Red Hat Enterprise Linux 7
Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Aneta Šteflová Petrová Marc Muehlfeld Tomáš Čapek
Ella Deon Ballard

Red Hat Enterprise Linux 7 Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Aneta Šteflová Petrová
Red Hat Customer Content Services
apetrova@redhat.com

Marc Muehlfeld
Red Hat Customer Content Services
mmuehlfeld@redhat.com

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

Abstract

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

Table of Contents

Chapter 1. Ways to Integrate Active Directory and Linux Environments

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

1.1. Defining Windows Integration

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

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

User Identities and Authentication

- Where are user accounts located; in a central authentication system running on Windows (AD domain) or in a central identity and authentication server running on Linux?
- How are users authenticated on a Linux system; through a local Linux authentication system or a central authentication system running on Windows?
- How is group membership configured for users? How is that group membership determined?
- Will users authenticate using a user name/password pair, Kerberos tickets, certificates, or a combination of methods?
- POSIX attributes are required to access services on Linux machines. How are these attributes stored: are they set in the Windows domain, configured locally on the Linux system, or dynamically mapped (for UID/GID numbers and Windows SIDs)?
- What users will be accessing what resources? Will Windows-defined users access Linux resources? Will Linux-defined users access Windows resources?

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

Host and Service Principals

- What resources will be accessed?
- What authentication protocols are required?
- How will Kerberos tickets be obtained? How will SSL certificates be requested or verified?
- Will users need access to a single domain or to both Linux and Windows domains?

DNS Domains, Queries, and Name Resolution

- What will be a DNS configuration?
- Is there a single DNS domain? Are there subdomains?
- How will system host names be resolved?
- How will service discovery be configured?

Security Policies

- Where are access control instructions set?
- 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, for example the DNS service, how are those changes propagated?
- Is configuration maintained through domain-related tools or a provisioning system?
- Does the integration path require additional applications or configuration on the Windows server?

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

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

1.2. Direct Integration

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

Native LDAP and Kerberos PAM and NSS modules

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

Samba Winbind

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

The recent versions of the System Security Services Daemon (SSSD) closed a feature gap between Samba Winbind and SSSD and SSSD can now be used as a replacement for Winbind. In certain corner cases, Winbind might still be necessary to use but it is no longer the first choice in general.

System Security Services Daemon (SSSD)

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

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

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

1.3. Indirect Integration

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

Trust-based solution

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

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

Note

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

Synchronization-based solution

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

- duplication of users
- the need to synchronize passwords, which requires a special component on all domain controllers in an AD domain
- to be able to capture passwords, all users must first manually change them
- synchronization supports only a single domain
- only one domain controller in AD can be used to synchronize data to one instance of IdM or RHDS

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

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 back-end system. That can be a simple LDAP directory, domains for Active Directory (AD) or Identity Management (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. SSSD can also pull in group information. Authorization information is gathered by SSSD by using host-based access control (HBAC) in IdM and group policy object (GPO) settings in AD.

2.1. About SSSD

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

- *Reduced load on identification and authentication servers.* Rather than having every application service attempt to contact the identification server directly, each local application can contact SSSD, which in turn connects to the identification server or checks its cache.
- *Option for offline authentication.* SSSD keeps a cache of user identities (and optionally also user credentials) that it retrieves from remote services. This allows users to authenticate even if the remote identification server or the local machine is offline.
- *Single user account.* Users can have two or more user accounts. For example, one for their local system and another 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.

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.

Note

SSSD allows all user identities to be created and maintained in a separate, external identity source. For Windows integration, an AD domain is typically used to manage user accounts. Local system users do not need to be created or synced with user accounts in AD – SSSD uses those Windows identities and lets the 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 Service Switch (NSS) and Pluggable Authentication Modules (PAM), which are then used by higher-level applications.

Example 2.1. Simple sssd.conf File

```
[sssd]
domains = WIN.EXAMPLE.COM
services = nss, pam
config_file_version = 2

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

2.1.2. Active Directory Domain Configuration

As shown in Example 2.1, “Simple **sssd.conf** File”, the SSSD configuration file has two major sections: the first configures the SSSD service (**[sssd]**), the second configures configures the identity domains (**[domain/NAME]**). There might be additional sections that configure system services which use SSSD as an identity cache; for example **[nss]** or **[pam]**.

By default, only the identity provider (**id_provider**) and authorization provider (**access_provider**) options need to be configured. The **id_provider** option is used for the authentication (**auth_provider**) and password provider (**chpass_provider**) options if no other types or servers are set. Active Directory can be configured as any kind of provider using the **ad** value.

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

ad_server = dc1.example.com
# only needed if DNS discovery is not working
ad_hostname = client.example.com
# only needed if the host name of the client machine is incorrect
ad_domain = example.com
# only needed if AD domain is named differently than SSSD domain
```

