the SSSD configuration for SSSD to integrate the local system with Active Directory. That can be complex.

realmd simplifies that configuration. **realmd** can run a service discovery to identify different, available domains (both Active Directory and Red Hat Enterprise Linux Identity Management), and then join the domain and manage user access.

realmd can discover and support multiple domains because the underlying service (SSSD) supports multiple domains.

3.2. realmd Commands

The **realmd** command has two major task areas: managing the system's enrollment in a domain and setting which domain users are allowed to access the local system resources.

The **realmd** command mostly specifies an action and the realm for which to perform that action.

```
realm command realmName
```

For example:

```
realm join ad.example.com
```

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	Removes the system from the given domain.
list	Lists all configured realms for the system or (optionally) all configured and discovered realms.
Login Commands	
permit	Permits access for specified users or for all users within a configured realm to access the local system.

Command	Description
deny	Restricts access for specified users or for all users within a configured realm to access the local system.

3.3. Discovering and Joining Active Directory Domains

3.3.1. Discovering Domains

All that the discovery process requires is the **discover** command. This returns all of the realm configuration and a list of packages that must be installed for the system to be enrolled in the realm.

```
[root@server ~]# realm discover
ad.example.com
  type: active-directory
  realm-name: AD.EXAMPLE.COM
  domain-name: ad.example.com
  configured: kerberos-member
  server-software: active-directory
  client-software: sssd
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  login-formats: %D\%U
  login-policy: allow-realm-logins
```

realmd checks for the DNS SRV record:

```
_ldap._tcp.dc._msdcs.domain.example.com.
```

That is created by default when Active Directory is configured, which enables it to be found by the service discovery. **realmd** uses the domain assigned through DHCP to discover any LDAP servers on the network.

realmd can discover both Active Directory and Identity Management domains. If both are in the environment, then it is possible to limit the discovery results to a specific type of server:

```
[root@server ~]# realm discover --server-software=active-directory
```

It is also possible to run a discovery for a specific domain, using the domain controller's hostname or IP address. For example:

```
[root@server ~]# realm discover ad.example.com
```

3.3.2. Joining an Active Directory Domain

The **join** command only requires the realm name:

```
[root@server ~]# realm join ad.example.com
See: journalctl REALMD_OPERATION=r1088239.6316
realm: Joined ad.example.com domain
```

This performs the join as the default Windows administrator and, in most environments, will prompt for the password. The command can connect to the Active Directory environment as a different user, by using the

-U option:

```
[root@server ~]# realm join ad.example.com -U AD.EXAMPLE.COM\jsmith
```

If Kerberos is properly configured on the Linux system, then it can also be performed using a Kerberos ticket to authenticate. The **realmd** command can use the **-U** option then to select which principal to use or can use the default credential cache or **KRB5_CCACHE** variable.

```
[root@server ~]# kinit jsmith
[root@server ~]# realm join ad.example.com -U jsmith
```

The actual join process configures both the local system services and the entries in the Active Directory domain.

1. Runs a discovery scan for the specified realm.
2. Installs any required packages to join the system to the domain. This includes SSSD and the PAM home directory job packages.
3. Attempts to join the Active Directory domain as Administrator (unless a different user is specified with the **-U** option). The command first attempt to connect without credentials, but it prompts for a password if required.
4. Once it connects to the domain, it creates an account entry for the system in the directory.
5. It creates a host keytab (**/etc/krb5.keytab**).
6. It configures the domain in SSSD and restarts the service.
7. Enables domain users for the system services (**/etc/nsswitch.conf**).

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, then they can be manually allowed using the **permit** command. This is covered in [Section 3.4, “Managing User Logins from Active Directory”](#).

3.3.3. Removing a System from the Active Directory Domain

If a system should ever be removed from an Active Directory domain, this is done with the **leave** command. This removes the domain configuration from SSSD and NSS on the local system.

```
[root@server ~]# realm leave ad.example.com
```

This attempts to perform the removal as the default Administrator in Active Directory. The script may prompt for a password or, depending on how the system was joined to the domain, it may require performing the operation as a different user. This can be specified with the **-U** option.

```
[root@server ~]# realm leave ad.example.com -U AD.EXAMPLE.COM\jsmith
```

3.3.4. Listing Domains

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

```
[root@server ~]# realm list
linux.example.com
  type: kerberos
  realm-name: LINUX.EXAMPLE.COM
  domain-name: linux.example.com
  configured: kerberos-member
  server-software: ipa
  client-software: sssd
  required-package: ipa-client
  required-package: oddjob
  required-package: oddjob-mkhomedir
  required-package: sssd
  login-formats: %U
  login-policy: allow-realm-logins
```

The **--all** option includes discovered domains (Active Directory, Identity Management, and Kerberos) as well as configured domains. The **--name-only** limits the results to the domain name, without the configuration details.

```
[root@server ~]# realm list --all
linux.example.com
example.com
ad.example.com
```

3.4. Managing User Logins from Active Directory

By default, login policies for domain users are defined in the domain itself. (This can be overridden in the realm configuration so that the local policies only define who is allowed to login.) *In addition to* those domain policies, **realmd** can configure basic allow/deny access rules for users from a specific domain.



NOTE

These access rules either allow all access to the system or no access. Finer-grained access rules must be set on a specific system resource or in the domain.

There are two commands that set access rules: **permit** and **deny**. **Permit** is the more flexible of the two commands. The **deny** command is a blanket command that prohibits access from any user within the realm. The **permit** command, on the other hand, can grant access to all users (**--all**) or to only specified users, and it can also withdraw permission from specified users (**-x**).

For example, this adds an allow rule for every user within the **ad.example.com** domain, and then withdraws login permission from the **jsmith** user.

```
[root@server ~]# realm permit ad.example.com --all
[root@server ~]# realm permit ad.example.com --x AD.EXAMPLE.COM\jsmith
```

3.5. Adding Default User Configuration

The **/etc/realmd.conf** configuration file can add custom configuration for global logged-in user settings. Some POSIX attributes not be set in the Windows user accounts or may be set to something different than other users on the local system. There are two such areas:

- ▶ The user home directory
- ▶ A default user shell

User settings are defined in the `[users]` section in the `/etc/realmd.conf` file.

- ▶ The `default-shell` parameter can be any supported system shell.
- ▶ The `default-home` parameter sets a template to use to create a home directory if none is defined in the realm. A common format is `/home/%d/%u`, where `%d` is the domain name and `%u` is the user name.

For example:

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

3.6. Additional Configuration for the Active Directory Domain Entry

Each individual domain can have custom settings in the realm entry in the `/etc/realmd.conf` configuration file. Each realm can have a configuration section:

```
[realm.name]
attribute = value
attribute = value
```

Each attribute can be set by manually adding them to the configuration file or by passing them as arguments when the system is joined to the realm.

Table 3.2. Realm Configuration Options

Parameter	Description
computer-ou	Sets the directory location to use to add 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 whether to create a host principal for the system.
automatic-id-mapping	Sets whether to enable dynamic ID mapping or whether to disable mapping and use POSIX attributes configured in Active Directory.
manage-system	Sets whether certain login policies are set in the local system or by Active Directory.

For example, this disables ID mapping, enables the host principal, and adds the system to the specified subtree.

```
[domain.example.com]
computer-ou = OU=Linux Computers,DC=domain,DC=example,DC=com
user-principal = yes
automatic-id-mapping = no
manage-system = no
```

These same parameters can be passed when the system is joined to the domain:

```
[root@server ~]# realm join --computer-ou="ou=Linux Computers," --automatic-id-  
mapping=no --user-principal=yes
```


Chapter 4. Using Samba, Kerberos, and Winbind

Samba allows Linux systems to join an Active Directory environment by making them appear to be Windows clients. Samba is an intermediary. As a means of systems integration, this allows a Linux client to join an Active Directory Kerberos realm and to use Active Directory as its identity store.

Once a central Samba server is configured, then clients can be configured to connect to that Samba server, and from there to Active Directory. This is done using the local *Winbind* client. Winbind itself is a local authentication tool, but within a larger Samba domain, it helps maintain and resolve user identities to help maintain file ownership and access.

4.1. About Samba and Active Directory Authentication

Samba was one of the first points for Windows integration, starting with a way to share files and data between Linux and Windows environments. The Samba group has an excellent reference for [understanding and configuring Samba](#).

Samba has the slogan *Opening Windows to a wider world*, and it has a very broad range of features and deployment options. Its core functionality is filesharing and printing between Windows and Linux environments — and that is outside the scope of this guide.

This chapter looks at one aspect of using Samba to open up Windows: allowing Linux clients to authenticate using Active Directory. This means adding Samba as a domain member of the Windows domain.

4.1.1. About Samba, Kerberos, and Active Directory Domains

Active Directory is the domain controller for a number of services for the Windows environment, including Kerberos realms and DNS domains. Samba has several different means of integrating with a Windows domain, both NTLM and Active Directory. The different options track with differences inherent in the Windows servers themselves. For Active Directory integration (which is the scope of this document), integrating with Active Directory means configuring a security environment that uses Kerberos the native security context in Active Directory.

Several different system services should be configured on the Linux client to use Active Directory as a domain controller:

- ▶ Samba (for users and authentication)
- ▶ DNS (to set the Active Directory server as the nameserver)
- ▶ Kerberos (to use the Active Directory KDC)
- ▶ PAM (to use Winbind)
- ▶ NSS (to use Winbind and Kerberos)

4.1.1.1. Samba

There are several different ways that a Samba server can join an Active Directory domain. The central decision is how the users can be associated with the domain. In Samba, the user/domain structure is called a *security mode*. There are four user security modes:

- ▶ *user*, the Samba default setting, requires simple usernames and plaintext passwords. The username format must be *domain\user*, such as **EXAMPLE\jsmith**. [2]

This is a local-level security mode, and is not used for Active Directory integration.

- ▶ *server* configures pass-through authentication to another Samba server.
- ▶ *domain* configures the local Samba server as a domain member within a Windows NT4 domain.
- ▶ *ads* configures the local Samba server as a domain member within an Active Directory domain.

Of those security modes, only **domain** or **ads** are used for Windows integration, and **ads** allows Kerberos security, which is native to Active Directory.

The critical settings are the security type (**security**); the name of the Active Directory Kerberos realm (**realm**), which is resolved by DNS discovery; the Samba workgroup (**workgroup**); and enabling encrypted passwords. There are other settings for Winbind, such as ID ranges for mapping UID and GID numbers to users and templates for group membership and home directories; these can either be set or left to default values.

This is all configured in the **[global]** section of `/etc/samba/smb.conf`.

```
#===== Global Settings =====
[global]
    workgroup = ADEXAMPLE
    security = ads
    realm = ADEXAMPLE.COM
    encrypt passwords = yes
    ... 8< ...
```

4.1.1.2. Kerberos

Kerberos must be configured to use the Active Directory server as its KDC. This allows users to use Kerberos tickets for authentication. Additionally, Samba must be configured to use the Active Directory Kerberos realm; this allows Winbind to manage the Kerberos principals.

The Active Directory realm should be set as the default domain in the **[libdefaults]** section of the `/etc/krb5.conf` file, and then as a KDC in the **[realms]** section. The **[domain_realm]** section should define the Active Directory domain.

```
[libdefaults]
... 8< ...
    default_realm = ADEXAMPLE.COM

[realms]
    ADEXAMPLE.COM = {
        kdc = kdc.adexample.com
    }

[domain_realm]
    adexample.com = ADEXAMPLE.COM
    .adexample.com = ADEXAMPLE.COM
```

4.1.1.3. DNS

The local DNS service must be configured to use Active Directory as its domain controller. DNS is critical for proper resolution of hostnames and domains for Kerberos. While many systems have proper DNS settings, so that Samba-Active Directory integration could work well without configuring Active Directory as a nameserver, using Active Directory as a nameserver avoids any potential resolution problems. The domain should also be added as a **search** directive, so that the Active Directory domain is used for searches and discovery.

This is configured in the `/etc/resolv.conf` file.

```
nameserver 1.2.3.4
search adexample.com
```

4.1.1.4. PAM and NSS

PAM (and NSS) allow local applications to use the Kerberos credentials provided by Active Directory, which enables single sign-on for system applications and domain users. It is possible to use Kerberos directly or to use Winbind, but for ease of use, offline caching of credentials, and other features, it is recommended that you use Winbind.

For PAM, the Winbind libraries are set for authentication, account, password, and (optionally) session management. This is configured in the `/etc/pam.d/system-auth` file.

```
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth sufficient pam_winbind.so use_first_pass
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account [default=bad success=ok user_unknown=ignore] pam_winbind.so
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password sufficient pam_winbind.so use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet
use_uid
session     required      pam_unix.so
session     optional      pam_krb5.so
session optional pam_winbind.so use_first_pass
```

For NSS, Active Directory can be used for passwords, shadow (users), and groups by setting Winbind as an option. Additionally, this can be used for hosts by setting the WINS service option. Always leave **files** as the first location to check for accounts; this allows local system users and services to be able to log in and access resources.

This is configured in the `/etc/nsswitch.conf` file.

```
passwd:    files winbind
shadow:    files winbind
group:     files winbind

hosts:     files dns wins
```

4.1.2. About Winbind v3 and Samba v3, and Authentication

There are two important tasks when dealing with files: establishing the proper ownership and controlling access to appropriate parties. Both of these ultimately relate to an effective way to identify and authenticate users, and that was supplied by Winbind.

Winbind does three related but separate things:

- ▶ Authenticate users (using local PAM configuration)
- ▶ Resolve IDs, usernames, and groups (using NSS lookups)
- ▶ Create a database of mapped Active Directory SIDs and local UID/GID numbers

Winbind as a client interacts with *Samba*, and then Samba actually connects to the Active Directory domain. The local system is not enrolled in the Windows domain in any meaningful way; it simply uses the Active Directory environment (through Samba) as an identity store. PAM and NSS are configured to use Winbind for user identities on the local system.

Winbind is a part of Samba, and that structure has some inherent limits for more general Windows integration:

- ▶ Samba is configured only for filesharing, so the core design for authentication and identity management is designed to manage file ownership and authenticate users. Users in Active Directory (and even applications on the Linux resources) are limited to certain PAM modules and NSS configuration for trying to leverage authentication or implement other security policies.
- ▶ Winbind is only used for authentication, so only the username and password attributes in the Active Directory entry are ever used or available for the Windows user.
- ▶ SIDs are mapped to a locally-chosen UID/GID pair. This ignores any POSIX settings within the Active Directory entry.
- ▶ Winbind's behavior is bound to the Windows domain. Things like Kerberos ticket management, and even joining the domain, are performed through Windows tools like **net join ads**.
- ▶ ID ranges, ID mapping, and group management are all configured manually and in different configuration files for Samba and PAM.

4.2. Summary of Configuration Files, Options, and Packages

Table 4.1. System Configuration Files, Required Options, and Required Packages

Service	Configuration File	Required Parameters	Required Packages
Samba	/etc/samba/smb.conf	<pre>[global] workgroup = ADEXAMPLE security = ads realm = ADEXAMPLE.COM encrypt passwords = yes ... 8< ...</pre>	samba
Winbind			samba-winbind

Service	Configuration File	Required Parameters	Required Packages
Kerberos	/etc/krb5.conf	<pre>[libdefaults] ... 8< ... default_realm = AEXAMPLE.COM [realms] AEXAMPLE.COM = { kdc = kdc.adexample.co m } [domain_realm] adexample.com = AEXAMPLE.COM .adexample.com = AEXAMPLE.COM</pre>	krb5-workstation
PAM	/etc/pam.d/system-auth or /etc/pam.d/system-auth-ac (with authconfig)	<pre>auth sufficient pam_winbind.so use_first_pass account [default=bad success=ok user_unknown=igno re] pam_winbind.so password sufficient pam_winbind.so use_authtok session optional pam_winbind.so use_first_pass</pre>	
NSS	/etc/nsswitch.conf	<pre>#required passwd: files winbind shadow: files winbind group: files winbind #optional hosts: files dns wins</pre>	

Service	Configuration File	Required Parameters	Required Packages
DNS	/etc/resolv.conf	<pre>nameserver IPaddress search domainName</pre>	

4.3. Configuring a Domain Member Using authconfig


All of the configuration outlined in [Section 4.2, “Summary of Configuration Files, Options, and Packages”](#) can be done automatically using the **authconfig** tool, with the exception of the DNS configuration. This also provides the ability to back up the configuration files, which allows any changes to be reverted later (which can be especially convenient if there is a mistake or a change to the configuration).

4.3.1. authconfig Arguments and Configuration File Parameters

The Authentication Configuration tool automatically updates the required configuration files for Samba, Kerberos, and Active Directory integration when it is used to configure Winbind as the authentication store for the local system. The relationships between the **authconfig** options and the required configuration changes are not always obvious. [Table 4.2, “authconfig Arguments and Configuration File Parameters”](#) shows what parameters are set with each command option; this can help ensure that the correct values are used and that optional settings are refined.

Table 4.2. authconfig Arguments and Configuration File Parameters

Service	CLI Option	GUI Field	Configuration File	Configuration Parameter
Samba	--smbsecurity	Security Model	/etc/samba/smb.conf	security
Samba	--smbworkgroup	Winbind Domain	/etc/samba/smb.conf	workgroup
<ul style="list-style-type: none"> ▶ Samba ▶ Kerberos 	--smbrealm	Winbind ADS Realm	<ul style="list-style-type: none"> ▶ Samba <ul style="list-style-type: none"> ▪ /etc/samba/smb.conf ▶ Kerberos <ul style="list-style-type: none"> ▪ /etc/krb5.conf 	<ul style="list-style-type: none"> ▶ Samba <ul style="list-style-type: none"> ▪ realm in [global] ▶ Kerberos <ul style="list-style-type: none"> ▪ default_realm in [libdefaults] ▪ realm entry (REALMNAME {...}) in [realms]
Kerberos	--smbservers	Winbind Domain Controllers	/etc/krb5.conf	The KDC in the realm entry (e.g., REALMNAME {...}) in [realms]

Service	CLI Option	GUI Field	Configuration File	Configuration Parameter
Kerberos	--krb5realm		/etc/krb5.conf	The domain entry in [domain_realm]
	<div style="border: 1px solid black; padding: 5px; background-color: #f0f0f0;"> <div style="background-color: #d4af37; color: white; padding: 2px; display: inline-block; border: 1px solid white;">  IMPORTANT </div> <p style="margin-top: 5px;">This value must be identical to the value given in --smbrealm for the domain to be configured properly.</p> </div>			
PAM	--enablewinbindauth		/etc/pam.d/system-auth	auth, account, password, sessions
NSS	--enablewinbind		/etc/nsswitch.conf	passwd, shadow, group
NSS	--enablewins		/etc/nsswitch.conf	hosts
Winbind	--enablecache			
Winbind	--enablewinbindkrb5			
Winbind	--enablewinbindoffline			

4.3.2. Configuring Active Directory Authentication with authconfig in the Command Line

1. Install the **samba-winbind** package. This is required for Windows integration features in Samba services, but is not installed by default.

```
[root@server ~]# yum install samba-winbind
```

2. Install the **krb5-workstation** package. This is required to connect to a Kerberos realm and manage principals and tickets.

```
[root@server ~]# yum install krb5-workstation
```

3. Edit the DNS configuration to use the Active Directory domain as a nameserver and for search.

```
[root@server ~]# vim /etc/resolv.conf
nameserver 1.2.3.4
search adexample.com
```

4. The **authconfig** command does not set any requirements for what options must be invoked at a given time, since it can be used to modify configuration as well as to define new configuration.

This example provides all of the required Samba, Kerberos, PAM, and NSS parameters, as well as options for Winbind which allow offline access and for the local system which allow system accounts to continue to work. This command should be all on one line; it has been broken into multiple lines and annotated to show what options are being used.

```
[root@server ~]# authconfig
// NSS
--enablewinbind
--enablewins
// PAM
--enablewinbindauth
// Samba
--smbsecurity ads
--smbworkgroup=AEXAMPLE
--smbrealm AEXAMPLE.COM
// Kerberos
--smbservers=ad.example.com
--krb5realm=AEXAMPLE.COM
// winbind
--enablewinbindoffline
--enablewinbindkrb5
--winbindtemplateshell=/bin/sh
// general
--winbindjoin=admin
--update
--enablelocalauthorize
--savebackup=/backups
[/usr/bin/net join -w AEXAMPLE -S ad.example.com -U admin]
```

The **--winbindjoin** parameter automatically runs the **net join** command to add the system to the Active Directory domain.