The connection information is required to identify what Active Directory server to use. Past the basic configuration, the Active Directory identity provider can be configured specifically for the Active Directory environment and specific features, such as whether 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 is 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 shortcut 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

SSSD, for the most part, 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 tools which can prepare the initial configuration for the SSSD Active Directory domain; The **`realmd`** suite edits all underlying configuration files automatically. It simplifies editing the configuration but must be run separately on each system. 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. Active Directory Identities on the Local System

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

- ID mapping in SSSD can 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.
- Unix services can manage POSIX attributes on Windows user and group entries. This requires more configuration and information within the Active Directory environment, but it provides more administrative control over the specific UID/GID values and other POSIX attributes.

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. Performance-wise, the global catalog replication is the recommended way for SSSD to get information about users and groups, so that SSSD has access to all user data for all domains within the topology. As a result, SSSD can be used by applications which need to query the Active Directory global catalog for user or group information.

2.3.1.1. About Security ID Mapping

The Mechanism of ID Mapping

Linux/Unix systems use a local user ID number (UID) and group ID number (GID) to identify users on the system. These **`UID:GID`** numbers are a simple integer, for example **`501:501`**.

Microsoft Windows and Active Directory use a different user ID structure to identify users, groups, and machines. Each ID, also called a *Security Identifier* (SID) is constructed of different segments that identify the security version, the issuing authority type, the machine, and the identity itself. 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 is always assigned to SSSD. As this is a local range, it is the same for every machine. By default, this range is divided into 10,000 sections with each section allocated 200,000 IDs.

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 consistent way of assigning ID sections, so the same ID range is assigned to the same Active Directory domain on all client machines.

```
|    AD      |    AD      |          |
| domain 1   | domain 2   |    ...   |
|___________|___________|_________|
|  slice 1  |  slice 2  |    ...   |
min ID                        max ID
```

Note

As long as all clients use SSSD for the ID mapping, the mapping will be consistent. However, if some clients use different software, ensure that the same mapping algorithm is used or use explicit POSIX attributes.

ID Mapping Parameters

The ID mapping is enabled by default in the AD provider. The **`ldap_id_mapping`** parameter enables the mapping while the **`ldap_schema`** parameter configures which LDAP attribute is mapped to which SSSD attribute.

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.

Mapping Users

When an Active Directory user attempts to log into a local system 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 shell attribute is used according to SSSD settings.
- If the user belongs to any groups in the Active Directory domain, SSSD uses the SID to add 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. Once they are in the global catalog, they are available to SSSD and any application which uses SSSD for its identity information.

Replicating the attributes is beneficial for performance but is not required. SSSD tries to detect if POSIX attributes are present and if not, SSSD connects to the individual domain controllers directly on the LDAP port instead of requiring POSIX attributes to be replicated to the global catalog.

Note

Note that it is possible to use ID mapping even when POSIX attributes are defined on the server. In such a case, SSSD ignores the POSIX attributes.

To use existing POSIX attributes for the best performance, ensure the following:

- publish the POSIX attributes to Active Directory's global catalog,
- disable ID mapping in SSSD by setting **`ldap_id_mapping = False`** in the Active Directory domain entry.

2.3.1.3. Accessing a CIFS share with SSSD

SSSD is able to handle ID mapping between Windows security IDs (SIDs) and POSIX IDs. An SSSD client can therefore access and fully use a Common Internet File System (CIFS) share.

Note

In order to use a CIFS share for proper access control, it is necessary to translate the Windows SIDs to Linux POSIX UIDs and GIDs. Previously, only Winbind provided this functionality. Now, SSSD clients are no longer required to run Winbind alongside SSSD for this purpose.

The CIFS file-sharing protocol is widely deployed on Windows machines; SSSD enables seamless use of CIFS in environments with a trust between Identity Management and Active Directory as if it was a standard Linux file system. The SID-to-ID or SID-to-name algorithm that the SSSD client uses for system identity information can now also be used for a CIFS share. For example, accessing the Access Control Lists (ACLs) no longer requires to run Winbind in parallel to SSSD.

It is recommended to use SSSD for accessing a CIFS share instead of Winbind. IdM clients use SSSD by default to map AD users to UNIX users; using SSSD for the CIFS mapping avoids the possibility of inconsistent mapping, which can occur when IdM clients use Winbind. If a Linux client uses SSSD instead of Winbind for general AD user mappings in an environment with direct AD integration, where the client is directly joined into an AD domain, the client should also use SSSD as the mapping service for CIFS.

On the server side, SSSD also enables SID-to-POSIX ID mapping, providing access to a CIFS share to clients. However, Winbind on the server side still provides certain services that SSSD cannot, such as support for authentication using the NT LAN Manager (NTLM) or NetBIOS name lookup. This does not pose a problem for IdM clients because in IdM domains, Kerberos authentication and DNS name lookup are available for the same purposes.

Note

If you require NTLM authentication or NetBIOS name lookup, use Winbind for accessing a CIFS share instead of SSSD.

Configuring Samba for Accessing a CIFS Share with SSSD

To configure Samba to provide access to a CIFS share using SSSD, modify the **`[global]`** section of the **`/etc/samba/smb.conf`** file.

The following example of **`smb.conf`** is intended for environments with direct AD integration. The ***`system keytab`*** setting specifies that the keytab required for Kerberos access to the CIFS share is the same as the keytab that SSSD uses:

```
[global]
 security = ads
 workgroup = ADSHORTNAME
 realm = ADREALM
 kerberos method = system keytab
```

An example of **`smb.conf`** for environments with an AD trust, where the most widely used solution is to specify a dedicated keytab for Samba:

```
[global]
 security = ads
 workgroup = IPA
 realm = IPA.TEST
 dedicated keytab file = FILE:/etc/samba/samba.keytab
 kerberos method = dedicated keytab
```

Packages Required for Accessing a CIFS Share with SSSD

For a client to use SSSD to access a CIFS share, the following two packages are required.

sssd-client

The *sssd-client* package is installed automatically as an SSSD dependency. The package provides the SSSD plug-in for the *cifs-utils* package. This plug-in contains the **`libsss_nss_idmap.so`** library.

sssd-libwbclient

The *sssd-libwbclient* package is not installed automatically. To install it, run:

```
# yum install sssd-libwbclient
```

The package provides the **libwbclient.so.0.11-64** library, which is the SSSD alternative to the library provided by the *libwbclient* package used by the Winbind service.

After installing *sssd-libwbclient*, you can verify that your system uses the SSSD implementation with the **alternatives** tool. The tool displays the currently used alternative:

```
# alternatives --list | grep -E cifs\|libwbclient
cifs-idmap-plugin       auto     /usr/lib64/cifs-utils/cifs_idmap_sss.so
libwbclient.so.0.11-64 auto
/usr/lib64/sssd/modules/libwbclient.so.0.11.0
```

Switching Between SSSD and Winbind

To find out if you are currently using SSSD or Winbind for accessing a CIFS share, use the **alternatives** tool.

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

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

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

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

Important

It is recommended that IdM clients always use the SSSD plug-in.

If you want to switch to the Winbind plug-in, make sure that Winbind is running on the system. Similarly, if you want to switch to SSSD, make sure that SSSD is running.

2.3.2. Active Directory Users and Range Retrieval Searches

Microsoft Active Directory has an attribute, ***MaxValRange***, which sets a limit on how many values for a multi-valued attribute is returned. This is the *range retrieval* search extension. The extension runs multiple partial 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 1500 matching results or more, then ***MaxValRange*** limits how many are displayed at once. The given attribute has an additional flag set, showing which range in the set the result is in:

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

For example, to display results 100 to 500 in a search, use:

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

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

When the search base in SSSD specifies a custom filter or scope, some LDAP provider attributes which are available to configure searches (such as ***ldap_user_search_base***) cannot be used with range retrievals. When configuring search bases in the Active Directory provider domain, be aware what searches may trigger a range retrieval.

2.3.3. Linux Clients and Active Directory DNS Sites

SSSD connects a local Linux system to a larger Active Directory environment. This requires SSSD to 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.

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. SSSD sends an LDAP search to a domain controller 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.

3. 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.*):

   ```
   _tcp._ldap.site-name._sites.domain.name
   ```

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

   ```
   _tcp._ldap.domain.name
   ```

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

4. SSSD retrieves a list of primary and fallback 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 [1] and Active Directory-specific configuration [2]. This includes setting of 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 2.6, “Additional Configuration Examples”.

Note

Note that the following procedure covers the manual configuration of an Active Directory domain. By using **realmd**, steps 3 to 7 below can be done automatically by using the **realm join** command. See Chapter 3, *Using **realmd** to Connect to an Active Directory Domain* for details.

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. 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 example.com
   nameserver 192.0.2.1
   ```

3. 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. Install the following packages:

      ```
      [root@server ~]# yum install krb5-workstation samba-common-
      tools sssd-ad
      ```

   b. 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 auto-discovery is not used with SSSD, then also configure the **`[realms]`** and **`[domain_realm]`** sections to explicitly define the Active Directory server.

c. 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
   ```

d. 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`**.