The **--enablelocalauthorize** sets local authorization operations to check the **/etc/passwd** file. This allows local accounts to be used to authenticate users as well as the Active Directory domain.



TIP

The **--savebackup** option is not required, but it is recommended. This backs up the configuration files to the given directory before making the changes. If there is a configuration error or of the configuration later is changed, **authconfig** can use the backups to revert the changes.

4.3.3. Configuring Active Directory Authentication in the **authconfig** GUI

There are fewer configuration options in the **authconfig** UI than are in the CLI. For example, it is possible to configure Samba, NSS, and Winbind and to join the domain, but it does not configure Kerberos or PAM. Those must be configured manually if using the UI.

**NOTE**

The **authconfig** command-line tools are installed by default, but the GUI requires the **authconfig-gtk** package, which is not available by default.

1. Install the **samba-winbind** package. This is required for Windows integration features in Samba services, but is not installed by default.

```
[root@server ~]# yum install samba-winbind
```

2. Install the **krb5-workstation** package. This is required to connect to a Kerberos realm and manage principals and tickets.

```
[root@server ~]# yum install krb5-workstation
```

3. Configure the Active Directory Kerberos realm as the default realm and KDC for the local system.

```
[root@server ~]# vim /etc/krb5.conf

[libdefaults]
... 8< ...
    default_realm = ADEXAMPLE.COM

[realms]
    ADEXAMPLE.COM = {
        kdc = kdc.adexample.com
    }

[domain_realm]
    adexample.com = ADEXAMPLE.COM
    .adexample.com = ADEXAMPLE.COM
```

4. Edit the DNS configuration to use the Active Directory domain as a nameserver and for search.

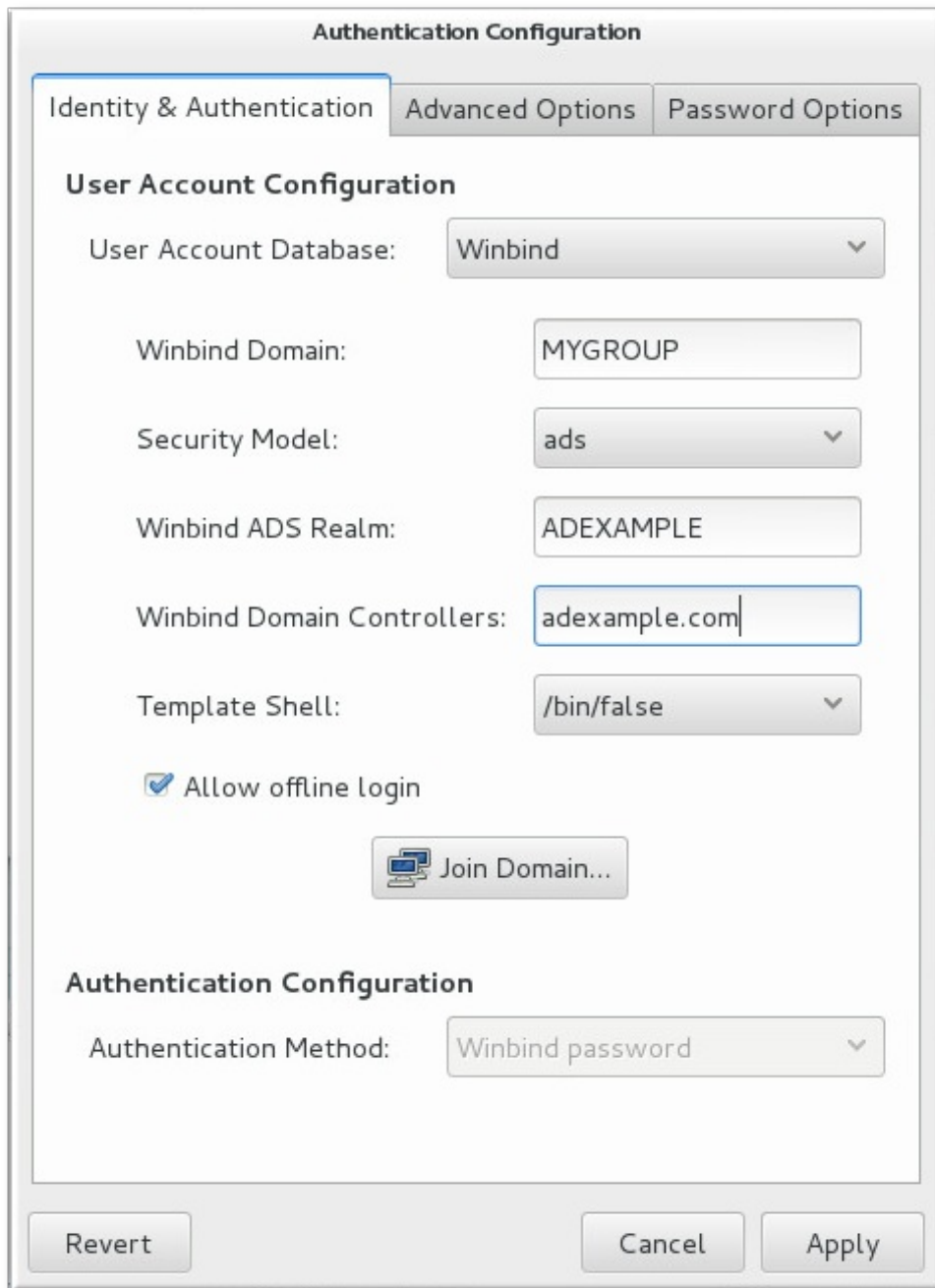
```
[root@server ~]# vim /etc/resolv.conf

nameserver 1.2.3.4
search adexample.com
```

5. Open the Authentication Configuration Tool.

```
[root2server ~]# authconfig-gtk
```

6. In the **Identity & Authentication** tab, select **Winbind** in the **User Account Database** drop-down menu.



7. Set the information that is required to connect to the Microsoft Active Directory domain controller.
 - » **Winbind Domain** gives the Windows work group.
This should be in the Windows 2000 format, such as **DOMAIN**.
 - » **Security Model** sets the security model to use for Samba clients. This is **ads**, to configure Samba to act as a domain member in an Active Directory Server realm.
 - » **Winbind ADS Realm** gives the Active Directory realm that the Samba server will join.
 - » **Winbind Domain Controllers** gives the hostname or IP address of the domain controller to use.
 - » **Template Shell** sets which login shell to use for Windows user account settings. This is optional.
 - » **Allow offline login** allows authentication information to be stored in a local cache. The cache is referenced when a user attempts to authenticate to system resources while the system

is offline.

8. Click the **Join Domain** to run the **net ads join** command to join the Active Directory domain. This is only to join the domain immediately; the configuration can be saved and then the **net ads join** command can be run manually later.
9. Click the **Apply** button to save the configuration.

[2] When verifying that a given user exists in the Windows domain, always use Windows 2000-style formats and escape the backslash (\) character. For example:

```
[root@server ~]# getent passwd domain\user DOMAIN\user:*:16777216:16777216:Name
Surname:/home/DOMAIN/user:/bin/bash
```

This can also be specified in the `authconfig` command-line tool with the `--winbindseparator=` option.

Part II. Integrating a Linux Domain with an Active Directory Domain

Chapter 5. Creating Cross-Realm Trusts with Active Directory and Identity Management

Kerberos allows the configuration of *trusted realms*. Each realm has its own resources and users, yet the trust relationship allows users of any trusted realm to obtain tickets and connect to machines or services in a peer realm as if they were members of that peer realm.

Because of differences in the way that Windows and Linux domains implement LDAP services, DNS management, and even Kerberos realms, it is difficult to establish a direct trust between Active Directory and Linux domains manually. A trust relationship using IdM centrally defines and establishes the Kerberos trust and DNS mappings so that Active Directory users can access Linux hosts and services completely transparently, using one set of credentials.

5.1. The Meaning of "Trust"

Kerberos has the ability to create a relationship between two otherwise separate realms. This is called a *cross-realm trust*. These realms create a shared ticket and key so a member of one realm is perceived as a member of both realms. One realm *trusts* another.

A cross-realm Kerberos trust is limited to Kerberos realms. However, identity management environments such as Active Directory and IdM in Red Hat Enterprise Linux include services other than Kerberos (most notably DNS) within their domain definitions. It is possible to establish a trusted relationship across the full domain, not just the Kerberos realm, through IdM trusts.

5.1.1. The Architecture of a Trust Relationship

Both Active Directory and Identity Management manage a variety of core services: Kerberos, LDAP, DNS, certificate services. For these two disparate domains to be integrated transparently, all of these core services need to be able to interact cleanly with one another.

Those services can be broken into two major points of interaction: a Kerberos realm and a DNS domain. Certificate services, LDAP entries, and other services can be managed independently for Active Directory and IdM. The place where they intersect is where identities need to be authenticated (Kerberos) and a mechanism to route queries between domains (DNS).

A trust establishes an identity/access relationship between one domain and another domain. Active Directory environments can be complex, so there are different possible types and arrangements for Active Directory trusts, between subdomains, root domains, forests, and external domains. Microsoft TechNet provides an excellent primer, [How Domain and Forest Trusts Work](#), on the internal components used to create trust relationships and different types of trusts.

A trust is a path from one domain or realm to another. The way that identities and information moved between the domains is the *trust flow*.

At its most basic, a trust flows one direction. One domain contains users (the *trusted domain*), and one domain contains resources (the *trusting domain*). The domain with resources *trusts* the domain with the users. This means that the users can access the trusting domain resources — but it does not work the other way. The users in the *trusting* domain cannot access resources in the *trusted* domain.

This is illustrated in [Figure 5.1, "Basic Unidirectional Trusts"](#). Realm A is trusted by Realm B, but Realm B is not trusted by Realm A. The trust is *unidirectional*.

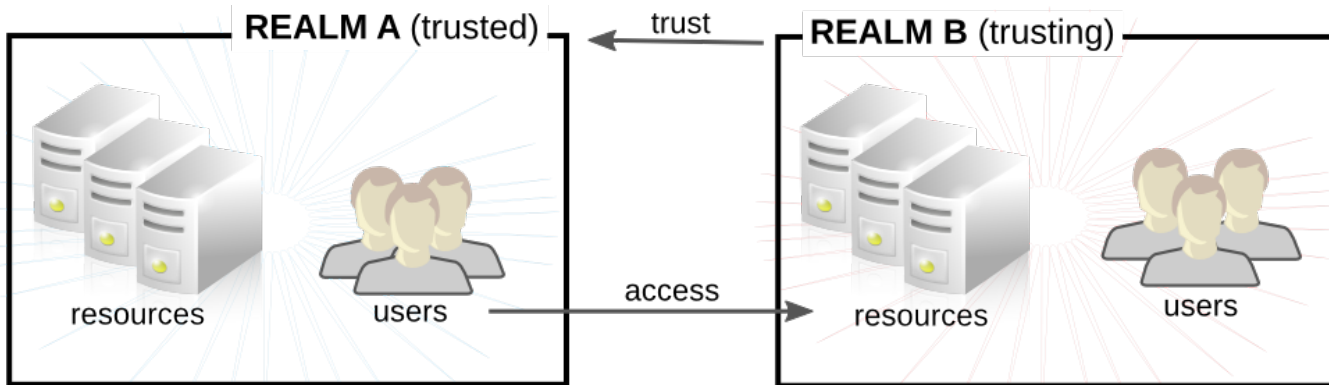


Figure 5.1. Basic Unidirectional Trusts

In many environments, there may be multiple domains which have trust agreements that are only explicit between one or two domains. There are two ways that trust can be handled: trusts can be followed recursively so that a domain trusts another domain and any other domain trusted by that second domain (*transitive trusts*), or a trust can be limited only to the explicitly included domains (*non-transitive trusts*).

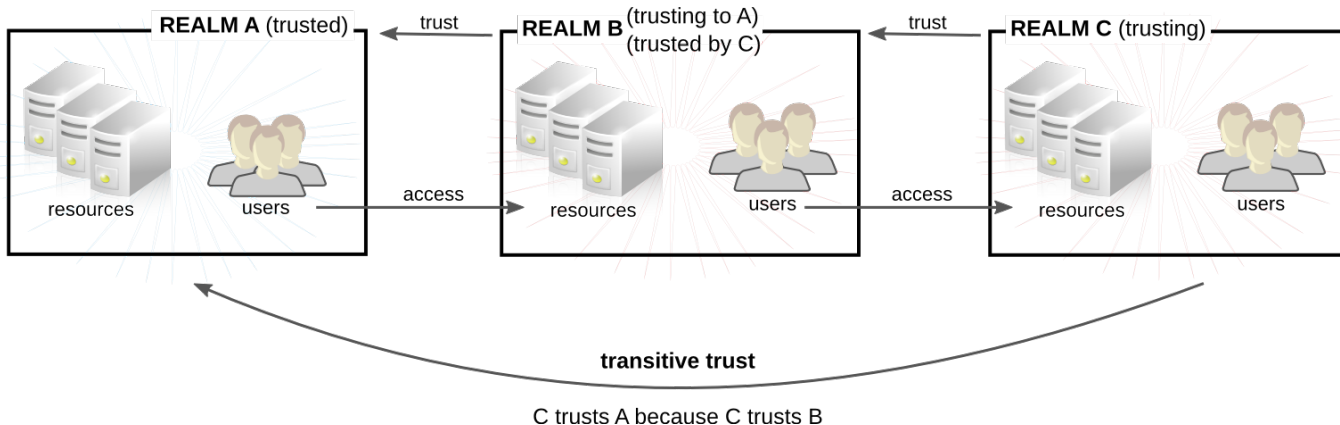


Figure 5.2. Transitive Trusts

Within a homogeneous Active Directory environment most (but not all) trusts are bidirectional and transitive by default. However, in a heterogeneous environment, Active Directory can establish a trust with a non-Active Directory domain, such as Identity Management in Red Hat Enterprise Linux. This is a *realm trust*, and, as Active Directory defines it, it has a slightly different default configuration than an Active Directory domain trust:

- ▶ It is unidirectional by default. (Even for bidirectional realm trusts, the actual configuration is set up so that there are two unidirectional trusts, pointing different ways.)
- ▶ It uses Kerberos authentication rather than Microsoft's native NTLM authentication.
- ▶ It is non-transitive by default, but some additional configuration can help support transitive trusts.
- ▶ The default direction is for the external (IdM) domain to trust the Active Directory domain. That means that the external domain trusts all security principles within the Active Directory domain.

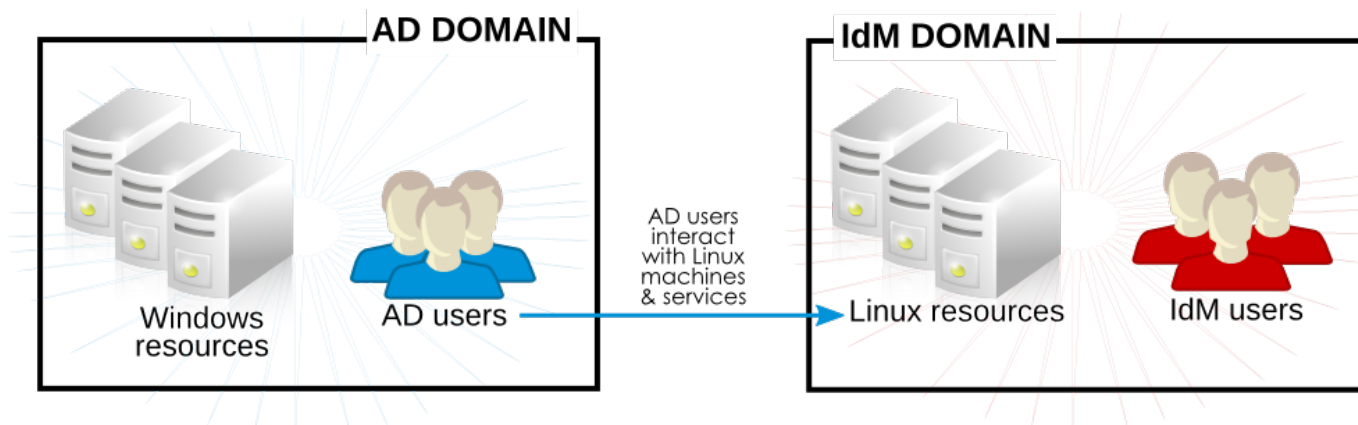


Figure 5.3. Trust Direction

5.1.2. Active Directory Principles, Security Objects, and Trust

The key to a trust is evaluating principles within the Active Directory domain as those users attempt to access resources in the external domain. The protocols that are used to evaluate those identities is the *trust path*. Active Directory trusts use NTLM to evaluate identities. Realm trusts — including the Identity Management trust — use Kerberos to create send and verify privileged access certificates and create Kerberos tickets for the external applications.

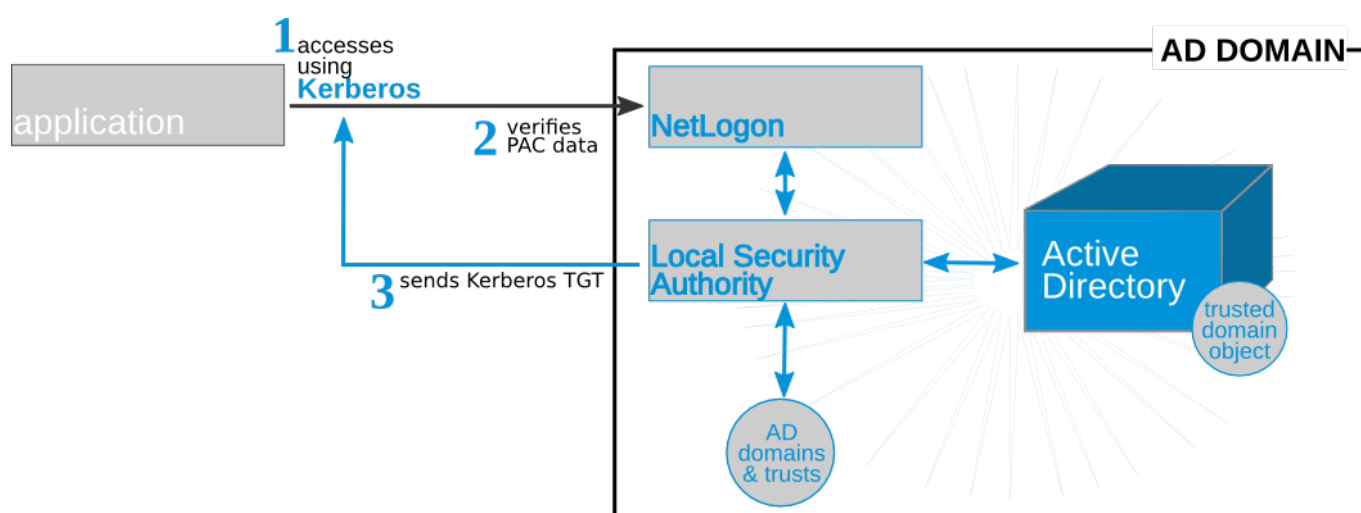


Figure 5.4. Authentication Path for Windows Trusts

The protocol which an application uses to process authentication requests is different at the front, either NTLM or Kerberos. Both of those authentication protocols interact with the Net Logon layer in Active Directory, and, from there, the process for accessing domain objects is the same. Each Active Directory server maintains a *local security authority* (LSA), which maintains all of the locally-defined security policies and provides a way to identify local users and identifiers, tickets and PACs, and other security data.



NOTE

All Kerberos communication for both Active Directory and IdM for trusts uses GSS-API.

Past the local security authority, there is the larger Active Directory configuration. The domain System container has all of the security information, including principles, user, and group information, for all of the objects in the domain. At the root of the domain is a global catalog of all users, groups, and objects within the entire forest. With a trust, information about Windows users can be retrieved from the System container or from the global catalog.

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 contains the required information about the user — their security ID, domain username, and group memberships. Identity Management has three components to analyze that data in the PAC on the Kerberos ticket:

- ▶ Samba, to perform identity lookups on Active Directory and retrieve user and group security identifiers (SIDs) for authorization information
- ▶ SSSD, to cache user, group, and ticket information for users and to map 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.