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

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

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

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

5. 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
```

6. Open the SSSD configuration file.

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

7. 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
      auth_provider = ad
      chpass_provider = ad
      access_provider = ad
      ```

   c. 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
      ```

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

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

9. Restart SSSD after changing the configuration file.

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

2.5. Configuring an Active Directory Domain with POSIX Attributes

Warning

The *Identity Management for UNIX* extension used in the following section is now deprecated. As explained on the Microsoft Developer Network, an attempt to upgrade a system running Identity Management for UNIX might fail with a warning suggesting you to remove the extension. No replacement for the extension is currently available.

It is recommended to avoid using Identity Management for UNIX and instead set POSIX information on the IdM server using the *ID Views* mechanism, described in Chapter 8, *Using ID Views in Active Directory Environments*.

To use Active Directory-defined POSIX attributes in SSSD, it is recommended to replicate them to the global catalog for better performance. Once they are in the global catalog, they are available to SSSD and any application which uses SSSD for its identity information. Additionally, if the POSIX attributes are used, ID mapping has to 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 [3] and Active Directory-specific configuration [4]. This includes setting of 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 2.6, “Additional Configuration Examples”.

Note

Note that the following procedure covers the manual configuration of an Active Directory domain. By using **`realmd`**, steps 4 to 11 below can be done automatically by using the **`realm join`** command. See Chapter 3, *Using **`realmd`** to Connect to an Active Directory Domain* for details.

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. 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.

   c. For the relevant POSIX attributes (***`uidNumber`***, ***`gidNumber`***, ***`unixHomeDirectory`***, and ***`loginShell`***), open the **`Properties`** menu, select the **`Replicate this attribute to the Global Catalog`** check box, 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 example.com
nameserver 192.0.2.1
```

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. Install the following packages:

   ```
   [root@server ~]# yum install krb5-workstation samba-common-
   tools sssd-ad
   ```

   b. 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 auto-discovery is not used with SSSD, then also configure the **`[realms]`** and **`[domain_realm]`** sections to explicitly define the Active Directory server.

   c. 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
      ```

   d. 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. Start the SSSD service.

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

6. Open the SSSD configuration file.

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

7. 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.

   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
   auth_provider = ad
   chpass_provider = ad
   access_provider = ad
   ```

   c. 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
   ```

   d. 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 then 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
```

e. 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 user shortnames.
use_fully_qualified_names = True
```

f. 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
```

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

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

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

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

10. 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
```

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

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

12. 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
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. Additional Configuration Examples

2.6.1. Account Settings

With Linux users, certain system preferences are set by default for new users. 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 and default user shell.

2.6.1.1. 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 set the home directory attribute value 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/AD_EXAMPLE]
id_provider = ad
auth_provider = ad
...
override_homedir = /home/%d/%u
```

2.6.1.2. 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:

- Set a fallback value if no shells are supplied using ***shell_fallback***,
- Set lists of allowed or blacklisted shells using ***allowed_shells*** and ***vetoed_shells***,
- Set a default value using ***default_shell***,
- Set a value to use, even if another value is given in the identity provider, using ***override_shell***.

Note

The **allowed_shells**, **vetoed_shells**, and **shell_fallback** parameters 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/AD_EXAMPLE]
id_provider = ad
auth_provider = ad
...
default_shell = /bin/rbash
```

2.6.2. 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. Note that DNS scavenging is not enabled by default on the AD side.

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

Note

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
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 and must be set to **true** for any other dynamic DNS setting to be enabled. The default value is **true**.	Boolean
dyndns_ttl	Sets a time to live (TTL) for the client's DNS record. The default value 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 value 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 value is **true**.	Boolean

2.6.3. Using a Filter with Access Controls

The Active Directory access provider is used as the source for authorization information. The following configuration parameter option is actually a combination of several other generic LDAP parameters:

```
access_provider = ad
```

This is the same as setting the following LDAP parameters:

```
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, based on an LDAP filter. First, accounts must match the filter, and then they must pass the expiration check, which is implicit in the ***`access_provider = ad`*** setting. For example, the following 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=admins,ou=groups,dc=example,dc=com)
(unixHomeDirectory=*))
```

2.7. Group Policy Object Access Control

Group Policy is a Microsoft Windows feature that enables administrators to centrally manage policies for users and computers in AD environments. A *group policy object* (GPO) is a collection of policy settings, such as name and value pairs, that are stored on a domain controller (DC) and can be applied to policy targets, such as computers and users.

GPO policy settings related to Windows *logon rights* are commonly used to manage computer-based access control in AD environments. SSSD can retrieve GPOs applicable to host systems and AD users; based on the retrieved GPO configuration, it determines if a user is allowed to log on to a particular host. Therefore, with the GPO-based access control provided by SSSD, the administrator can define login policies that are honored by both Red Hat Enterprise Linux and Windows clients centrally on the AD DC.

Note

SSSD only allows using GPO for the computer-based access control. Other GPO-related access control options are currently not supported.

Warning

Note that SSSD only handles rules that apply to a whole site, domain, or AD organizational unit (OU). If you want to apply the SSSD-supported GPO-based access control to a specific machine, you can create a new OU in the AD domain, move the machine to the OU, and then link the GPO to this OU.

2.7.1. Configuring GPO-based Access Control

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

`ad_gpo_access_control = permissive`

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

`ad_gpo_access_control = enforcing`

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

`ad_gpo_access_control = disabled`

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

Important

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

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

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

  To move a service allowed by default to the deny list, remove it from the allow list. For example, to remove the **`su`** service from the allow list:

  ```
  ad_gpo_map_interactive = -su
  ```

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

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

[1] See the **`sssd-ldap`** man page.

[2] See the **`sssd-ad`** man page.

[3] See the **`sssd-ldap`** man page.

[4] See the **`sssd-ad`** man page.

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

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

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

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

3.1. Supported Domain Types and Clients

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

- Microsoft Active Directory
- Red Hat Enterprise Linux Identity Management

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

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

3.2. Prerequisites for Using `realmd`

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

```
# yum install realmd
```

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

Note

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

3.3. `realmd` Commands

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

- managing system enrollment in a domain
- setting which domain users are allowed to access the local system resources

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

```
realm command arguments
```

For example:

```
realm join ad.example.com
realm permit user_name
```

Table 3.1. realmd Commands

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

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

3.4. Discovering and Joining Identity Domains

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

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

1. Running a discovery scan for the specified domain.
2. Automatic installation of the packages required to join the system to the domain.

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

   Note

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

3. Joining the domain by creating an account entry for the system in the directory.
4. Creating the **/etc/krb5.keytab** host keytab file.
5. Configuring the domain in SSSD and restarting the service.
6. Enabling domain users for the system services in PAM configuration and the **/etc/nsswitch.conf** file.

Discovering Domains

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

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

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

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

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

Note

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

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

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

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

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

Joining a Domain

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Testing the System Configuration after Joining a Domain

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

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

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

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

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

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

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

4. Verify that the **`id`** utility prints the same information as the **`id`** ***`user@domain_name`*** command from the first step.

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

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

3.5. Removing a System from an Identity Domain

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

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

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

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

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

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

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

3.6. Listing Domains

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

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

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

`--all`

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

`--name-only`

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

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

3.7. Managing Login Permissions for Domain Users

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

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

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

`realm deny`

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

`realm permit`

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

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

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

- grant access to specified users, for example:

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

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

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

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

Important

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

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

3.8. Changing Default User Configuration

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

Important

Changing the configuration as described in this section only works if the **realm join** command has not been run yet. If a system is already joined, change the default home directory and shell in the **/etc/sssd/sssd.conf** file, as described in Section 2.6.1, “Account Settings”.

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

default-home

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

default-shell

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

For example:

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

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

3.9. Additional Configuration for the Active Directory Domain Entry

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

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

Important

Changing the configuration as described in this section only works if the **realm join** command has not been run yet. If a system is already joined, changing these settings does not have any effect. In such situations, you must leave the domain, as described in Section 3.5, “Removing a System from an Identity Domain”, and then join again, as described in Section 3.4, “Joining a Domain”. Note that joining requires the domain administrator's credentials.

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

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

Note that the same configuration can also be set when originally joining the system to the domain using the **realm join** command, described in Section 3.4, “Joining a Domain”:

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

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

Table 3.2. Realm Configuration Options

Option	Description

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

Chapter 4. Using Samba, Kerberos, and Winbind

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

Winbind is a component of the Samba suite to provide unified logon. It uses a UNIX implementation of Microsoft RPC calls, Pluggable Authentication Modules (PAMs), and the Name Service Switch (NSS) to allow Windows domain users to appear and operate as UNIX users on a UNIX system.

4.1. About Samba and Active Directory Authentication

While the core functionality of Samba is to perform client-server networking for file and printer sharing and associated operations, this chapter only focuses on one aspect of using Samba to interact with Windows: allowing Linux clients to authenticate by using Active Directory.