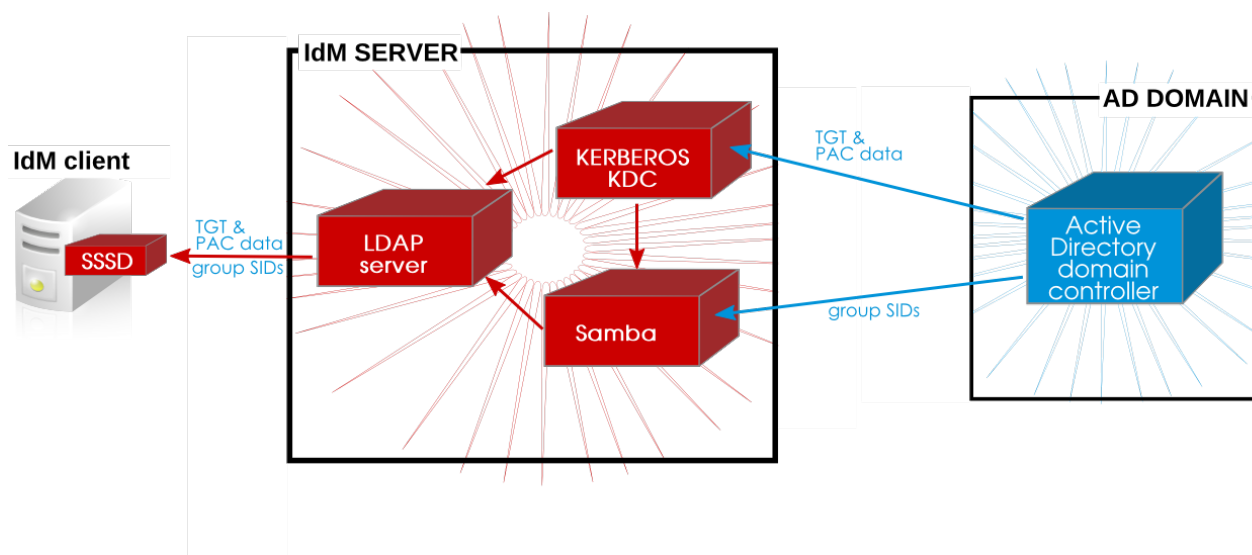


Figure 5.5. Applications and Services for Trust

5.1.3.1. Active Directory PACs and IdM Tickets

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. A PAC is essentially an account usability extension.

Understanding the group mapping for trusts can help clarify how groups should be structured in trust environments.

Microsoft uses a special authorization structure called *privileged access certificates* or MS-PAC. A PAC is embedded in a Kerberos ticket as a way of identifying the entity to other Windows clients and servers in the Windows domain.

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. The PAC information sent back by Active Directory is embedded in the Kerberos ticket.

IdM (through SSSD, as an IdM client), extracts the Active Directory group *security identifiers* (SIDs) from the PAC. IdM then compares the Active Directory SIDs in the PAC to the group SIDs configured as members in IdM groups. If the Active Directory group is a member of an IdM group, then the IdM group SID is added to the PAC, and the Kerberos ticket is updated with the new PAC.

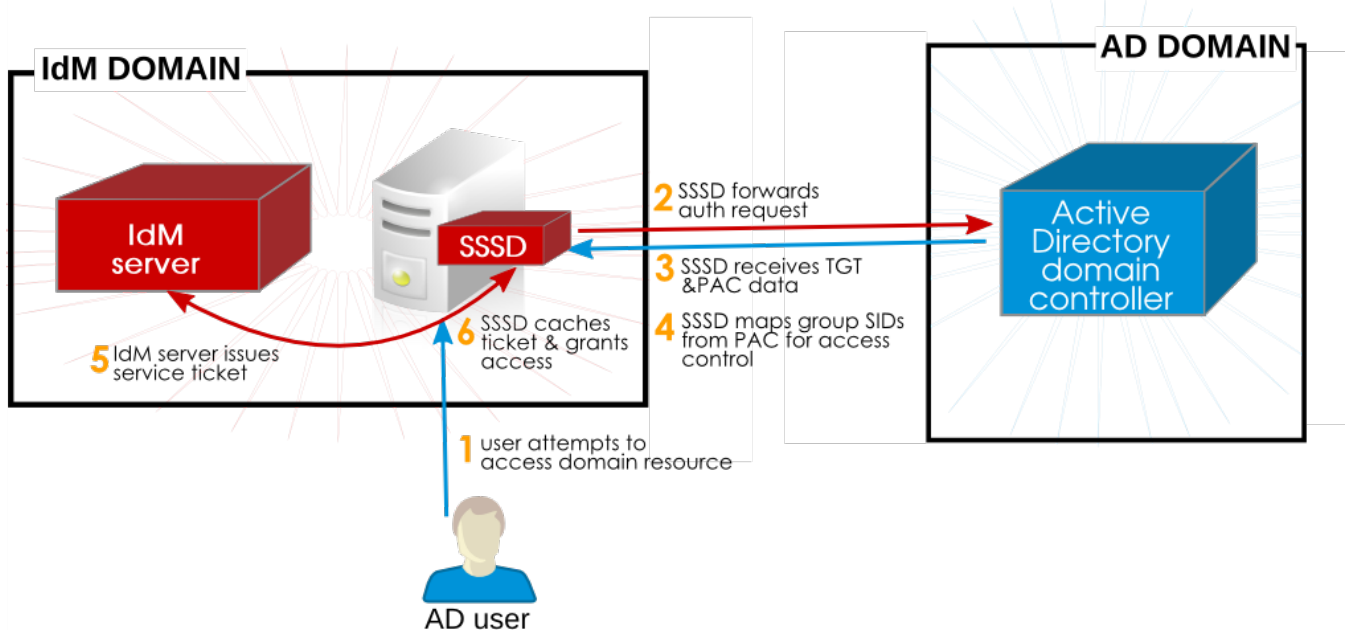


Figure 5.6. IdM, SSSD, and Active Directory

That new ticket is then used to generate a service ticket for the user, and the user is granted access to the IdM-hosted services, according to their access rules. Additionally, the IdM group information in the SSSD user cache is updated to include the mapped IdM groups for the Active Directory user.

SSSD stores multiple TGTs and tickets for each user, as new services are accessed.

A simpler way of saying this is that Active Directory supplies a list of groups for each user, based on an identifier for the group. IdM compares that list of Active Directory groups to memberships in IdM groups (where each group member is identified by that SID, rather than by a name or DN). If the Active Directory groups to which the user belongs are known to the IdM domain, then the user is recognized by the IdM domain.

5.1.3.2. Active Directory Users and IdM Groups

The crucial factor is that Active Directory users are recognized to the IdM domain not by their Active Directory user entry, but by their Active Directory group memberships. In a sense, Active Directory *users* are not trusted by the IdM domain — Active Directory *groups* are.

But this method of mapping Active Directory group SIDs to IdM group members means that group structure in IdM is important. Active Directory groups have different attributes than Linux groups and, therefore, different attributes than IdM groups. Most critically, Active Directory groups are not POSIX groups, while IdM groups are.

IdM uses an intermediary, non-POSIX group type, *external groups*, which allow entities outside IdM or a Linux system to be added as member. That external group can then be added to a standard IdM (POSIX) group as a member.

When Active Directory groups are added to an IdM group, they can be identified by their SID or by name, in the formats `DOMAIN\group_name` or `group_name@domain`. IdM then resolves the group name to the SID and stores the SID as the group member entry, to be compared to any offered user PAC.

Actually configuring groups for Active Directory users is described in [Section 5.4, “Creating IdM Groups for Active Directory Users”](#).

5.1.3.3. Mapping User IDs and Using POSIX Attributes

Active Directory user entries do not exist in the IdM server; they are not synchronized or copied over. Any information required by IdM resources is pulled in from Active Directory and cached. The unique identifiers on the Active Directory side (the security ID, a combination of the security domain identifier and the user identifier) are associated with usernames. When the username is used to access IdM resources, IdM (through Samba) resolves that username to its SID, and then looks up the information for that SID within the Active Directory domain.

Linux systems require that every user has a local UID number and group ID number. When users are created in IdM, the users are assigned UID/GID numbers by default. Even trusted users require a UI/GID number on a Linux system. That UID/GID number can be generated by IdM, but if the Active Directory entry already has UID/GID numbers assigned, then assigning different numbers creates a conflict. It is possible to use the Active Directory-defined POSIX attributes (including the UID/GID number, preferred login shell, and home directory location).

Active Directory stores a subset of information for all objects within the forest in a *global catalog*. This global catalog includes every entry for every domain in the forest. The global catalog can be queried to



NOTE

To use Active Directory-defined POSIX attributes, those attributes must be published to the global catalog. Otherwise, IdM generates user IDs based on the SID and the IdM ID range.

IdM auto-detects what kind of ID range to use when the trust is configured, but the range type can be manually set with the `ipa trust-add` command.

```
ipa trust-add -range-type=ipa-ad-trust-posix
```

Table 5.1. Types of Ranges

Range Option	Description
ipa-local	For IDs defined locally.
ipa-ad-winsync	For ID numbers assigned through the synchronization process.
ipa-ad-trust	For IDs set with SID - username mapping.
ipa-trust-posix	For IDs defined in POSIX attributes in the Active Directory entry.

Range Option	Description
ipa-ipa-trust	For IDs generated through trust configuration in the IdM server.

5.1.3.4. Active Directory Users and IdM Policies and Configuration

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

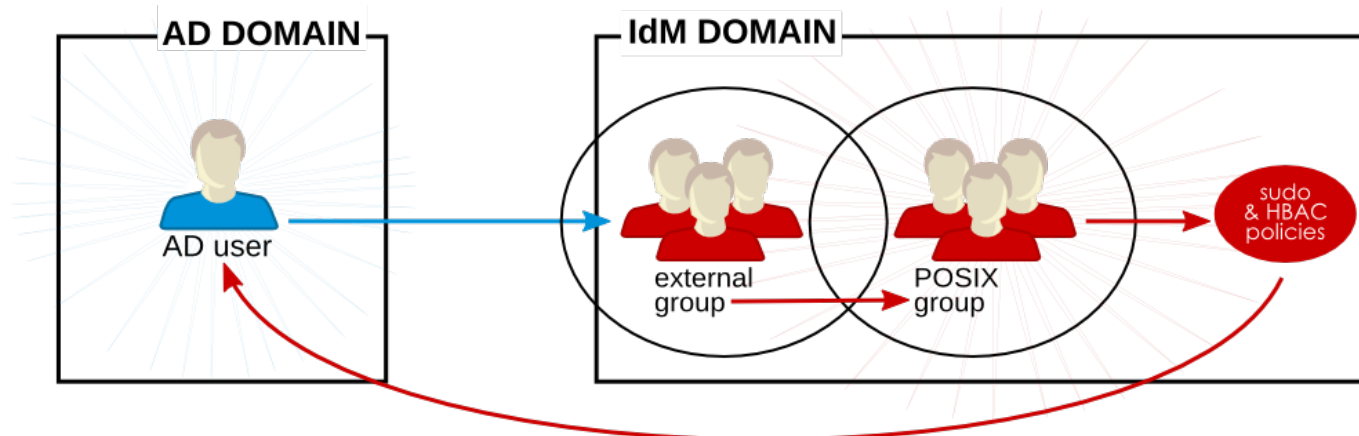


Figure 5.7. Active Directory Users and IdM Groups and Policies

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. External groups in IdM are non-POSIX groups. The external group is then added as a member of an IdM group (a POSIX group).

The sudo, host-based access controls, and other policies are applied to that POSIX group and, ultimately, to the Active Directory user when accessing IdM domain resources.

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



NOTE

Testing tools such as **hbactest** will not work with trusted users because the trusted user group associations are resolved dynamically, not stored in the IdM directory.

5.1.4. Different DNS-Trust Environments

Both Active Directory and Identity Management can define DNS services, and those DNS domains must interact cleanly with each other. There are two potential DNS configurations:

- ▶ The DNS domains can be independent.
- ▶ Identity Management can be configured as a subdomain of Active Directory.

In all cases, the different domains forward requests to each other as necessary and maintain different DNS namespaces. It is just a matter of defining how they recognize each other for forwarding queries.



IMPORTANT

DNS caches all data, at all levels of the DNS domain. The time to propagate any changes to the DNS records depends on the *cumulative* time for all of the configured time to live parameters.

Required: Identical Records in Superior and Subdomains

Within the Identity Management environment, every machine name must be fully resolvable. When using trusts, this includes machines within the trusted Active Directory domain. Depending on the configuration of the DNS environment, there can be a couple of different ways that the DNS services must be configured.

If IdM and Active Directory are subdomains within a larger, shared namespace or if IdM is a subdomain of the Active Directory DNS namespace, then the best configuration is to use *delegation* to create relationships between DNS domains. Delegation (NS) and glue (A or AAAA) records must be identical in every superior zone (such as **example.com**) and inferior zone (such as **ipa.example.com**). This means that the superior and subdomains must contain the exact same NS, A, and AAAA records.

If the Active Directory and IdM DNS domains are in entirely different namespaces, then use conditional forwarders. In that case, forwarding rules must be in place in both environments to allow all machines to be resolved.

Configuration Option 1: Separate DNS Domains

In this case, there are two entirely different namespaces, such as **ipaexample.com** and **adexample.com**. For these domains to communicate, they must be configured as conditional forwarders of each other's domain.

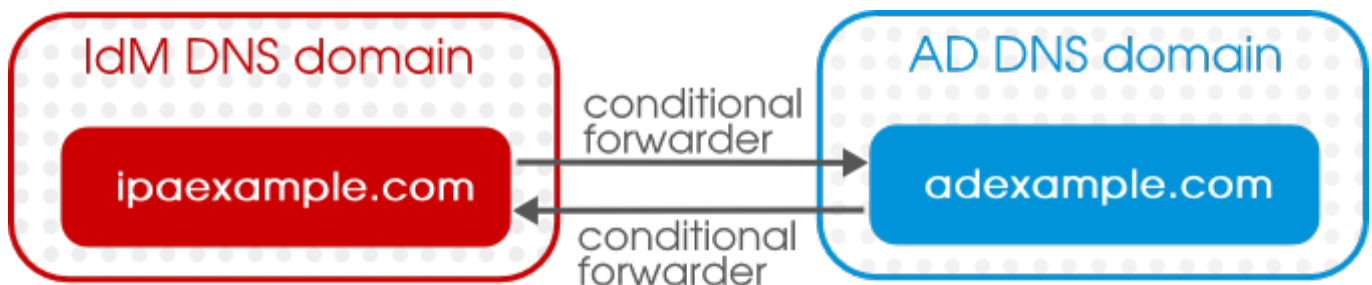


Figure 5.8. Trust with Separate DNS Domains

Configuration Option 2: Separate DNS Subdomains

A similar scenario is where both the Active Directory domain and the IdM domain are subdomains of a larger, central domain. For example, the Active Directory domain is **ad.example.com**, the Identity Management domain is **ipa.example.com**, and the superior domain for both is **example.com**. When using equivalent subdomains, *do not use forwarders* — use DNS delegation instead. Forwarders set in global configuration override any delegation rules, even if the local server is configured as authoritative. Instead, identical delegation rules should be set in the superior domain and the subdomains.

Configuration Option 3: Identity Management as a Subdomain of Active Directory

In this case, Identity Management is a namespace within the larger Active Directory space, such as **linux.example.com** and **example.com**. IdM can be configured to send all requests to the Active Directory domain (a forward-only policy) or it can send queries first to Active Directory and then attempt to resolve them itself (a forward-first policy).

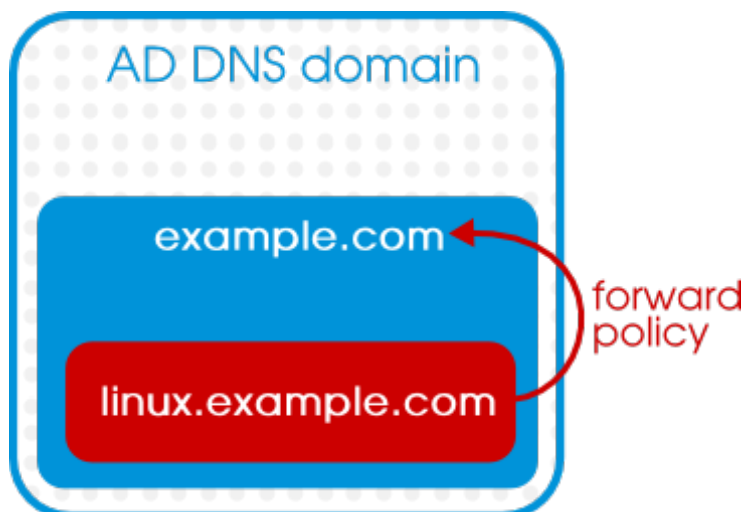


Figure 5.9. Trust with IdM as a DNS Subdomain of Active Directory

5.1.5. Identity Management Interactions with Multiple Domains

Whenever the IdM server connects to its peer server in trust, the information about the Active Directory DNS domains are fetched from the domain controller and stored in the `cn=Realm Domains,cn=ipa,cn=etc,suffix` subtree.

If there are multiple DNS domains defined within the Active Directory domain, then those DNS domains are added individually to the configuration subtree, and those are used to route queries to the appropriate domain.

5.1.6. Potential Behavior Issues with Active Directory Trust

5.1.6.1. Active Directory Users and IdM Administration

Trust relationships are unidirectional. Active Directory users exist only within the Active Directory domain and are limited to what resources within the IdM domain they can access. **Active Directory users can not be administrators for IdM because they do not exist within IdM.**

Active Directory users, then, cannot use any IdM administrative tools, including the web UI and command-line tools.

5.1.6.2. Authenticating Deleted Active Directory Users

By default, every IdM client uses the System Security Services Daemon to cache user identities and credentials. This allows the local system to reference identities for users who have already logged in successfully once, even if one of the backend providers (IdM or Active Directory) is temporarily unavailable.

Because SSSD maintains a list of users locally, it is possible that changes that are made on the backend are not immediately visible to clients.

If an Active Directory user has successfully logged into an IdM client resource, that user identity is cached in SSSD on both the local client and on the IdM server. If that Active Directory user is then deleted in Active Directory, that identity is *still* cached in IdM, which means that the user could successfully log into IdM resources.

The deleted Active Directory user will be able to log into IdM resources until the SSSD cache on any local client *and* on the IdM server expire. Then, when the IdM server attempts to retrieve the identity from Active Directory, it will receive notification that the user does not exist, and the login attempt fails.

5.1.6.3. Credential Cache and Selecting Active Directory Principals

The Kerberos credentials cache attempts to match a client principal to a server principal based on the service name, then the hostname, then (possibly) the realm name. Because the client/server mapping is done using the hostname and the realm name, this can create seemingly unexpected behavior in binding as an Active Directory user because the realm name of the Active Directory user is different than the realm name of the IdM system.

What this means in practice is that if an Active Directory user **kinit**s and then uses SSH to connect to an IdM resource, that principal is not selected for the resource ticket; an IdM principal is used because the IdM principal matches the resource's realm name.

For example, if the Active Directory 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
27.11.2013 11:25:23 27.11.2013 21:25:23  krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
renew until 28.11.2013 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
27.11.2013 11:25:23 27.11.2013 21:25:23  krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
renew until 28.11.2013 11:25:16

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

Valid starting      Expires            Service principal
27.11.2013 11:25:18 28.11.2013 11:25:16  krbtgt/EXAMPLE.COM@EXAMPLE.COM
27.11.2013 11:25:48 28.11.2013 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.

5.1.6.4. Resolving Group SIDs

Losing Kerberos Tickets

Running any command to obtain a SID from the Samba service — such as **net getlocalsid** or **net getdomainsid** — kills any existing admin ticket in the Kerberos cache.

Cannot Verify Group Membership for Users

There is no way 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

For Linux system users, local group associations can be shown for a user using the `id` command. However, Active Directory group memberships are not displayed with `id` for Active Directory users, even though they are with Samba tools.

The `wbinfo` command can be used to obtain a SID for an Active Directory user and then to display groups associated with that SID.

```
[root@ipaserver ~]# wbinfo -n ADDDOMAIN\jsmith
S-1-5-21-1689615952-3716327440-3249090444-1104 SID_USER (1)

[root@ipaserver ~]# wbinfo --user-domgroups=S-1-5-21-1689615952-3716327440-
3249090444-1104
S-1-5-21-1689615952-3716327440-3249090444-513
S-1-5-21-1689615952-3716327440-3249090444-1106
```

The same query using `id` shows only the user information, not the Active Directory group membership information.

```
[root@ipaserver ~]# id ADDDOMAIN\jsmith
uid=1921801104(jsmith@adexample.com) gid=1921801104(jsmith@adexample.com)
groups=1921801104(jsmith@adexample.com)
```



TIP

To work around this, ssh into an IdM client machine as the given Active Directory user. After the first successful login, the Active Directory group memberships are detected and returned in the `id` search.

```
[root@ipaserver ~]# id ADDDOMAIN\jsmith
uid=1921801107(jsmith@adexample.com) gid=1921801107(jsmith@adexample.com)
groups=1921801107(jsmith@adexample.com), 129600004(ad_users), 1921800513(domain
users@adexample.com)
```

5.2. Environment and Machine Requirements to Set up Trusts

Make sure that both the Active Directory and IdM servers, machines, and environments meet the requirements and settings in this section *before* configuring a trust agreement.

5.2.1. Supported Windows Platforms

Trust relationships can be configured with these Windows server versions:

- ▶ Windows Server 2008 R2 (32-bit)
- ▶ Windows Server 2008 R2 (64-bit)

5.2.2. Domain and Realm Names

The IdM DNS domain name and Kerberos realm name *must* be different than the Active Directory DNS domain name and Kerberos realm name.

5.2.3. NetBIOS Names

The NetBIOS name is the far-left component of the domain name. For example, if the domain is *linux.example.com*, the NetBIOS name is *linux*, while if the domain name is simply *example.com*, it is *example*. The NetBIOS name is critical for identifying the Active Directory domain and, if the IdM domain is within a subdomain of Active Directory DNS, for identifying the IdM domain and services.

The IdM domain and Active Directory domain must have different NetBIOS names.

5.2.4. Integrated DNS

Both the Active Directory server and the IdM server must be configured to run their own respective DNS services.

5.2.5. Integrated Certificate Authorities

Both Active Directory and Identity Management must be configured with integrated certificate services.

5.2.6. Firewalls and Ports

Required Ports

For a trust relationship, the Active Directory server and IdM server must have almost all of the required system ports open that are required for an IdM server installation, **with the exception of the LDAP ports**.

Table 5.2. IdM Ports

Service	Ports	Type
HTTP/HTTPS	80	TCP
	443	
Kerberos	88	TCP and UDP
	464	
DNS	53	TCP and UDP
NTP	123	UDP



IMPORTANT

The IdM backend LDAP server *must not be reachable* by the Active Directory domain controller. The associated ports — 389 and 636 — on the IdM server host must be shut down for the Active Directory domain controller.

Starting firewalld at Boot Time

Configure the `firewalld` service to start when the system boots:

```
[root@ipaserver ]# chkconfig firewalld on
```


Setting firewalld Configuration

The **firewalld** configuration needs to allow access to the required IdM ports and reject access to the IdM LDAP ports. The order of the rules is important. Active Directory-based requests to LDAP ports must be blocked first (based on the Active Directory server IP address), then there must be connections allowed to all IdM TCP and UDP ports.

1. Open the **firewalld** configuration file.

```
[root@ipaserver ~]# vim /etc/firewalld
```

2. Add the rule to restrict access to LDAP ports for the Active Directory host.

```
-A INPUT -s ad_ip_address -p tcp -m multiport --dports 389,636 -m state --state NEW,ESTABLISHED -j REJECT
```

3. Make sure that there lines to allow access to the TCP and UDP ports required by IdM.

```
-A INPUT -p tcp -m multiport --dports 80,443,389,636,88,464,53,138,139,445 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m multiport --dports 88,464,53,123,138,139,389,445 -m state --state NEW,ESTABLISHED -j ACCEPT
```

4. Save the file.
5. Restart the **firewalld** service:

```
[root@ipaserver ~]# systemctl restart firewalld.service
```

Example 5.1. Example firewalld Configuration File

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -s ad_ip_address -p tcp -m multiport --dports 389,636 -m state --state NEW,ESTABLISHED -j REJECT
-A INPUT -p tcp -m multiport --dports 80,443,389,636,88,464,53,138,139,445 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m multiport --dports 88,464,53,123,138,139,389,445 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p udp -j REJECT
-A INPUT -p tcp -j REJECT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

5.2.7. IPv6 Settings

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

5.2.8. Clock Settings

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

5.2.9. Supported Username Formats

Username mapping is performed in the local SSSD client. A Python regular expression is used by SSSD to identify the username and the domain to which it belongs.

By default in SSSD, the username format is defined in the form *name@domain*. This uses the regular expression:

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

Active Directory can support several different kinds of name formats, however, so the *re_expression* parameter in the SSSD configuration file for IdM backends or Active Directory backends uses a more complex expression:

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

This supports usernames in multiple formats:

- ▶ *username*
- ▶ *username@domain.name*
- ▶ *DOMAIN\username*



TIP

An additional SSSD parameter, *default_domain_suffix*, can be used to supply a default domain value for usernames. For example, if all users are in a trusted Active Directory domain of **adexample.com** and the identity backend is the IdM domain of **ipa.example.com**, the *default_domain_suffix* parameter can be set with the value **adexample.com**. All users are automatically assumed to belong to that user domain unless the domain value is explicitly given with the username.

5.3. Creating Trusts

5.3.1. Setting up Trust with IdM as a DNS Subdomain of Active Directory

The central purpose of a trust is to create a relationship between Kerberos realms and DNS domains. The Kerberos configuration is fairly consistent, but there can be very different DNS configurations for the IdM and Active Directory domains. [Configuration Option 3: Identity Management as a Subdomain of Active Directory](#) describes the configuration if IdM is a subdomain within a larger Active Directory-managed DNS domain.

1. Stop the Windows firewall service.
2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Install the required trust packages, updated Samba4 packages, and LDAP-DNS packages for IdM DNS management.

```
[root@ipaserver ]# yum install ipa-server "*ipa-server-trust-ad" samba-winbind-clients bind-dyndb-ldap samba-client
```

4. Set up the IdM server to use its own, integrated DNS services (**--setup-dns**), its own DNS domain (**-n**), and the Active Directory DNS server as a forwarder (**--forwarder**). For example:

```
[root@ipaserver ]# ipa-server-install --setup-dns --forwarder=255.255.255.255 -p secret -a secret -r IPAEXAMPLE -n linux.adexample.com --hostname ipaserver.linux.adexample.com -U
```

5. Add the IdM domain as a subdomain entry in the Active Directory configuration and add an NS record for the IdM DNS. For this example, the IdM configuration has a NetBIOS name of **linux** (the subdomain) and the domain name is **adexample.com**.
 - a. Open the command prompt, using *Run as Administrator*.
 - b. Use the **dnscmd** command to add the A record for the IdM server, using the IdM hostname, NetBIOS name, and IP address.

```
/RecordAdd ad_domain ipa_hostname.ipa_netbios A ipa_ip_address
```

For example:

```
C:\> dnscmd 127.0.0.1 /RecordAdd adexample.com ipaserver.linux A 255.255.255.0
```

- c. Then add the NS record for the IdM server. This has the format:

```
/RecordAdd ad_domain ipa_netbios NS ipa_hostname.ipa_subdomain
```

For example:

```
C:\> dnscmd 127.0.0.1 /RecordAdd adexample.com linux NS ipaserver.linux.adexample.com
```

6. Check the SRV records for both domains from both servers.

On the IdM server, use the **dig SRV** command to list the records for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# dig SRV _ldap._tcp.adexample.com
; ANSWER SECTION:
_ldap._tcp.adexample.com. 600      IN      SRV     0 100 389
adserver.adexample.com.
; ADDITIONAL SECTION:
adserver.adexample.com. 3600   IN      A       192.168.2.161
; ADDITIONAL SECTION:
adserver.adexample.com. 3600   IN      A       192.168.2.161

[root@ipaserver ~]# dig SRV _ldap._tcp.linux.adexample.com
; ANSWER SECTION:
_ldap._tcp.linux.adexample.com. 86400  IN      SRV     0 100 389
ipaserver.linux.adexample.com.
```

```
;; AUTHORITY SECTION:
linux.adexample.com.      86400    IN      NS
ipaserver.linux.adexample.com.
;; ADDITIONAL SECTION:
ipaserver.linux.adexample.com.  1200    IN      A      192.168.2.158
```

On the Active Directory server, open the **nslookup** tool and check the corresponding SRV records.

```
> nslookup
> set type=srv
> _ldap._tcp.adexample.com
> _ldap._tcp.linux.adexample.com
> quit
```

7. Enable DNS lookups in the Kerberos realm for the Kerberos client.

a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

b. In the **[libdefaults]** section, add or set the **dns_lookup_kdc** value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

8. Configure the IdM server to enable trust services.

a. Run the **ipa-adtrust-install** command.

This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator. These can be passed with the script; otherwise, it prompts for them.

```
[root@ipaserver ]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret
```

Optionally, use the **-U** argument to run the script non-interactively.

b. The script prompts to configure the **slapi-nis** plug-in. This is 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
```

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The **smbclient** command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----      -
  IPC$           IPC             IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server          Comment
  -----
  Workgroup       Master
  -----
```

The **wbinfo** command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
BUILTIN : online
IPAEXAMPLE : online
```

10. *If there are existing IdM users and groups.* For existing IdM users, it is required that all users and groups have an Active Directory-style security identifier (SID). A new **ipaNTSecurityIdentifier** containing a SID can be created automatically for each entry by running a special **ipa-sidgen-task** operation on the backend LDAP directory.

If there are no existing IdM users or groups, then this step can be skipped.

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

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"
```

When the task completes successfully, there will be a message 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-IPALAB-
QE/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].
```

11. Create a trust agreement for the Active Directory domain and the IdM domain. This command requires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=type ad_domain_name --admin ad_admin_username --password
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
Realm name: adexample.com
Domain NetBIOS name: AEXAMPLE
Domain Security Identifier: S-1-5-21-1689615952-3716327440-3249090444
Trust direction: Two-way trust
Trust type: Active Directory domain
Trust status: Established and verified
```

12. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

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

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@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.ipaexample.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.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

13. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
```

```
....
dns_lookup_kdc = true
```

- c. 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. One rule should have a value of `DEFAULT`, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format `first.last@ADDOMAIN`, `username@ADDOMAIN[.something]`, or `username@addomain[.something]`; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
auth_to_local = DEFAULT
}
```

Alternatively, the username mapping can be used by creating `.k5login` files for each user. This is covered in [Section 5.8.2, “Using SSH Without Passwords”](#).

- d. Restart the KDC service.

```
[root@ipaserver ~]# service krb5kdc restart
```

14. Configure domain mapping in SSSD.

- a. Open the `/etc/sss/sss.conf`.

```
[root@ipaserver ~]# vim /etc/sss/sss.conf
```

- b. In the `[sss]` section, add `pac` to the `services` list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

- c. In the IdM domain section, add the `subdomains_provider` parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed;

adding this line makes it possible to use the existing configuration.

- d. Save the changes to the **sssd.conf** file.
- e. Restart SSSD.

```
[root@ipaserver ]# systemctl restart sssd.service
```

15. Restart the **firewalld** services on the IdM server.

```
[root@ipaserver ]# systemctl start firewalld.service
```

16. Restart the Windows firewall.

5.3.2. Setting up Trust with IdM and Active Directory in Different DNS Domains

The central purpose of a trust is to create a relationship between Kerberos realms and DNS domains. The Kerberos configuration is fairly consistent, but there can be very different DNS configurations for the IdM and Active Directory domains. [Configuration Option 1: Separate DNS Domains](#) describes the configuration if IdM and Active Directory are in separate DNS domains.

1. Stop the Windows firewall service.
2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Install the required trust packages, updated Samba4 packages, and LDAP-DNS packages for IdM DNS management.

```
[root@ipaserver ]# yum install ipa-server "*ipa-server-trust-ad" samba-winbind-clients bind-dyndb-ldap samba-client
```

4. Set up the IdM server to use its own, integrated DNS services (**--setup-dns**), its own DNS domain (**-n**), and the Active Directory DNS server as a forwarder (**--forwarder**). For example:

```
[root@ipaserver ]# ipa-server-install --setup-dns --forwarder=ad-dns.adserver.example.com -p secret -a secret -r IPAEXAMPLE -n ipaexample.com -hostname ipaserver.ipaexample.com -U
```

5. Add the IdM server as a conditional forwarder in the Active Directory DNS configuration.
 - a. Open the **Administrative Tools** menu, and select the **DNS** item.
 - b. Right-click the **Conditional Forwarders** item in the left column of the window.
 - c. Select the **New Conditional Forwarder...** button.
 - d. Enter the DNS domain name of the IdM domain and the IP address of the IdM DNS server.
 - e. Save the new forwarder.

Alternatively, use the **dnscmd** command-line utility to add the forwarder entry:

```
> dnscmd 127.0.0.1 /ZoneAdd IPAEXAMPLE.COM /Forwarder 255.255.255.0
```


6. Check the SRV records for both domains from both servers.

On the IdM server, use the **dig SRV** command to list the records for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# dig SRV _ldap._tcp.adexample.com
;; ANSWER SECTION:
_ldap._tcp.adexample.com. 600      IN      SRV     0 100 389
adserver.adexample.com.
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600    IN      A       192.168.2.161

[root@ipaserver ~]# dig SRV _ldap._tcp.ipaexample.com
;; ANSWER SECTION:
_ldap._tcp.ipaexample.com. 86400   IN      SRV     0 100 389
ipaserver.ipaexample.com.
;; AUTHORITY SECTION:
ipaexample.com. 86400   IN      NS      ipaserver.ipaexample.com.
;; ADDITIONAL SECTION:
ipaserver.ipaexample.com. 1200    IN      A       192.168.2.158
```

On the Active Directory server, open the **nslookup** tool and check the corresponding SRV records.

```
> nslookup
> set type=srv
> _ldap._tcp.adexample.com
> _ldap._tcp.ipaexample.com
> quit
```

7. Enable DNS lookups in the Kerberos realm for the Kerberos client.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ~]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, add or set the **dns_lookup_kdc** value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

8. Configure the IdM server to enable trust services.

- a. Run the **ipa-adtrust-install** command.

This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator. These can be passed with the script; otherwise, it prompts for them.

```
[root@ipaserver ~]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret
```

Optionally, use the **-U** argument to run the script non-interactively.

- b. The script prompts to configure the slapi-nis plug-in. This is 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
```

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The **smbclient** command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----      -
  IPC$           IPC             IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server          Comment
  -----
  Workgroup       Master
  -----
```

The **wbinfo** command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
BUILTIN : online
IPAEXAMPLE : online
```

10. Create a trust agreement for the Active Directory domain and the IdM domain. This command requires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=type ad_domain_name --admin ad_admin_username --password
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
  Realm name: adexample.com
  Domain NetBIOS name: ADEXAMPLE
  Domain Security Identifier: S-1-5-21-1689615952-3716327440-3249090444
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  Trust status: Established and verified
```

11. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

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

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@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.ipaexample.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.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

12. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
....
dns_lookup_kdc = true
```

c. 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. One rule should have a value of **DEFAULT**, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format *first.last@ADDOMAIN*, *username@ADDOMAIN[.something]*, or *username@adomain[.something]*; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDDOMAIN$)s/@ADDDOMAIN/@adddomain/
auth_to_local = DEFAULT
}
```

- d. Restart the KDC service.

```
[root@ipaserver ~]# systemctl restart krb5kdc.service
```

13. Configure domain mapping in SSSD.

- a. Open the `/etc/sss/sss.conf`.

```
[root@ipaserver ]# vim /etc/sss/sss.conf
```

- b. In the `[sss]` section, add `pac` to the `services` list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

- c. In the IdM domain section, add the `subdomains_provider` parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed; adding this line makes it possible to use the existing configuration.

- d. Save the changes to the `sss.conf` file.
- e. Restart SSSD.

```
[root@ipaserver ]# systemctl restart sss.service
```

14. Restart the `firewalld` services on the IdM server.

```
[root@ipaserver ]# systemctl start firewalld.service
```

15. Restart the Windows firewall.

5.3.3. Setting up Trust with a Shared Secret

Trusts within Active Directory can be configured with a shared secret, a password that is known to trusted peers and can be used by other domains to join the trust. In Active Directory, this is stored as a *trusted domain object* within the trust configuration. The secret is updated periodically.

A trust can be created in IdM using a shared secret rather than the Active Directory administrator credentials, but this requires additional configuration in Active Directory to create the shared secret. Additionally, the trust must be validated manually on the Active Directory side and then the Active Directory topology must be manually loaded on the IdM side to complete the configuration.

1. Stop the Windows firewall service.
2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Install the required trust packages, updated Samba4 packages, and LDAP-DNS packages for IdM DNS management.

```
[root@ipaserver ]# yum install ipa-server "*ipa-server-trust-ad" samba-winbind-clients bind-dyndb-ldap samba-client
```

4. Set up the IdM server. This example configures its integrated DNS services (**--setup-dns**), its own DNS domain (**-n**), and the Active Directory DNS server as a forwarder (**--forwarder**). The value of the **-n** and **--hostname** arguments is slightly different, depending on the planned DNS configuration. **-n** gives the name of the IdM domain, and **--hostname** gives the machine name. This example shows the IdM domain as a subdomain of the Active Directory domain.

```
[root@ipaserver ]# ipa-server-install --setup-dns --forwarder=2555.255.255.255 -p secret -a secret -r IPAEXAMPLE -n linux.adexample.com --hostname ipaserver.linux.adexample.com -U
```

5. Set up the DNS realms. If IdM will be a subdomain of Active Directory, then follow step 5 in [Section 5.3.1, “Setting up Trust with IdM as a DNS Subdomain of Active Directory”](#). If IdM will host its own domain, then follow step 5 in [Section 5.3.2, “Setting up Trust with IdM and Active Directory in Different DNS Domains”](#).
6. Enable DNS lookups in the Kerberos realm for the Kerberos client.
 - a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, add or set the **dns_lookup_kdc** value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

7. Set up the external trust on Active Directory. This sets the trust password, which is the shared secret used in configuring the trust on IdM.

An external trust is configured in the **Active Directory Domains and Trusts** console. Select the appropriate domain, and create a new trust. These are the settings to use:

- ▶ The **Trust Type** is **Realm**.
- ▶ The **Transitivity of Trust** can be either transitive or non-transitive.
- ▶ The **Direction of Trust** is **One-way: incoming**.
- ▶ The **Sides of Trust** is **This domain only**.
- ▶ The **Trust Password** can be anything. This must be used when configuring the trust in IdM.

The full instructions are in the [Microsoft TechNet documentation](#).

8. Configure the IdM server to enable trust services.

- a. Run the **ipa-adtrust-install** command.

This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator. These can be passed with the script; otherwise, it prompts for them.

```
[root@ipaserver ~]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret
```

Optionally, use the **-U** argument to run the script non-interactively.