4.1.1. Samba, Kerberos, and Active Directory Domains

Active Directory is the domain controller for a number of services in Windows environments, including Kerberos realms and DNS domains. Samba supports the full range of protocols used in Active Directory, including Kerberos, DNS, NTLMSSP, or DCE/RPC. Integration with Active Directory means configuring a security environment that uses Kerberos as 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 name server
- Kerberos – to use the Active Directory KDC
- PAM – to use Winbind
- NSS – to use Winbind

4.1.1.1. Samba

There are several different ways how a Samba server can join an Active Directory domain. In SMB/CIFS networking, there are two types of security: user-level and share level. Samba provides four ways to use user-level security. Collectively, we call them the *security modes*. Only two of them are important for Windows integration:

- **`ads`** configures the local Samba server as a domain member within an Active Directory domain. It also enables support for the internal usage of LDAP queries and Kerberos authentication. This is the preferred security mode.
- **`domain`** configures the Samba server as a domain member server within an Active Directory domain by using the DCE/RPC protocol.

The necessary configuration is located in the **`[global]`** section of **`/etc/samba/smb.conf`**. The essential settings include the security type (***`security`***); the name of the Active Directory Kerberos realm (***`realm`***), which is resolved by DNS discovery; and the Samba workgroup (***`workgroup`***):

```
#================== Global Settings =====================

[global]
    workgroup = ADEXAMPLE
    security = ads
    realm = ADEXAMPLE.COM

...
```

4.1.1.2. Kerberos

Kerberos must be configured to use the Active Directory server as its KDC. It allows users to use Kerberos tickets for authentication. Additionally, Samba must be configured to use the Active Directory Kerberos realm, which 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.

For seamless Kerberos experience, ensure the Winbind Kerberos locator plug-in is installed from the *samba-winbind-krb5-locator* package. It ensures that Winbind and all its users and the Kerberos library and all its users use the same KDC all the time.

```
[libdefaults]
...

  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 host names and domains for Kerberos. While many systems have proper DNS settings so that the Samba-Active Directory integration could work well without configuring Active Directory as a name server, using Active Directory as a name server 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.

DNS settings are 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. For ease of use, offline caching of credentials, and other features, it is recommended to 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
```

Another important configuration file is **`/etc/security/pam_winbind.conf`**. In it, various parameters and defaults are set, including Kerberos authentication, offline authentication, or automatic home directory creation. For further details, see the pam_winbind.conf(5) man page.

For NSS, Active Directory can be used for passwords, shadow (users), and groups by setting Winbind as an option. Additionally, you can add the **`WINS`** service option to use the configuration also for hosts. Always use **`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.

NSS settings are configured in the **`/etc/nsswitch.conf`** file:

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

hosts:      files dns wins
```

Note

Note that PAM and NSS should not configured manually for integration with Active Directory. Instead, use the **`authconfig`** utility. See Section 4.3, "Configuring a Domain Member Using **`authconfig`**" for details.

4.1.2. Authentication Using Winbind and Samba

There are two important tasks when managing files: to establish the proper ownership and to control access to appropriate parties. Both relate to an effective way to identify and authenticate users. Winbind provides three related but separate capabilities:

- Authenticate users using local PAM configuration,
- Resolve IDs, user names, and groups using NSS lookups,
- Create a database of mapped Active Directory SIDs and local UID/GID numbers.

Winbind is part of Samba and connects directly to the Active Directory domain. The local Linux system is a full domain member in Windows terminology, represented with a complete machine account stored in AD. PAM and NSS are configured to use Winbind for user identities on the local system.

Among other aspects of using Winbind are:

- Winbind primarily maintains the machine account credentials (the Linux machine representation as a machine account in Active Directory). Among other functions, it can be used to update the machine account credentials or to update (or comply to) local stores of password policies.
- Winbind supports POSIX attributes in the form of RFC 2307 attributes or in the form of "Microsoft Services for Unix" extensions (both version 3.5 and 3.0). See the idmap_ad(8) man page for details.
- Joining the domain is done with utilities provided by Samba (using commands such as **`net ads join`**). Kerberos ticket management is done by Winbind, including ticket refresh and ticket re-acquisition.
- The **`smb.conf`** file is the only location for defining ID mappings.

Note

In Red Hat Enterprise Linux, it is recommended to use SSSD as a capable alternative for direct integration with Active Directory. See Chapter 2, *Using Active Directory as an Identity Provider for SSSD* for more information.

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`**	`[global]`
		`    workgroup = ADEXAMPLE`
		`    security = ads`
		`    realm = ADEXAMPLE.COM`
			samba

Winbind	**`/etc/security/pam_winbind.conf`**		samba-winbind

Kerberos	**`/etc/krb5.conf`**	`[libdefaults]`
		`  default_realm = ADEXAMPLE.COM`
		
		`[realms]`
		`  ADEXAMPLE.COM = {`
		`    kdc = kdc.adexample.com`
		`}`
		
		`[domain_realm]`
		` adexample.com = ADEXAMPLE.COM`
		`  .adexample.com = ADEXAMPLE.COM`
			krb5-workstation

PAM	**`/etc/pam.d/system-auth`** or **`/etc/pam.d/system-auth-ac`** (with authconfig)	**`auth`** `      sufficient`
		`pam_winbind.so use_first_pass`
		`account` `   [default=bad success=ok`
		`user_unknown=ignore] pam_winbind.so`
		`password` `  sufficient`
		`pam_winbind.so use_authtok`
		`session` `   optional`
		`pam_winbind.so use_first_pass`	

NSS	**`/etc/nsswitch.conf`**	`#required`
		`passwd:      files winbind`
		`shadow:      files winbind`
		`group:       files winbind`
		
		`#optional`
		`hosts:       files dns wins`	

DNS	**`/etc/resolv.conf`**	`nameserver` *`IPaddress`*
		`search` *`domainName`*	

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`** utility, with the exception of the DNS configuration. Configuration files can also be backed up by **`authconfig`**.

4.3.1. Arguments and Configuration Parameters of `authconfig`

The Authentication Configuration utility 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. Table 4.2, "authconfig Arguments and Configuration File Parameters" shows what parameters are set with each command option.

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

• Samba
• Kerberos
	`--smbrealm`	Winbind ADS Realm	• Samba
			  ◦ **`/etc/samba/smb.conf`**
			• Kerberos
			  ◦ **`/etc/krb5.conf`**
				• Samba
				  ◦ realm in [global]
				• Kerberos
				  ◦ default_realm in [libdefaults]
				  ◦ realm entry (*REALMNAME = {...}*) in [realms]

Kerberos	**`--smbservers`**	Winbind Domain Controllers	**`/etc/krb5.conf`**	The KDC in the realm entry (for example, *REALMNAME {...}*) in [realms]
Kerberos	**`--krb5realm`**		**`/etc/krb5.conf`**	The domain entry in [domain_realm]
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`**			

Important

The value of the **--krb5realm** option must be identical to the value given in **--smbrealm** for the domain to be configured properly.

4.3.2. CLI Configuration of Active Directory Authentication with `authconfig`

1. Install the *samba-winbind* package. It 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. It is required to connect to a Kerberos realm and manage principals and tickets:

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

3. Install the *samba-winbind-krb5-locator* package. It contains a plug-in for the system Kerberos library to allow the local Kerberos library to use the same KDC as Samba and Winbind use.

   ```
   [root@server ~]# yum install samba-winbind-krb5-locator
   ```

4. Edit the DNS configuration in the **/etc/resolv.conf** file to use the Active Directory domain as a name server and for search:

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

5. The **authconfig** utility 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.

   The following example shows all required parameters for Samba, Kerberos, PAM, and NSS. It also includes options for Winbind, which allow offline access, and for the local system, which allow system accounts to continue to work. The example command is split into multiple lines and annotated for better readability.

   ```
   [root@server ~]# authconfig
         // NSS
         --enablewinbind
         --enablewins
         // PAM
         --enablewinbindauth
         // Samba
         --smbsecurity ads
         --smbworkgroup=ADEXAMPLE
         --smbrealm ADEXAMPLE.COM
         // Kerberos
         --smbservers=ad.example.com
         --krb5realm=ADEXAMPLE.COM
         // winbind
         --enablewinbindoffline
   ```

```
        --enablewinbindkrb5
        --winbindtemplateshell=/bin/sh
        // general
        --winbindjoin=admin
        --update
        --enablelocauthorize
        --savebackup=/backups

[/usr/bin/net join -w ADEXAMPLE -S ad.example.com -U admin]
```

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

The **`--enablelocalauthorize`** option 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.

Note

The **`--savebackup`** option is recommended but not required. It backs up the configuration files to the specified directory before making the changes. If there is a configuration error or the configuration is later 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`** GUI than are in the CLI. For example, it is possible to configure Samba, NSS, 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 utilities are installed by default, but the GUI requires the *authconfig-gtk* package, which is not available by default.

1. Install the **`samba-winbind`** package. It 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. It 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]
...

  default_realm PLE.COM

[realms]
  ADEXAMPLE.COM
    kdc = kdc.adcom
}

[domain_realm]
 adexample.com =LE.COM
  .adexample.comMPLE.COM
```