- b. The script prompts to configure the slapi-nis plug-in. This is 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
```

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The **smbclient** command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----      -
  IPC$           IPC             IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server                Comment
  -----
  Workgroup             Master
  -----
```

The **wbinfo** command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
```

```
BUILTIN : online
IPAEXAMPLE : online
```

10. Create a trust agreement for the Active Directory domain and the IdM domain, specifying the quires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=ad ad_domain_name --trust-secret
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --trust-secret
Shared secret for the trust:
-----
Added Active Directory trust for realm "adexample.com"
-----
  Realm name: adexample.com
  Domain NetBIOS name: ADEXAMPLE
  Domain Security Identifier: S-1-5-21-1910160501-511572375-3625658879
  SID blacklist incoming: S-1-0, S-1-1, S-1-2, S-1-3, S-1-5-1, S-1-5-2, S-1-
5-3, S-1-5-4, S-1-5-5, S-1-5-6, S-1-5-7, S-1-5-8, S-1-5-9, S-1-5-10, S-1-5-
11, S-1-5-12, S-1-5-13, S-1-5-14, S-1-5-15, S-1-5-16,
                        S-1-5-17, S-1-5-18, S-1-5-19, S-1-5-20
  SID blacklist outgoing: S-1-0, S-1-1, S-1-2, S-1-3, S-1-5-1, S-1-5-2, S-1-
5-3, S-1-5-4, S-1-5-5, S-1-5-6, S-1-5-7, S-1-5-8, S-1-5-9, S-1-5-10, S-1-5-
11, S-1-5-12, S-1-5-13, S-1-5-14, S-1-5-15, S-1-5-16,
                        S-1-5-17, S-1-5-18, S-1-5-19, S-1-5-20
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  Trust status: Waiting for confirmation by remote side
```

11. On the Active Directory server, validate the new trust. This is done in the **Domains and Trusts** console.
12. On the IdM server, fetch the Active Directory domain topology.

```
[root@ipaserver ~]# kinit admin
[root@ipaserver ~]# ipa trust-fetch-domains adexample.com
-----
List of trust domains successfully refreshed
-----
  Realm name: test.adexample.com
  Domain NetBIOS name: TEST
  Domain Security Identifier: S-1-5-21-91314187-2404433721-1858927112
-----
Number of entries returned 1
-----
```

13. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

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

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ~]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@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.ipaexample.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.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

14. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
....
dns_lookup_kdc = true
```

- c. 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. One rule should have a value of **DEFAULT**, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format *first.last@ADDOMAIN*, *username@ADDOMAIN[.something]*, or *username@addomain[.something]*; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
auth_to_local = DEFAULT
}
```

Alternatively, the username mapping can be used by creating **.k5login** files for each user. This is covered in [Section 5.8.2, “Using SSH Without Passwords”](#).

- d. Restart the KDC service.


```
[root@ipaserver ~]# systemctl restart krb5kdc.service
```

15. Configure domain mapping in SSSD.

a. Open the `/etc/sss/sss.conf`.

```
[root@ipaserver ]# vim /etc/sss/sss.conf
```

b. In the `[sss]` section, add `pac` to the `services` list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

c. In the IdM domain section, add the `subdomains_provider` parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed; adding this line makes it possible to use the existing configuration.

d. Save the changes to the `sss.conf` file.

e. Restart SSSD.

```
[root@ipaserver ]# systemctl restart sss.service
```

16. Restart the `firewalld` services on the IdM server.

```
[root@ipaserver ]# systemctl start firewalld.service
```

17. Restart the Windows firewall.

5.3.4. Creating Trusts on an Existing IdM Instance

When establishing a trust on a new IdM instance, certain configurations for both the IdM server and entries within its domain are set from the beginning. However, when a trust is configured for an existing IdM instance, at least two things need to be done: the DNS configuration needs to be set for the Active Directory domain and all existing IdM users and groups need to be assigned Active Directory-style SIDs.

1. Stop the Windows firewall service.

2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Install the required trust packages.

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

4. Add the Active Directory DNS domain as a forwarder for the IdM server.

```
[root@ipaserver ]# ipa dnsconfig-mod --forwarder=255.255.255.255 --forward-
policy=first
```

Using a first policy means that queries are sent to the forwarder first and then to the local **named** process. Alternatively, this can be set to only, so that only the DNS forwarder is queried, never **named**.

5. Set up the DNS realms. If IdM will be a subdomain of Active Directory, then follow step 5 in [Section 5.3.1, “Setting up Trust with IdM as a DNS Subdomain of Active Directory”](#). If IdM will host its own domain, then follow step 5 in [Section 5.3.2, “Setting up Trust with IdM and Active Directory in Different DNS Domains”](#).
6. Check the SRV records for both domains from both servers.

On the IdM server, use the **dig SRV** command to list the records for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# dig SRV _ldap._tcp.adexample.com
;; ANSWER SECTION:
_ldap._tcp.adexample.com. 600      IN      SRV     0 100 389
adserver.adexample.com.
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600   IN      A       192.168.2.161
;; ADDITIONAL SECTION:
adserver.adexample.com. 3600   IN      A       192.168.2.161

[root@ipaserver ~]# dig SRV _ldap._tcp.linux.adexample.com
;; ANSWER SECTION:
_ldap._tcp.linux.adexample.com. 86400  IN      SRV     0 100 389
ipaserver.linux.adexample.com.
;; AUTHORITY SECTION:
linux.adexample.com. 86400  IN      NS
ipaserver.linux.adexample.com.
;; ADDITIONAL SECTION:
ipaserver.linux.adexample.com. 1200   IN      A       192.168.2.158
```

On the Active Directory server, open the **nslookup** tool and check the corresponding SRV records.

```
> nslookup
> set type=svr
> _ldap._tcp.adexample.com
> _ldap._tcp.linux.adexample.com
> quit
```

7. Enable DNS lookups in the Kerberos realm for the Kerberos client.

- a. Open the `/etc/krb5.conf` configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the `[libdefaults]` section, add or set the `dns_lookup_kdc` value to true.

```
[libdefaults]
....
dns_lookup_kdc = true
```

8. Configure the IdM server to enable trust services.

- a. Run the `ipa-adtrust-install` command.

This requires the NetBIOS name **of the IdM server** and the password of the IdM administrator. These can be passed with the script; otherwise, it prompts for them.

```
[root@ipaserver ]# ipa-adtrust-install --netbios-name=IPAEXAMPLE -a secret
```

Optionally, use the `-U` argument to run the script non-interactively.

- b. The script prompts to configure the `slapi-nis` plug-in. This is 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.

Select no (the default).

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

9. To verify the IdM configuration at this point, use the Samba tools to check that the Windows-related services are running and accessible. The `smbclient` command shows whether the domain is in the Samba registry.

```
[root@ipaserver ~]# smbclient -L ipaserver.ipaexample.com -k
lp_load_ex: changing to config backend registry
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Sharename      Type            Comment
  -----      -
  IPC$           IPC            IPC Service (Samba 4.0.0rc4)
Domain=[IPAEXAMPLE] OS=[Unix] Server=[Samba 4.0.0rc4]
  Server                Comment
  -----
  Workgroup            Master
  -----
```

The `wbinfo` command shows whether the IdM domain is online.

```
[root@ipaserver ~]# wbinfo --online-status
BUILTIN : online
IPAEXAMPLE : online
```

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

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

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"
```

When the task completes successfully, there will be a message 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-IPALAB-
QE/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].
```

11. Create a trust agreement for the Active Directory domain and the IdM domain. This command requires the Active Directory domain and the credentials of an administrative user to use to connect to the domain.

```
ipa trust-add --type=type ad_domain_name --admin ad_admin_username --password
```

For example:

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
  Realm name: adexample.com
  Domain NetBIOS name: ADEXAMPLE
  Domain Security Identifier: S-1-5-21-1689615952-3716327440-3249090444
  Trust direction: Two-way trust
  Trust type: Active Directory domain
  Trust status: Established and verified
```

12. Request a ticket for an IdM user to check the Kerberos configuration, and then check that that user can request service tickets.

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

First, request service tickets for services within the IdM domain.

```
[root@ipaserver ]# kvno host/ipaserver.ipaexample.com@IPA.DOMAIN
```

Then, request service tickets for services within the Active Directory domain.

```
[root@ipaserver ]# kvno cifs/adserver.adexample.com@AD.DOMAIN
```

If the Active Directory service ticket is successfully granted, then there will be a cross-realm TGT listed with all of the other requested tickets. This will have the name **krbtgt/AD.DOMAIN@IPA.DOMAIN**.

```
[root@ipaserver ]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: jsmith@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.ipaexample.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.adexample.com@AD.DOMAIN
```



NOTE

This ticket is requested as an IdM user because Kerberos realm mappings are not yet configured to allow Active Directory users to use Kerberos authentication to the realm.

13. Configure realm mapping in the Kerberos configuration. This allows Kerberos authentication for Active Directory users.

- a. Open the **/etc/krb5.conf** configuration file.

```
[root@ipaserver ]# vim /etc/krb5.conf
```

- b. In the **[libdefaults]** section, enable DNS lookups in the Kerberos realm.

```
[libdefaults]
....
dns_lookup_kdc = true
```

- c. 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. One rule should have a value of **DEFAULT**, for standard Unix usernames, and the other should include a rule which maps different Active Directory username formats and the specific Active Directory domain. For example, this rule allows usernames in the format *first.last@ADDOMAIN*, *username@ADDOMAIN[.something]*, or *username@addomain[.something]*; the last two expressions allow upper-case or lower-case domain names, since Kerberos is case-sensitive.

```
[realms]
IDM = {
....
auth_to_local = RULE:[1:$1@$0](^.*@ADDOMAIN$)s/@ADDOMAIN/@addomain/
auth_to_local = DEFAULT
}
```

Alternatively, the username mapping can be used by creating `.k5login` files for each user. This is covered in [Section 5.8.2, “Using SSH Without Passwords”](#).

- d. Restart the KDC service.

```
[root@ipaserver ~]# service krb5kdc restart
```

14. Configure domain mapping in SSSD.

- a. Open the `/etc/sss/sss.conf`.

```
[root@ipaserver ~]# vim /etc/sss/sss.conf
```

- b. In the `[sss]` section, add `pac` to the `services` list to enable the SSSD service to request and use Kerberos tickets with PAC data.

```
[sss]
services = nss, pam, ssh, pac
....
```

- c. In the IdM domain section, add the `subdomains_provider` parameter to explicitly enable SSSD to refer from the configured IdM domain to any domains trusted by that domain.

```
[domain/ipa.lan]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example2b.com
id_provider = ipa
auth_provider = ipa
access_provider = ipa
ipa_hostname = ipa2.example.com
chpass_provider = ipa
ipa_server = ipa2.example.com
ldap_tls_cacert = /etc/ipa/ca.crt
subdomains_provider = ipa
```

The trusted Active Directory domain is *not* explicitly defined in the SSSD configuration. The IdM domain is automatically created in the SSSD configuration when the client is installed; adding this line makes it possible to use the existing configuration.

- d. Save the changes to the `sss.conf` file.
- e. Restart SSSD.

```
[root@ipaserver ~]# systemctl restart sssd.service
```

15. Restart the `firewalld` services on the IdM server.

```
[root@ipaserver ~]# systemctl start firewalld.service
```

16. Restart the Windows firewall.

5.3.5. Adding a Second Trust

Much of the configuration for the first trust relates to general configuration for IdM to support trusts: installing packages, configuring SIDs for users, and enabling domain support in SSSD and Kerberos. After that, the process is much simpler, adding a trust requires updating the domain configuration for the new DNS domain and adding another trust agreement.

1. Stop the Windows firewall service.
2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Add the Active Directory server as a forwarder for IdM.

```
[root@ipaserver ]# ipa dnsconfig-mod --forwarder=255.255.255.255 --forward-policy=first
```

Using a first policy means that queries are sent to the forwarder first and then to the local **named** process. Alternatively, this can be set to only, so that only the DNS forwarder is queried, never **named**.

4. Set up the DNS realms. If IdM will be a subdomain of Active Directory, then follow step 5 in [Section 5.3.1, “Setting up Trust with IdM as a DNS Subdomain of Active Directory”](#). If IdM will host its own domain, then follow step 5 in [Section 5.3.2, “Setting up Trust with IdM and Active Directory in Different DNS Domains”](#).
5. Create a trust agreement for the Active Directory domain and the IdM domain.

```
[root@ipaserver ~]# ipa trust-add --type=ad adexample.com --admin
Administrator --password
Active directory domain administrator's password:
-----
Added Active Directory trust for realm "adexample.com"
-----
...
```

6. Restart the **firewalld** services on the IdM server.

```
[root@ipaserver ]# systemctl start firewalld.service
```

7. Restart the Windows firewall.

5.3.6. Adding a Trust in the Web UI

Much of the initial trust configuration is easiest to perform in the command-line, since it involved editing configuration files and installing packages. However, the actual operation to add the trust agreement can be done in the IdM UI; once much of the initial configuration is set, this can be an easy way to add additional trusts.



NOTE

This procedure shows how to add an additional trust. It is also possible to create an initial trust in the web UI, but those other steps have been excluded to focus on using the UI to add the trust configuration.

1. Stop the Windows firewall service.

2. Stop **firewalld** on the IdM server.

```
[root@ipaserver ]# systemctl stop firewalld.service
```

3. Set up the DNS realms. If IdM will be a subdomain of Active Directory, then follow step 5 in [Section 5.3.1, “Setting up Trust with IdM as a DNS Subdomain of Active Directory”](#). If IdM will host its own domain, then follow step 5 in [Section 5.3.2, “Setting up Trust with IdM and Active Directory in Different DNS Domains”](#).

4. Create a trust agreement for the Active Directory domain and the IdM domain.

- a. Open the IdM web UI.

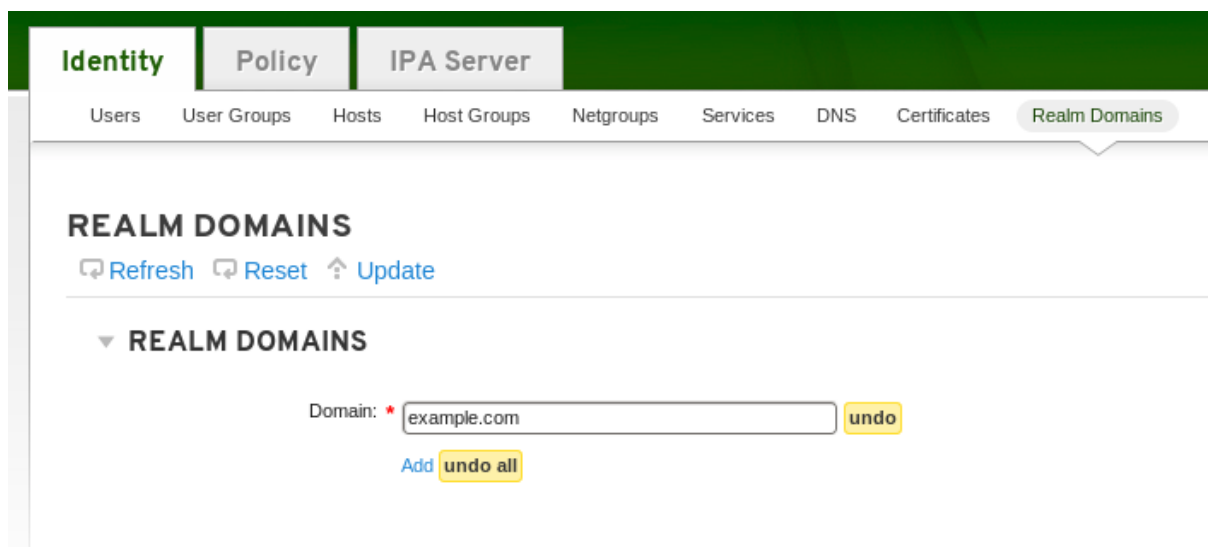
```
https://ipaserver.example.com
```

- b. Open the **IPA Server** main tab, and then select the **Trusts** subtab.

- c. In the **Trusts** subtab, click the **Add** link to open the new trust wizard.

- d. Fill in the required information about the trust. This includes the Active Directory domain name and the credentials (either the Active Directory administrator's username and password or a trusted password).

The base ID (the POSIX ID starting ID) and range are IdM-specific configuration. If these are not set manually, then they are autocalculated.



- e. Click the **Add** button to save the new trust.

5. Restart the **firewalld** services on the IdM server.

```
[root@ipaserver ]# systemctl start firewalld.service
```

6. Restart the Windows firewall.

5.4. 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.

As described in [Section 5.1.2, “Active Directory Principles, Security Objects, and Trust”](#), Active Directory users are added to the IdM domain in a kind of daisy chain. They are to an IdM external group (meaning, a non-POSIX group), and then that external group is added to a local POSIX group as a member. The IdM POSIX group can then be used for user/role management of Active Directory users.



TIP

It is also possible to add Active Directory user groups as members to IdM external groups. This may make it easier to define policies for Windows users, by keeping the user and group management within a single realm (Active Directory).

1. *Optional.* Create or select the group in the Active Directory domain to use to manage Active Directory 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. Using the **--external** argument indicates that this group will 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
```

3. Create the *POSIX* group for actually administering the IdM policies.

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

For example, for an Active Directory 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
-----
```

5.5. Maintaining Trusts

There are several layers to the trust configuration. There is the immediate trust agreement with IdM and its peer Active Directory. There is also a substantial backend configuration in IdM. When IdM is configured to support trusts, it creates a number of different kinds of configuration areas:

- ▶ Global trust configuration that is used to identify IdM within Windows domains (such as a SID)
- ▶ Identified DNS domains in Active Directory, which are pulled into the IdM DNS zone configuration (realm domains)
- ▶ The Kerberos trust configuration (the individual trust agreements, in trust domains)
- ▶ Assigned available ID ranges, per IdM server, to use to assign UID and GID numbers to Windows users as they enter the IdM domain

5.5.1. Editing the Global Trust Configuration

When the **ipa-adtrust-install** is run, it automatically configures background information for the IdM domain which is required to create a trust with the Active Directory domain. Even for external trusts, the Active Directory domain assumes that its trusted peer has certain configuration attributes, such as a security ID and domain ID. The attributes are created for the IdM server, so that it is Active Directory-compliant.

The global trust configuration contains five attributes:

- ▶ A Windows-style security ID
- ▶ A domain GUID
- ▶ A Kerberos domain name
- ▶ The default group to which to add Windows users

Only some of those attributes (the NetBIOS name and default group) can be edited. The GUID and SID are autogenerated, and the Kerberos realm name is from the IdM configuration.

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

5.5.1.1. Changing the NetBIOS Name

The NetBIOS name is the far-left component of the domain name; this is a key identifier for the host system of a domain controller. When IdM is enabled for trust, a NetBIOS name is set for the IdM server so that it is compatible within an Active Directory topology. This is configured in the **ipa-adtrust-install** command. To change it, rerun the **ipa-adtrust-install** command.

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

The **-a** option gives the IdM administrator password.

5.5.1.2. Changing the Default Group for Windows Users

IdM has a feature, *automembership*, which adds new users to specific groups automatically. There is a default automembership rule to add Windows users automatically to the *Default SMB Group* (a group created as part of the IdM trust configuration). This is a fallback group used if no other automembership rules apply to the Windows users.

The default group can be changed, which can be particularly useful if there are different external groups added for Windows users (Section 5.4, “Creating IdM Groups for Active Directory Users”). **This group is a fallback or default group which is used globally for all Windows users; other rules can be set to apply specific groups to different Windows users, rather than using the default.**

The default group can be set using the **trustconfig-mod** command:

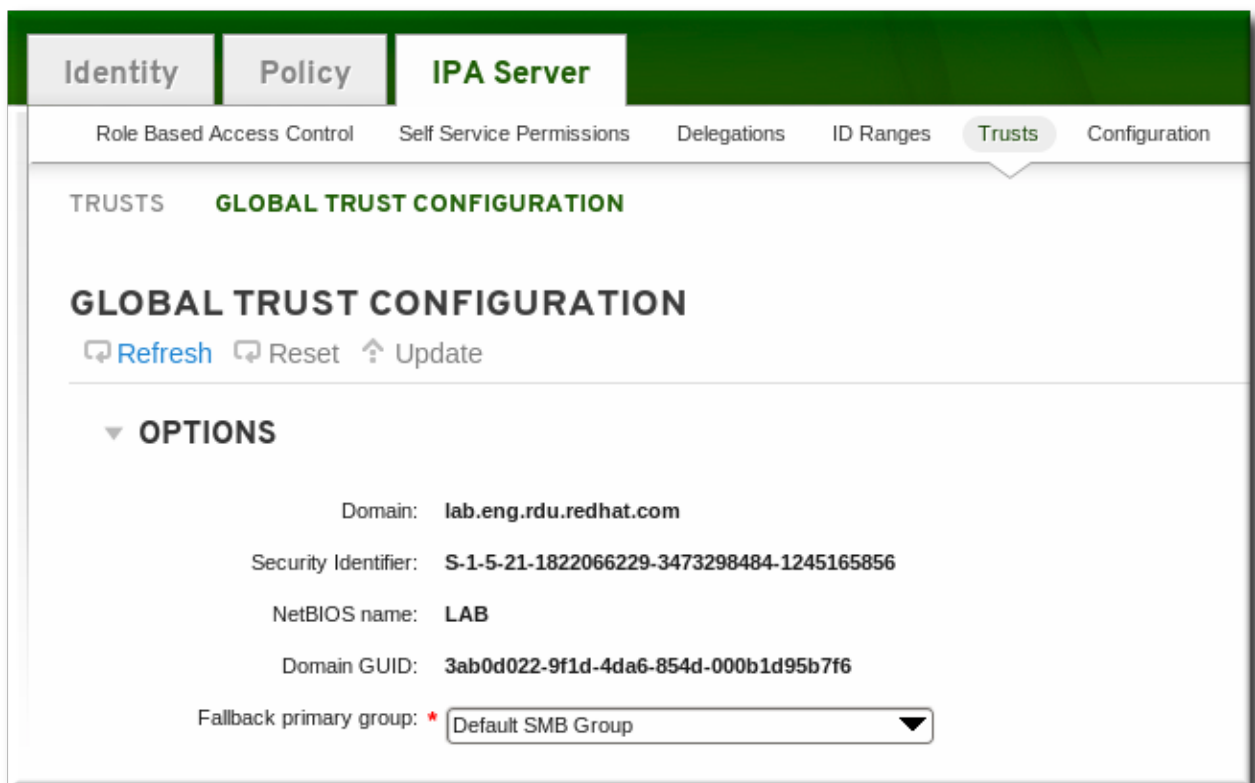
```
[root@server ~]# kinit admin
[root@server ~]# ipa trustconfig-mod --fallback-primary-group="Example Windows Group"
```

This can also be modified through the IdM web UI.

1. Open the IdM web UI.

<https://ipaserver.example.com>

2. Open the **IPA Server** main tab, and then select the **Trusts** subtab.
3. In the **Global Configuration** subtab, select a new group from all of the IdM groups listed in the **Fallback primary group**.



4. Click the **Update** link to save the new configuration.

5.5.2. Discovering, Enabling, and Disabling Trust Domains

A transitive trust means that the trust path can follow a chain of domains. If Domain A trusts Domain B, and Domain B trusts Domain C, then Domain A implicitly trusts Domain C. The trust it has with Domain B is transitive — it follows the trust path to Domain C.

IdM has a trust with the root domain in a forest, and all of its subdomains and trusted domains 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 subdomain 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 ([Section 5.3.3, “Setting up Trust with a Shared Secret”](#)) 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 adexample.com
-----
List of trust domains successfully refreshed
-----
  Realm name: test.adexample.com
  Domain NetBIOS name: TEST
  Domain Security Identifier: S-1-5-21-87535643-5658642561-5780864324

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

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

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

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

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

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

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

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

5.5.3. Viewing and Managing DNS Realms

When a trust is created, the Active Directory DNS configuration is added to the IdM DNS configuration, with each realm being added as a special *realm domain*. Each domain is stored in the **cn=Realm Domains,cn=ipa,cn=etc,dc=example,dc=com** subtree in the IdM directory.

Since these realm domains are added automatically, the DNS zones do not generally need to be added or modified. The list of configured realm domains can be displayed (instead of listing all DNS zones configured in IdM) using the **realmdomains-show** command.

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

If a single realm domain should be added to the configuration, this can be done with the **--add-domain** option.

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

A single domain can be removed using the **--del-domain** option.

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,adexample.com}
```

5.5.4. Adding Ranges for UID/GID Numbers in a Transitive Trust

Windows systems treat ID numbers differently than Linux systems. When a user is created in Linux, it is assigned a user ID number, and a private group is created for that user. The private group UI number is the same as the user ID number. In Linux, there is no conflict in that. On Windows, however, the security ID is unique for every object in the domain, so there is a conflict if a user and a group have the same ID.

If the POSIX attributes (including *uidNumber* and *gidNumber*) are pulled from the global catalog in Active Directory, then the ID numbers are unique, because they are unique within the Active Directory environment. This is an *ipa-ad-trust-posix* range type, which is set in the **trust-add** command when the trust is created. Essentially, no ID validation or range is required.

However, if instead the SIDs will be generated in IdM, using SID/username mapping, then the ID ranges for both the Windows identities and the IdM users and groups need to have unique, non-overlapping ranges available. This is *ipa-ad-trust* range type.

A unique ID range is created for each Active Directory domain automatically when it is added in a trust to IdM. However, Active Directory and IdM can work in a *transitive* trust. A transitive trust is a daisy change, where if Realm A trusts Realm B, and Realm B trusts Realms C, then Realm A also trusts Realm C. When the trust is configured, a range is only added for the domain specified in the trust agreement. Ranges for the transitively-trusted domains need to be added manually.

To add an ID range, set the base ID for the POSIX range (the starting number), the starting number of the RID (the far-right number in the SID), the size of the range, and the domain SID (since there can be multiple domains configured for trusts).

```
[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
```

The base ID is the starting POSIX ID number. The RID is a range to add to the base ID to prevent conflicts. If the base ID is 1200000 and the RID is 1000, then the resulting ID number is 1201000.

5.6. Verifying That IdM Machines Have Resolvable Names

As [Section 5.1.4, “Different DNS-Trust Environments”](#) explains, regardless of the DNS configuration, all hostnames within both the Identity Management and Active Directory DNS domains must be fully-resolvable for trusted services to function reliably.

After configuring trust, verify that the Identity Management servers are resolvable in both the IdM and Active Directory realms.

First, verify that the IdM-hosted services are resolvable from the IdM domain.

1. Run a DNS query for the Kerberos record over UDP.

```
[root@ipaserver ~]# dig +short -t SRV @10.1.1.1
_kerberos._udp.ipa.example.com.
0 100 88 ipamaster1.ipa.example.com.
```

2. Run a DNS query for the LDAP record over TCP.

```
[root@ipaserver ~]# dig +short -t SRV @10.1.1.1 _ldap._tcp.ipa.example.com.
0 100 389 ipamaster1.ipa.example.com.
```

3. Run a DNS query for the TXT record with the Kerberos realm name. This must match the Kerberos realm for the Identity Management servers.

```
[root@ipaserver ~]# dig +short -t TXT @10.1.1.1 _kerberos.ipa.example.com.
```

On the Active Directory server, verify that all of the IdM-hosted servers and services are resolvable.

Active Directory has a utility called **nslookup.exe** which can query the DNS configuration.

1. Set the **nslookup.exe** utility to look up service records.

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

2. Enter the name of the service and (optionally) the IP address of the IdM name server.

```
> _ldap._tcp.ipa.example.com 10.1.1.1
Server:    [10.1.1.1]
Address:   10.1.1.1

_ldap._tcp.ipa.example.com      SRV service location:
    priority                = 0
    weight                   = 100
    port                     = 389
    svr hostname             = ipaserver.ipa.example.com
ipaserver.ipa.example.com      internet address = 10.1.1.1
```

3. Change the service type to TXT to check the IdM Kerberos realm configuration [3]

```
> set type=TXT
```

4. Query the Kerberos records.

```
> _kerberos.ipa.example.com. 10.1.1.1
```

Active Directory caches the results of DNS lookups. The current cache can be viewed by running `ipconfig /displaydns`, and the cache can be deleted by running `ipconfig /flushdns`.

5.7. 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.

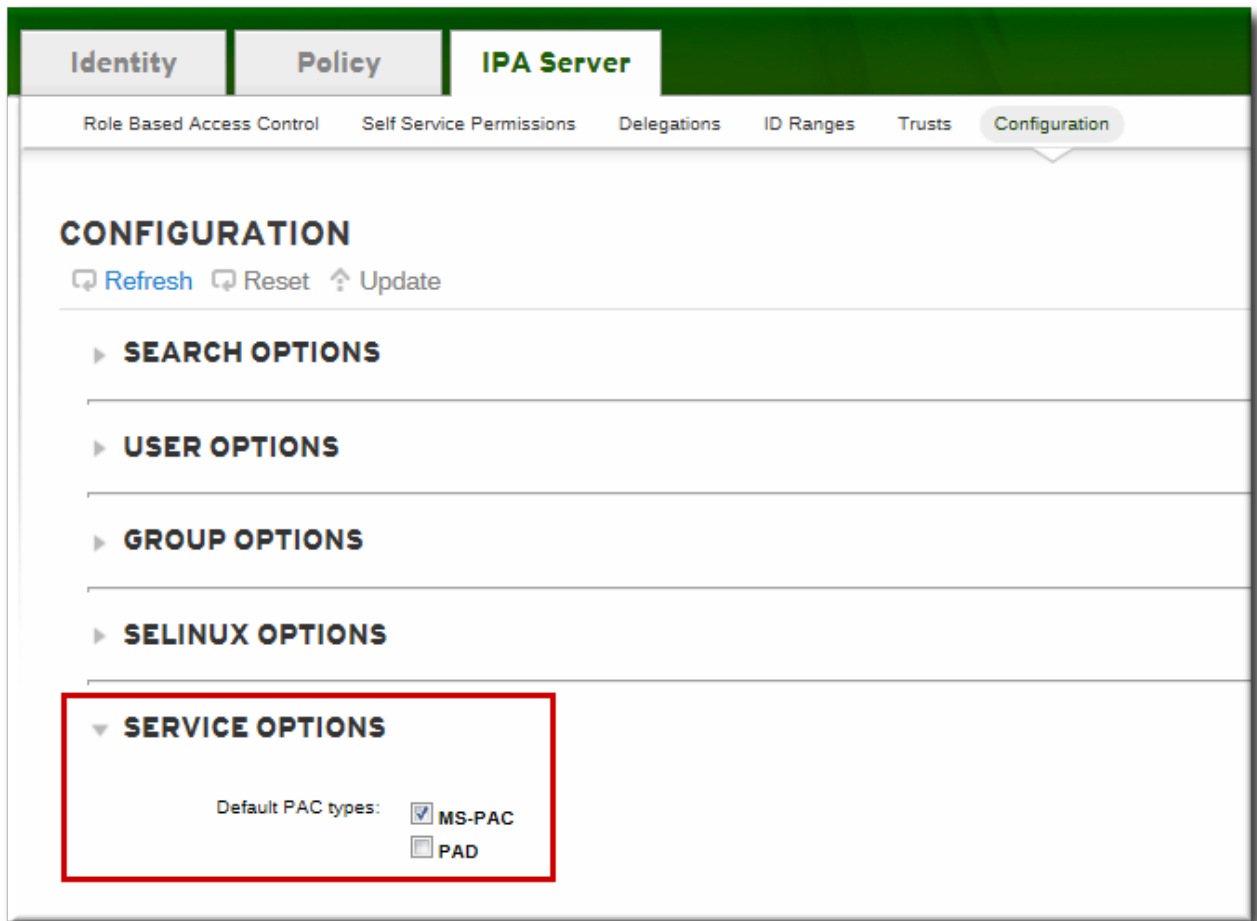
POSIX systems have a similar data set called a *POSIX authorization data* element. PADs, like PACs, contain group-based authorization data for a user. The access data is returned in response to the initial authentication request, so there is no additional cross-realm communication to retrieve group data.

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

5.7.1. 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.

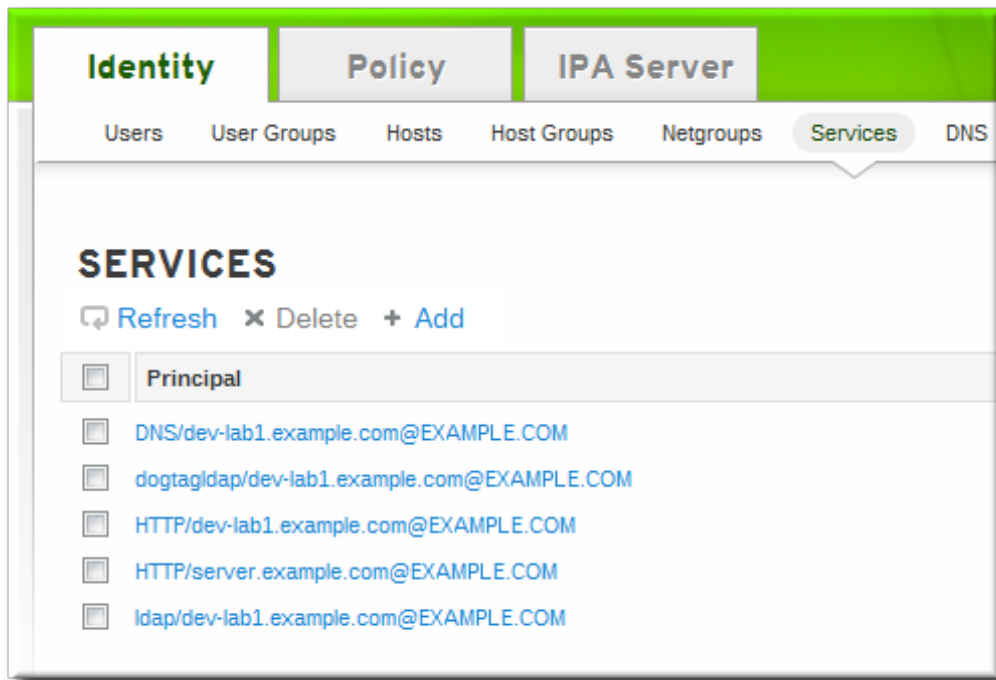


4. Select the check boxes by the PAC types to use. If both PAC types are selected, then both are added to the Kerberos ticket.
 - ▶ *MS-PAC* adds a certificate that can be used by Active Directory services.
 - ▶ *PAD* adds a certificate that can be used by POSIX (non-Windows) systems.
 - ▶ If no checkbox is selected, then no PACs are added to Kerberos tickets.
5. Click the **Update** link at the top of the page to save the changes.

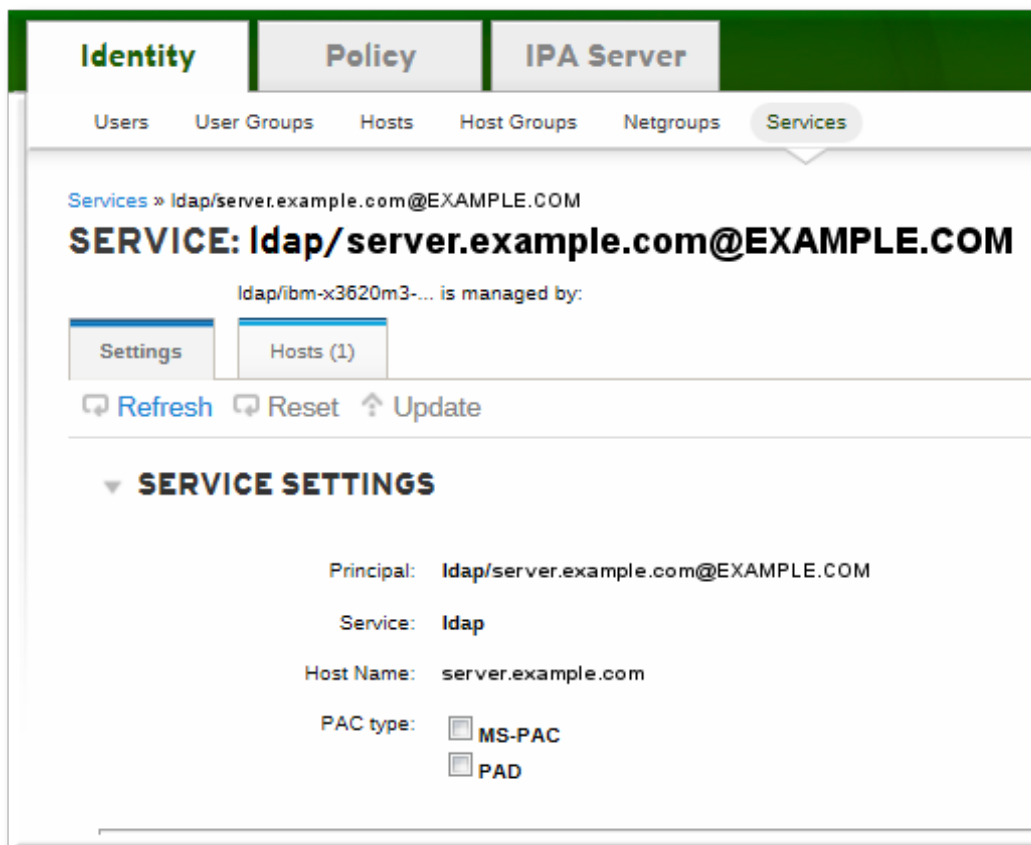
5.7.2. Setting PAC Types for a Service

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.

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, select the check boxes by the PAC types to use. If both PAC types are selected, then both are added to the Kerberos ticket.



- » *MS-PAC* adds a certificate that can be used by Active Directory services.
- » *PAD* adds a certificate that can be used by POSIX (non-Windows) systems.
- » If no checkbox is selected, then no PACs are added to Kerberos tickets.

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

5.8. 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 Active Directory credentials.



NOTE

When using PuTTY on the Windows machine, make sure that GSS-API credential delegation is enabled.

5.8.1. Username Requirements for SSH

One critical factor when using SSH is the username. The username must meet several criteria:

- ▶ The username must have the format *ad_user@ad_domain*.
- ▶ The domain name itself must be lower-case. This is required for Kerberos principal mapping.
- ▶ The case of the username must match, exactly, the case of the username in Active Directory. *jsmith* and *JSmith* are considered different users because of the different cases.

5.8.2. Using SSH Without Passwords

Even if a proper Kerberos ticket is obtained, using SSH still prompts for a user password for Active Directory domain users. SSH specifies the username with the **-l**, but the Kerberos ticket contains the Kerberos principal, not the username. The system requires a way to compare the offered, local username to a principal name to see if the user has a ticket. The **.k5login** file offers a simple way to map a local user to a Kerberos principal. The file is in a local user's home directory (and the user is identified by the **-l** option with SSH), and it lists Kerberos principals for that user. If the authenticating user matches the principal in an existing Kerberos ticket, then the user is allowed to log in using the ticket for authentication, rather than requiring a password.

To change to Kerberos authentication (meaning, to use passwordless SSH authentication), each Active Directory user should have a **.k5login** file in their Linux home directory. The only contents in the file are a list of Kerberos principals used by the user. The principals can be any of the formats in [Section 5.2.9, “Supported Username Formats”](#), such as *user@REALM.COM*, *AD.domain\user*, or *AD\user*.

For example, for the user *jsmith* in an Active Directory realm named *ENGINEERING.ADREALM.COM*, the **.k5login** is put in the home directory:

```
/home/engineering.adrealm/jsmith/.k5login
```

The contents of the file contain two different principal names:

```
jsmith@ENGINEERING.ADREALM
ENGINEERING.ADREALM.COM\jsmith
```

The **.k5login** file is case-sensitive, so listing multiple principals in different formats and different cases can be useful.

Because there are two different principals, either string can be used with **kinit** to obtain a ticket.

The **.k5login** manpage has more details.

5.9. Using Trust with Kerberized Web Applications

Any existing web application can be configured to use Kerberos authentication, which references the trusted Active Directory and IdM Kerberos realms. The full Kerberos configuration directives are covered in the [mod_auth_kerb](#) module man pages.

For example, for an Apache server, there are several parameters that define how the Apache server connects to the IdM Kerberos realm:

- ▶ The ***KrbAuthRealms*** directive gives the application location to the name of the IdM domain. This is required.
- ▶ The ***Krb5Keytab*** gives the location for the IdM server keytab. This is required.
- ▶ The ***KrbServiceName*** sets the Kerberos service name used for the keytab (HTTP). This is recommended.
- ▶ The Kerberos methods directives (***KrbMethodNegotiate*** and ***KrbMethodK5Passwd***) enables password-based authentication for valid users. This is recommended for ease of use for many users.
- ▶ The ***KrbLocalUserMapping*** directive enables normal web logins (which are usually the UID or common name of the account) to be mapped to the fully-qualified username (which has a format of *user@REALM.COM*).

This parameter 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.

[Section 5.2.9, “Supported Username Formats”](#) discusses different supported username formats.

Example 5.2. 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>
```

NOTE

After changing the Apache application configuration, restart the Apache service:

```
[root@ipaserver ~]# systemctl restart httpd.service
```

[3] There are usually no TXT records for Active Directory domains.

Chapter 6. Setting up Kerberos Cross-Realm Authentication

Kerberos v5 creates a realm of clients. That realm can be trust and integrate with an Active Directory domain, as well as other Kerberos domains. Kerberos itself is system-agnostic, so it can work in a number of different environments, systems, and applications.

A lot of Linux environments (and mixed environments) will already have a Kerberos realm deployed for single sign-on, application authentication, and user management. That makes Kerberos a potentially common integration path for mixed Windows-Linux environments, particularly if the Linux environment is not using a more structured domain configuration like Identity Management.

6.1. A Trust Relationship

A *trust* means that the users within one realm are trusted to access the resources in another domain as if they belonged to that realm. This is done by creating a shared key for a single principal that is held in common by both domains.

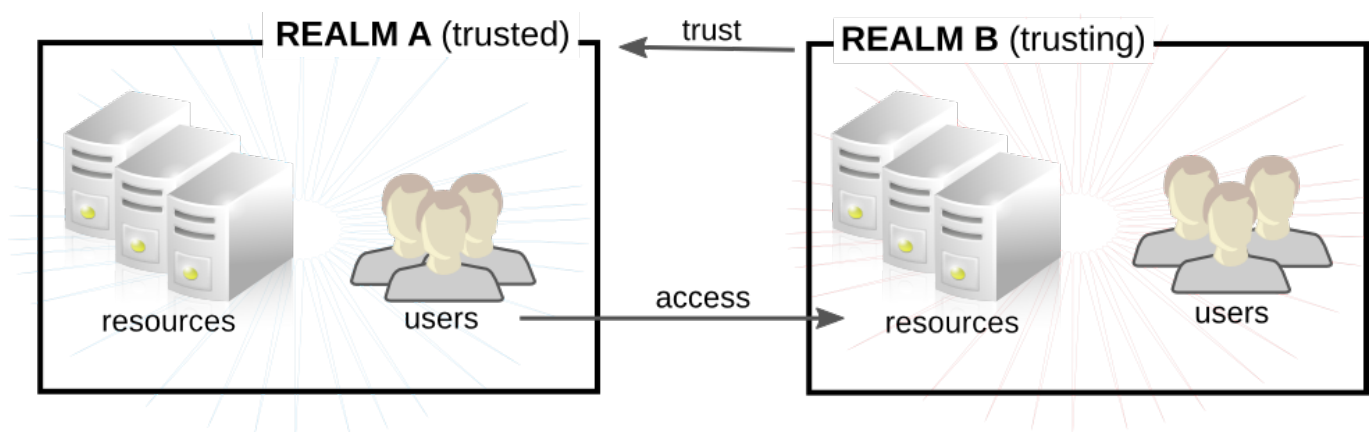


Figure 6.1. Basic Trust

In [Figure 6.1, “Basic Trust”](#), the shared principal would belong to Domain B (`krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM`). When that principal is also added to Domain A, then the clients in Domain A can access the resources in Domain B. The configured principal exists in both realms. That shared principal has three characteristics:

- ▶ It exists in both realms.
- ▶ When a key is created, the same password is used in both realms.
- ▶ The key has the same key version number (kvno).

A **cross-realm trust is unidirectional** by default. This trust is not automatically reciprocated so that the `B.EXAMPLE.COM` realm are trusted to authenticate to services in the `A.EXAMPLE.COM` realm. To establish trust in the other direction, both realms would need to share keys for the `krbtgt/A.EXAMPLE.COM@B.EXAMPLE.COM` service — an entry with a reverse mapping from the previous example.

A realm can have multiple trusts, both realms that it trusts and realms it is trusted by. With Kerberos trusts, the trust can flow in a chain. If Realm A trusts Realm B and Realm B trusts Realm C, Realm A implicitly trusts Realm C, as well. The trust flows along realms; this is a *transitive* trust.

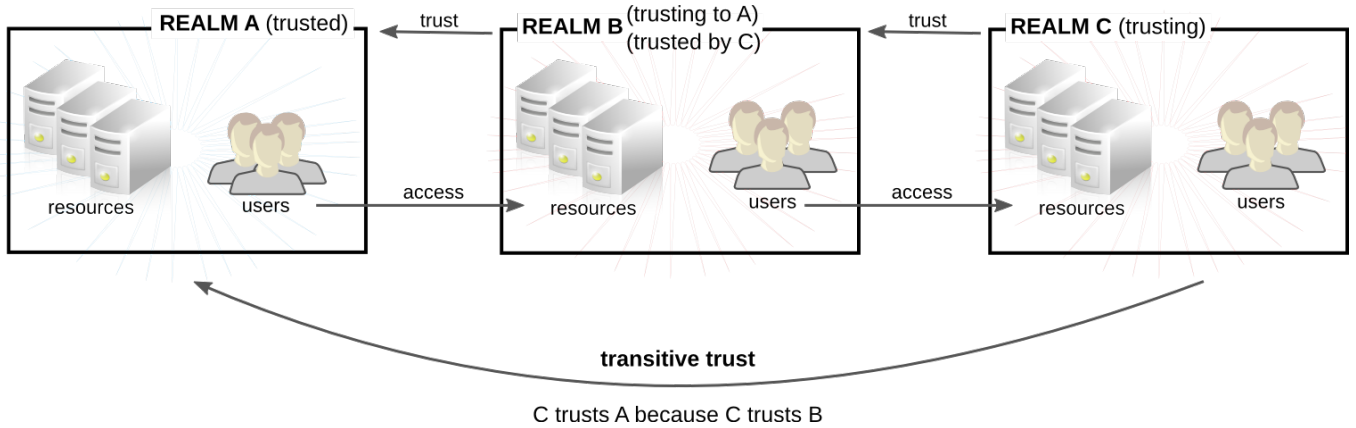


Figure 6.2. Transitive Trust



NOTE

While a Kerberos trust is transitive by default, that is not necessarily true for a Kerberos trust with a Windows domain. In Windows, trusts with other Windows domains is transitive, but trusts with external (meaning, non-Windows) realms are non-transitive by default. They can be configured to be transitive.

The direction of a transitive trust is the *trust flow*. The trust flow has to be defined, first by recognizing to what realm a service belongs and then by identifying what realms a client must contact to access that service.

A Kerberos principal name is structured in the format *service/hostname@REALM*. The *service* is generally a protocol, such as LDAP, IMAP, HTTP, or host. The *hostname* is the fully-qualified domain name of the host system, and the *REALM* is the Kerberos realm to which it belongs. Clients usually map the hostname or DNS domain name to the realm. The realm, then, somewhat related to the DNS domain name (unless a realm is explicitly defined in the `domain_realm` section of `/etc/krb5.conf`).

When traversing a trust, Kerberos assumes that each realm is structured like a hierarchical DNS domain, with a root domain and subdomains. This means that the trust flows up to a shared root. Each step, or *hop*, has a shared key. In [Figure 6.3, "Trusts in the Same Domain"](#), A shares a key with EXAMPLE.COM, and EXAMPLE.COM shares a key with B.

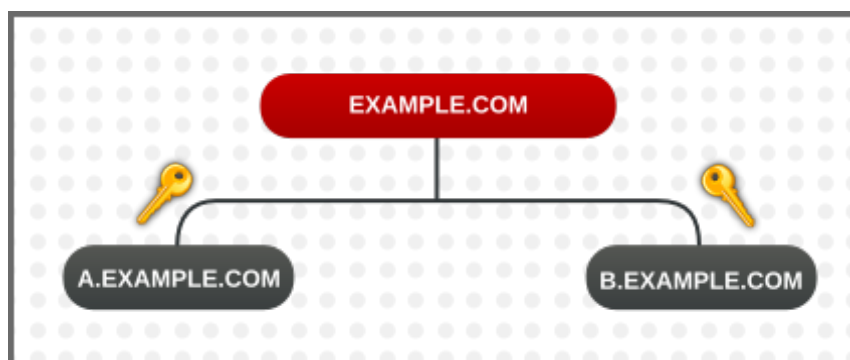


Figure 6.3. Trusts in the Same Domain

The client treats the realm name as a DNS name, and it determines its trust path by stripping off elements of its own realm name until it reaches the root name. It then begins prepending names until it reaches the service's realm.

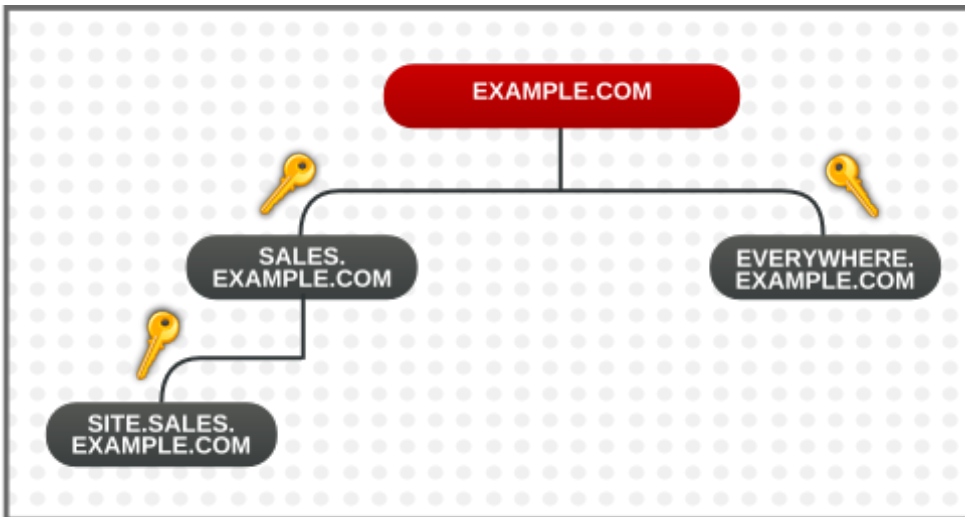


Figure 6.4. Child/Parent Trusts in the Same Domain

This is a nature of trusts being transitive. `SITE.SALES.EXAMPLE.COM` only has a single shared key, with `SALES.EXAMPLE.COM`. But because of a series of small trusts, there is a large trust flow that allows trust to go from `SITE.SALES.EXAMPLE.COM` to `EVERYWHERE.EXAMPLE.COM`.

That trust flow can even go between completely different domains by creating a shared key at the domain level, where the sites share no common suffix.

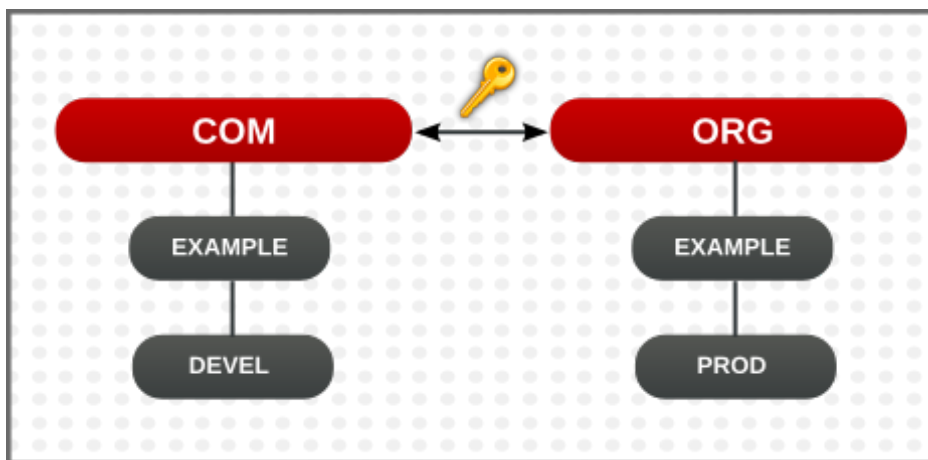


Figure 6.5. Trusts in Different Domains

It is also possible to reduce the number of hops and represent very complex trust flows by explicitly defining the flow. The `capaths` section of the `/etc/krb5.conf` file defines the trust flow between different realms.

The format of the `capaths` section is relatively straightforward: there is a main entry for each realm where a client has a principal, and then inside each realm section is a list of intermediate realms from which the client must obtain credentials.

For example, this has the realm of `A.EXAMPLE.COM`, and a set of hops from A to D. A client in Realm A must obtain credentials first from Realm B (the `.` means that it can obtain credentials directly, without any intermediate hops; otherwise, it would attempt to gain credentials by going through a hierarchy). It must then use the B credentials to obtain credentials from C, and then use the C credentials to obtain credentials for D.

```
[capaths]
A.EXAMPLE.COM = {
B.EXAMPLE.COM = .
C.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = C.EXAMPLE.COM
}
```

6.2. Setting up a Realm Trust

In this example, the Kerberos realm is **KRB.EXAMPLE.COM**, and the Active Directory realm is **AD.EXAMPLE.COM**.

1. Create the entry for the shared principal in Kerberos, using **kadmin**.

```
[root@server ~]# kadmin -r KRB.EXAMPLE.COM
kadmin: add_principal krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM
Enter password for principal "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM":
Re-enter password for principal "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM":
Principal "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM" created.
quit
```

2. A realm trust is configured in the **Active Directory Domains and Trusts** console. Select the appropriate domain, and create a new trust. These are the settings to use:

- ▶ The **Trust Type** is **Realm**.
- ▶ The **Transitivity of Trust** can be either transitive or non-transitive.
- ▶ The **Direction of Trust** is **One-way: incoming**. This trusts Active Directory users in the Kerberos realm.

This creates a unidirectional trust, where Active Directory users are trusted in the Kerberos realm. To create a two-way trust, set the direction of trust to two-way. This is described in the [Microsoft TechNet documentation](#).

- ▶ The **Sides of Trust** is **This domain only**.
- ▶ The **Trust Password** can be anything. This must be used when configuring the trust in Kerberos.

Chapter 7. Synchronizing Active Directory and Identity Management Users

Red Hat Enterprise Linux Identity Management uses active *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. This functionality is available immediately in Identity Management, with no additional configuration in the Active Directory domain.

Password synchronization is performed through a Windows service which is installed on the Windows server and then communicates to the Identity Management server.

7.1. Supported Windows Platforms

Synchronization are supported with these Windows servers:

- Windows Server 2008 R2 (32-bit)
- Windows Server 2008 R2 (64-bit)

The version of the password sync service which works with Windows is 1.1.5. This is available in the Red Hat Directory Server downloads part of Red Hat Network.

7.2. 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.

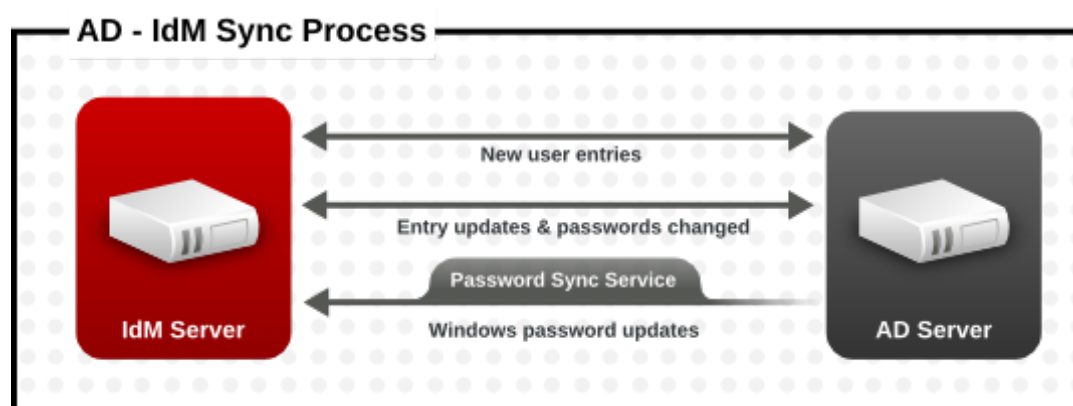


Figure 7.1. Active Directory and IdM Synchronization

Synchronization is defined in an *agreement* between an IdM server and an Active Directory domain controller. The sync agreement defines all of the information required to identify sync-able user entries (like the subtree to synchronize and requisite object classes in the user entries) as well as defining how account attributes are handled. The sync 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 7.1. Information in a Sync Agreement

Windows Information	IdM Information
<ul style="list-style-type: none"> ▶ User subtree (cn=Users, \$SUFFIX) ▶ Connection information <ul style="list-style-type: none"> ■ Active Directory administrator username and password ■ Password Sync Service password ■ CA certificate 	<ul style="list-style-type: none"> ▶ User subtree (ou=People, \$SUFFIX)

Synchronization is most commonly *bi-directional*. Information is sent back and forth between the IdM domain and the Windows domain in a process that is very similar to how IdM servers and replicas share information among themselves. It is possible to configure synchronization to only sync one way. That is *uni-directional* 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 synced 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.

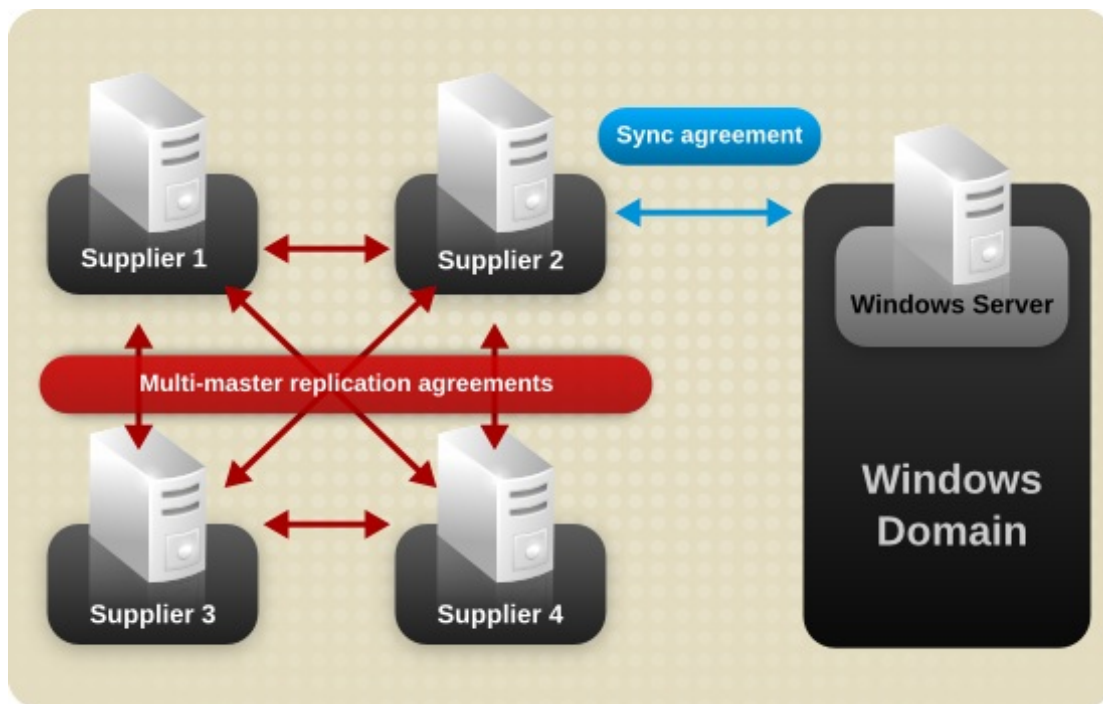


Figure 7.2. Synchronization Topology

There are some key features to IdM synchronization:

- ▶ A synchronization operation runs every five minutes.
- ▶ 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.
- ▶ Both user attributes and passwords can be synchronized.
- ▶ While modifications are bi-directional (going both from Active Directory to IdM and from IdM to Active Directory), creating accounts is only uni-directional, 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.
- ▶ 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 Sync 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 Sync client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

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 7.5.3, “Changing the Behavior for Syncing User Account Attributes”](#).

7.3. 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 ^[4]
- ▶ 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 7.2. 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
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 syncing from Identity Management to Active Directory. When syncing from Active Directory **cn** is mapped from the **name** attribute in Active Directory to the **cn** attribute in Identity Management.

7.3.1. User Schema Differences between Identity Management and Active Directory

Even though attributes may be successfully synced 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.

7.3.1.1. 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. If that **cn** change is written over to the Active Directory entry, then the entry is renamed, and the new named entry is written back over to Identity Management.

7.3.1.2. 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 synced 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.

7.3.1.3. 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.

7.3.1.4. 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 synced over to IdM since it fails with an object class violation.

7.3.2. Active Directory Entries and POSIX Attributes

Windows uses unique, random *security IDs (SIDs)* to identify users. These SIDs are assigned in blocks or ranges, identifying different system user types within the Windows domain. When users are synchronized between Identity Management and Active Directory, Windows SIDs for users are mapped to the Unix UIDs used by the Identity Management entry. Another way of saying this is that the Windows SID is the only ID within the Windows entry which is used as an identifier in the corresponding Unix entry, and then it is used in a mapping.

When Active Directory domains interact with Unix-style applications or domains, then the Active Directory domain may use Services for Unix or IdM for Unix to enable Unix-style **uidNumber** and **gidNumber** attributes. This allows Windows user entries to follow the specifications for those attributes in [RFC 2307](#).

However, the **uidNumber** and **gidNumber** attributes are not actually used as the **uidNumber** and **gidNumber** attributes for the Identity Management entry. The Identity Management **uidNumber** and **gidNumber** attributes are generated when the Windows user is synced over.



NOTE

The **uidNumber** and **gidNumber** attributes defined and used in Identity Management are not the same **uidNumber** and **gidNumber** attributes defined and used in the Active Directory entry, and the numbers are not related.

7.4. Setting up Active Directory for Synchronization

Synchronizing user accounts alone is enabled within IdM, so all that is necessary is to set up a sync agreement (Section 7.5.2, “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.

7.4.1. Creating an Active Directory User for Sync

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 sync user account **Replicating directory changes** rights to the synchronized Active Directory subtree. Replicator rights are required for the sync user to perform synchronization operations.

Replicator rights are described in <http://support.microsoft.com/kb/303972>.

- ▶ Add the sync user as a member of the **Account Operator** and **Enterprise Read-Only Domain controller** groups. It is not necessary for the user to belong to the full **Domain Admin** group.

7.4.2. 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](http://technet.microsoft.com/en-us/library/cc772393(v=WS.10).aspx).

7.5. Managing Synchronization Agreements

7.5.1. Trusting the Active Directory and IdM CA Certificates

Both Active Directory and Identity Management use certificates for server authentication. For the Active Directory and IdM SSL server certificates to be trusted by each other, both servers need to trust the CA certificate for the CA which issued those certificates. This means that the Active Directory CA certificate needs to be imported into the IdM database, and the IdM CA certificate needs to be imported into the Active Directory database.

1. On the Active Directory server, download the IdM server's CA certificate from `http://ipa.example.com/ipa/config/ca.crt`.
2. Install the IdM CA certificate in the Active Directory certificate database. This can be done using the Microsoft Management Console or the [certutil utility](#).

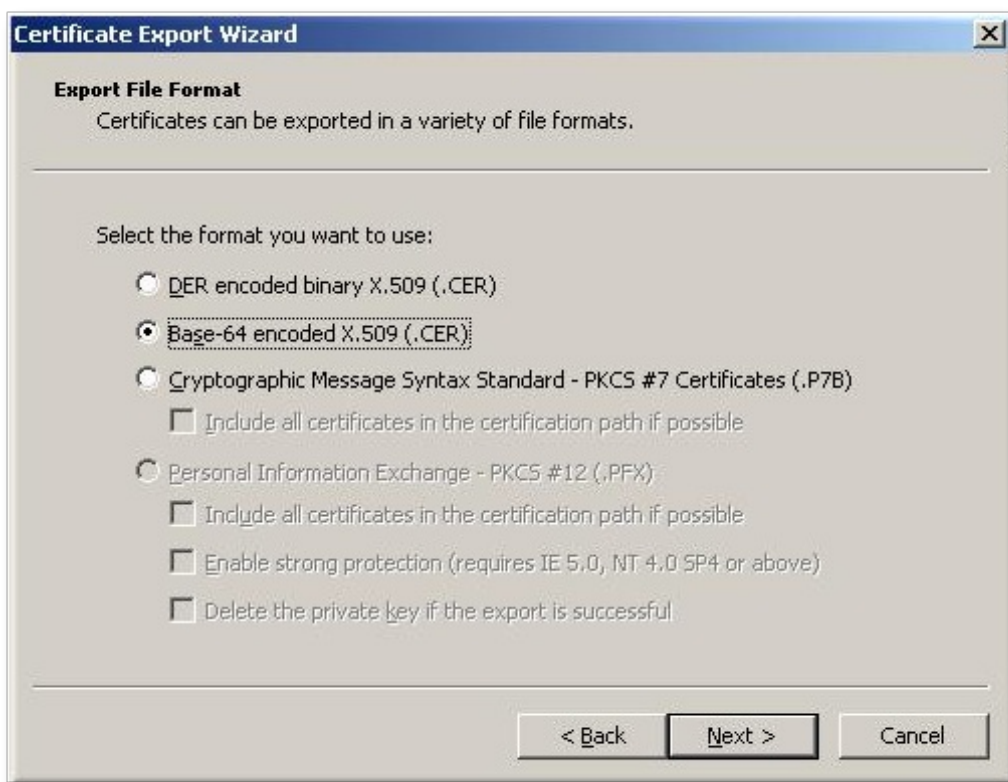
Right-click on the executable, and select **Run as administrator**, then run `certutil` with the `-installcert` option. For example:

```
C:\Windows\system32\certutil -installcert -v -config
"ipaserver.example.com\Example Domain CA" c:\path\to\ca.crt
```

This command must be run as an administrative account, or it will fail because it cannot access the certificate database.

For more details on installing certificates, see the Active Directory documentation.

3. Export the Active Directory CA certificate.
 - a. In **My Network Places**, open the CA distribution point.
 - b. Double-click the security certificate file (**.crt** file) to display the **Certificate** dialog box.
 - c. On the **Details** tab, click **Copy to File** to start the **Certificate Export Wizard**.
 - d. Click **Next**, and then select **Base-64 encoded X.509 (.CER)**.



- e. Specify a suitable directory and file name for the exported file. Click **Next** to export the certificate, and then click **Finish**.

4. Copy the Active Directory certificate over to the IdM server machine.
5. Download the IdM server's CA certificate from **<http://ipa.example.com/ipa/config/ca.crt>**.
6. Copy both the Active Directory CA certificate and the IdM CA certificate into the **/etc/openldap/cacerts/** directory.
7. Update the hash symlinks for the certificates.

```
cacertdir_rehash /etc/openldap/cacerts/
```


8. Edit the `/etc/openldap/ldap.conf` file, and add the information to point to and use the certificates in the `/etc/openldap/cacerts/` directory.

```
TLS_CACERTDIR /etc/openldap/cacerts/
TLS_REQCERT allow
```

7.5.2. 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. The options to create the synchronization agreement are listed in [Table 7.3, “Synchronization Agreement Options”](#).

1. Make sure that the Active Directory and IdM servers trust each other's CA certificates, as in [Section 7.5.1, “Trusting the Active Directory and IdM CA Certificates”](#).
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 Sync.

The `--binddn` and `--bindpwd` options give the username 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
```

4. When prompted, enter the Directory Manager password.
5. *Optional.* Configure Password Synchronization, as in [Section 7.6.2, “Setting up Password Synchronization”](#). Without the Password Sync client, user attributes are synchronized between the peer servers, but passwords are not.



NOTE

The Password Sync 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 Sync client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

Table 7.3. Synchronization Agreement Options

Option	Description
--winsync	Identifies this as a synchronization agreement.
--binddn	Gives the full user DN of the synchronization identity. This is the user DN that the IdM LDAP server uses to bind to Active Directory. This user must exist in the Active Directory domain and must have replicator, read, search, and write permissions on the Active Directory subtree.
--bindpw	Gives the password for the sync user.
--passsync	Gives the password for the Windows user account which is involved in synchronization.
--cacert	Gives the full path and file name of the Active Directory CA certificate. This certificate is exported in Section 7.5.1, “Trusting the Active Directory and IdM CA Certificates” .
--win-subtree	Gives the DN of the Windows subtree containing the users to synchronize. The default value is cn=Users,\$SUFFIX .
<i>AD_server_name</i>	Gives the hostname of the Active Directory domain controller.

7.5.3. Changing the Behavior for Syncing User Account Attributes

When the sync 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 list of attribute-related parameters are in [Table 7.4, “Synced Attribute Settings”](#).

The sync 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 sync 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"
```

Table 7.4. Synced Attribute Settings

Parameter	Description	Possible Values
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.	The default is (cn=ipaConfig) .

Parameter	Description	Possible Values
ipaWinSyncNewUserOCAAttr	Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.	The default is ipauserobjectclasses .
ipaWinSyncHomeDirAttr	Identifies which attribute in the entry contains the default location of the POSIX home directory.	The default is ipaHomesRootDir .
ipaWinSyncUserAttr	Sets an additional attribute with a specific value to add to Active Directory users when they are synced over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and the sync process adds all of the values to the entry.	ipaWinSyncUserAttr: <i>attributeName attributeValue</i>
<div style="border: 1px solid black; padding: 5px;">  <p>NOTE</p> <p>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 synced over.</p> </div>		
ipaWinSyncForceSync	Sets whether to check existing IdM users which match an existing Active Directory user should be automatically edited so they can be synchronized. 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 <i>not</i> synced by default. This attribute tells the sync service to add the ntUser and ntUserDomainId to the IdM user entries automatically, which allows them to be synchronized.	true false
User Account Lock Parameters		

Parameter	Description	Possible Values
<code>ipaWinSyncAcctDisable</code>	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 synced over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synced from both domains.	<ul style="list-style-type: none"> ▸ both (default) ▸ to_ad ▸ to_ds ▸ none
<code>ipaWinSyncInactivatedFilter</code>	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.	The default is (&(cn=inactivated)(objectclass=groupOfNames)) .
<code>ipaWinSyncActivatedFilter</code>	Sets the search filter to use to find the DN of the group used to hold active users. This does not need to be changed in most deployments.	The default is (&(cn=activated)(objectclass=groupOfNames)) .
Group Parameters		
<code>ipaWinSyncDefaultGroupAttr</code>	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.	The default is ipaDefaultPrimaryGroup .
<code>ipaWinSyncDefaultGroupFilter</code>	Sets the search filter to map the group name to the POSIX gidNumber .	The default is (&(gidNumber=*)(objectclass=posixGroup)(cn=groupAttr_value)) .
Realm Parameters		
<code>ipaWinSyncRealmAttr</code>	Sets the attribute which contains the realm name in the realm entry.	The default is cn .
<code>ipaWinSyncRealmFilter</code>	Sets the search filter to use to find the entry which contains the IdM realm name.	The default is (objectclass=krbRealmContainer) .

7.5.4. 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 sync 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 sync agreement entry.

1. Get the name of the sync 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=nsds7WindowsReplicationAgreement 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 sync agreement

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password -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 sync operation is currently running, then it takes effect as soon as the current operation completes.

7.5.5. Configuring Uni-Directional Sync

By default, all modifications and deletions are bi-directional. A change in Active Directory is synced over to Identity Management, and a change to an entry in Identity Management is synced 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 sync 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 sync agreement. The possible values are **fromWindows** (for Active Directory to Identity Management sync) and **toWindows** (for Identity Management to Active Directory sync).

For example, to sync 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=windows.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
```

```
tree,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```



IMPORTANT

Enabling uni-directional sync does *not* automatically prevent changes on the un-synchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync 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 than the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

7.5.6. Deleting Synchronization Agreements

Synchronization can be stopped by deleting the sync agreement which *disconnects* the IdM and Active Directory servers. In the inverse of creating a sync agreement, deleting a sync agreement uses the **ipa-replica-manage disconnect** command and then the hostname of the Active Directory server.

1. Delete the sync agreement.

```
# ipa-replica-manage disconnect adserver.example.com
```

2. Remove the Active Directory CA certificate from the IdM server database:

```
# certutil -D -d /etc/dirsrv/slapd-EXAMPLE.COM/ -n "Imported CA"
```

7.5.7. Winsync Agreement Failures

Creating the sync agreement fails because it cannot connect to the Active Directory server.

One of the most common sync 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                                 CTu,u,Cu
CA certificate                                       CT,,C
Imported CA                                         CT,,C
Server-Cert                                         u,u,u
Imported CA                                         CT,,C
```

To resolve this issue, clear the certificate database:

```
# certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"
```

This deletes the CA certificate from the LDAP database.

There are errors saying passwords are not being synced 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 Sync user, is not being changed. The Password Sync user is the operational user which is used by the service to change the passwords in IdM.

7.6. Managing Password Synchronization

Synchronizing user entries is configured with the sync 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 sync updates.



NOTE

The Password Sync 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 Sync client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

7.6.1. Setting up the Windows Server for Password Synchronization

Synchronizing passwords requires two things:

- ▶ Active Directory must be running in SSL.
- ▶ The Password Sync Service must be installed on *each* Active Directory domain controller.

The Password Sync Service records password changes and synchronizes them, over a secure connection, to the IdM entry.

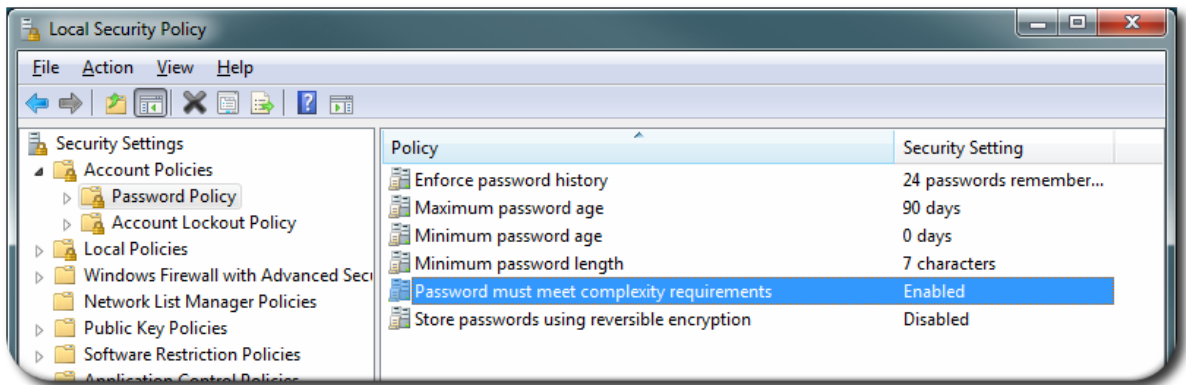


TIP

Install the Microsoft Certificate System in Enterprise Root Mode. Active Directory will then automatically enroll to retrieve its SSL server certificate.

1. Make sure that the Active Directory password complexity policies are enabled so that the Password Sync service will run.
 - a. Run **secpol.msc** from the command line.

- b. Select **Security Settings**.
- c. Open **Account Policies**, and then open **Password Policy**.
- d. Enable the **Password must meet complexity requirements** option and save.



2. If SSL is not already enabled, set up SSL on the Active Directory server. Setting up LDAPS is explained in more detail in the Microsoft knowledgebase at <http://support.microsoft.com/kb/321051>.
 - a. Install a certificate authority in the **Windows Components** section in **Add/Remove Programs**.
 - b. Select the **Enterprise Root CA** option.
 - c. Reboot the Active Directory server. If IIS web services are running, the CA certificate can be accessed by opening **http://servername/certsrv**.
 - d. Set up the Active Directory server to use the SSL server certificate.
 - a. Create a certificate request **.inf**, using the fully-qualified domain name of the Active Directory as the certificate subject. For example:

```

;----- request.inf -----

[Version]

Signature="$Windows NT$"

[NewRequest]

Subject = "CN=ad.server.example.com, O=Engineering, L=Raleigh,
S=North Carolina, C=US"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = False
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0

[EnhancedKeyUsageExtension]

```



```
OID=1.3.6.1.5.5.7.3.1
```

```
;-----
```

For more information on the `.inf` request file, see the Microsoft documentation, such as <http://technet.microsoft.com/en-us/library/cc783835.aspx>.

- b. Generate the certificate request.

```
certreq -new request.inf request.req
```

- c. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```



NOTE

If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is **http://servername/certsrv**.

- d. Accept the certificate request. For example:

```
certreq -accept certnew.cer
```

- e. Make sure that the server certificate is present on the Active Directory server.

In the **File** menu, click **Add/Remove**, then click **Certificates** and **Personal>Certificates**.

- f. Import the CA certificate from Directory Server into Active Directory. Click **Trusted Root CA**, then **Import**, and browse for the Directory Server CA certificate.

- e. Reboot the domain controller.

7.6.2. Setting up Password Synchronization

Install the Password Sync Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the **PassSync.msi** file to the Active Directory machine.
 - a. Log into the Customer Portal.
 - b. Click the **Downloads** tab.
 - c. Click the **Red Hat Enterprise Linux** downloads button in the middle of the page.
 - d. Filter the downloads by using a search term such as *Directory Server*, and then expand one of the Red Hat Enterprise Linux versions.
 - e. Click the Directory Server link.

- f. On the Directory Server page, download the appropriate version of the WinSync Installer. This is the Password Sync MSI file (**RedHat-PassSync-1.1.5-arch.msi**).



NOTE

Regardless of the Red Hat Enterprise Linux architecture, there are two PassSync packages available, one for 32-bit Windows servers and one for 64-bit. Make sure to select the appropriate packages for your Windows platform.

2. Double-click the Password Sync MSI file to install it.
3. The **Password Sync Setup** window appears. Hit **Next** to begin installing.
4. Fill in the information to establish the connection to the IdM server.
 - ▶ The IdM server connection information, including the hostname and secure port number.
 - ▶ The username of the system user which Active Directory uses to connect to the IdM machine. This account is configured automatically when sync is configured on the IdM server. The default account is **uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com**.
 - ▶ The password set in the **--passsync** option when the sync agreement was created.
 - ▶ 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**.
 - ▶ The certificate token is not used at this time, so that field should be left blank.

Red Hat Directory Password Sync Setup

Password Synchronization Information

Please enter your password synchronization information

Host Name:

Port Number:

User Name:

Password:

Cert Token:

Search Base:

< Back Next > Cancel

Hit **Next**, then **Finish** to install Password Sync.

5. Import the IdM server's CA certificate into the Active Directory certificate store.
 - a. Download the IdM server's CA certificate from **`http://ipa.example.com/ipa/config/ca.crt`**.
 - b. Copy the IdM CA certificate to the Active Directory server.
 - c. Install the IdM CA certificate in the Password Sync 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
```

```
cd "C:\Program Files\389 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 Sync.



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 Sync 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 Sync 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 Sync application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory sync peers. The tools to create the certificate and key databases is installed with the **.msi**.

7.6.3. Allowing Users to Change Other Users' Passwords Cleanly

By default, every time an administrator changes a user password, that user is required to reset the password at the next login. However, this behavior can be changed to allow administrators to reset a password *without* requiring an immediate password reset.

The ***passSyncManagersDNs*** attribute lists administrator accounts which are allowed to perform password change operations *and* which will not then require a password reset.

**IMPORTANT**

This is required for password synchronization because, otherwise, whenever a password is synchronized, the IdM server would interpret that as a password change operation and then require a password change at the next login.

Edit the password synchronization entry, **cn=ipa_pwd_extop,cn=plugins,cn=config**, and add the **passSyncManagersDNs** attribute with the name of the user. This attribute is multi-valued. For example:

```
$ ldapmodify -x -D "cn=Directory Manager" -w secret -h ldap.example.com -p 389
dn: cn=ipa_pwd_extop,cn=plugins,cn=config
changetype: modify
add: passSyncManagersDNs
passSyncManagersDNs: uid=admin,cn=users,cn=accounts,dc=example,dc=com
```

**WARNING**

Be careful to limit the listed DN's only to administrator accounts which require the ability to set user passwords. Any user listed here is given access to all user passwords, which is extremely powerful.

[4] The **cn** is treated differently than other synced attributes. It is mapped directly (**cn** to **cn**) when syncing from Identity Management to Active Directory. When syncing from Active Directory to Identity Management, however, **cn** is mapped from the **name** attribute on Windows to the **cn** attribute in Identity Management.

Index

A

Active Directory

- global catalog, [About Active Directory Identities on the Local System](#)
- schema differences between Identity Management, [User Schema Differences between Identity Management and Active Directory](#)

S

schema

- differences between Identity Management and Active Directory, [User Schema Differences between Identity Management and Active Directory](#)
 - cn, [Values for cn Attributes](#)
 - initials, [Constraints on the initials Attribute](#)
 - sn, [Requiring the surname \(sn\) Attribute](#)
 - street and streetAddress, [Values for street and streetAddress](#)

SSSD

- Active Directory
 - global catalog, [About Active Directory Identities on the Local System](#)

- Microsoft Active Directory domain, [Configuring an Active Directory Domain with ID Mapping](#),
[Configuring Active Directory as an LDAP Domain](#)