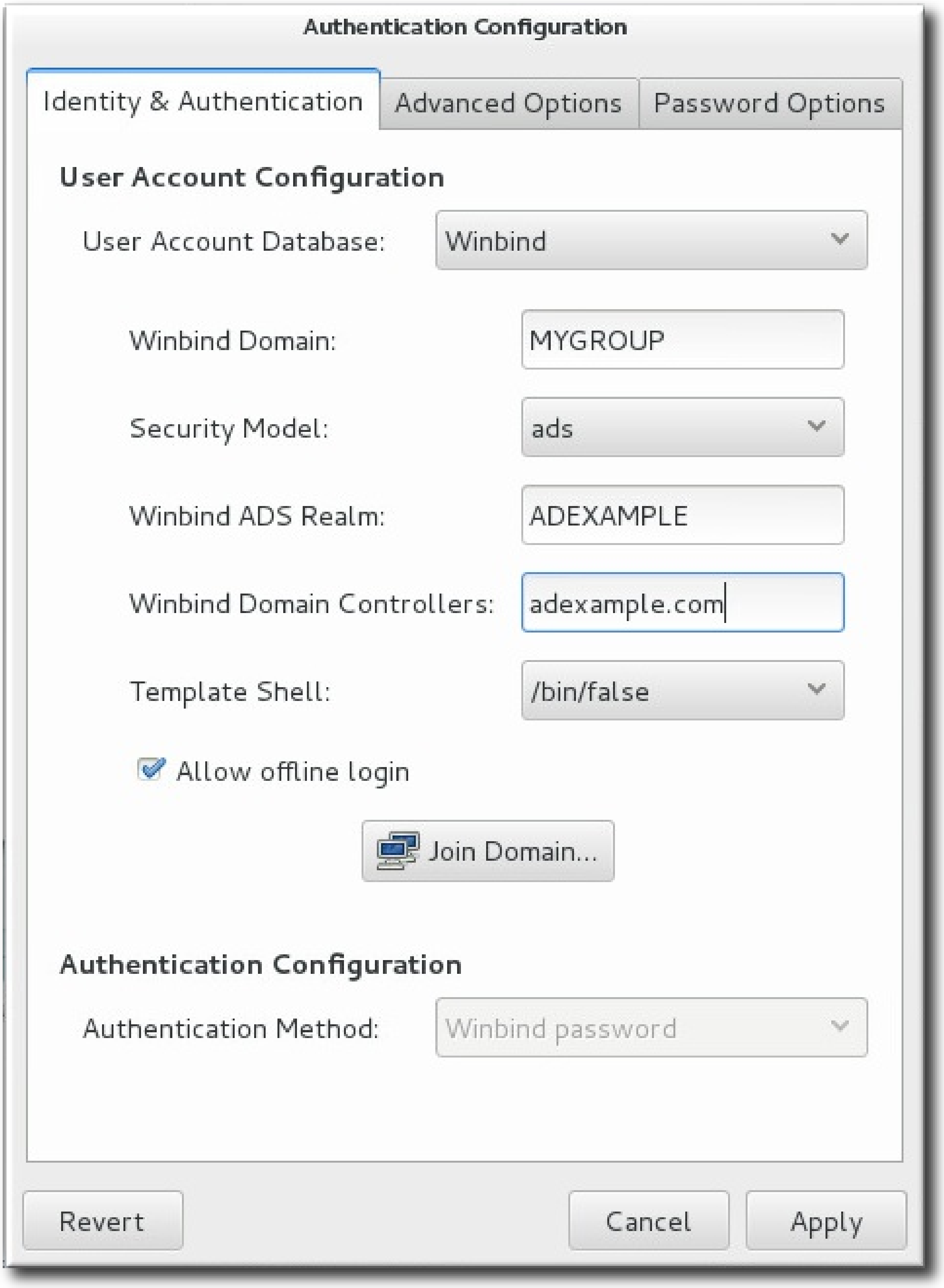
4. Edit the DNS configuration in the **`/etc/resolv.conf`** file to use the Active Directory domain as a name server and for search:

   ```
   nameserver 1.2.3
   search adexample
   ```

5. Open the Authentication Configuration Tool.

   ```
   [root@server ~]# authconfig-gtk
   ```

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



User Account Configuration

User Account Database: Winbind

Winbind Domain: MYGROUP

Security Model: ads

Winbind ADS Realm: ADEXAMPLE

Winbind Domain Controllers: adexample.com

Template Shell: /bin/false

✓ Allow offline login

Authentication Configuration

Authentication Method: Winbind password

7. Set the information that is required to connect to the Microsoft Active Directory domain controller.

   - **`Winbind Domain`** gives the Windows work group. The entry in this field needs to be in the Windows 2000 format, such as **`DOMAIN`**.

- **`Security Model`** sets the security model to use for Samba clients. The correct value is **`ads`** that configures 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 host name or IP address of the domain controller to use.
- **`Template Shell`** sets which login shell to use for Windows user account settings. This setting 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`** button to run the **`net ads join`** command and join the Active Directory domain. This action is 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.

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

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

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

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

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

5.1. Introduction to Cross-forest Trusts

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

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

5.1.1. The Architecture of a Trust Relationship

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

Active Directory Trusts, Forests, and Cross-forest Trusts

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

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

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

Trust Flow and One-way Trusts

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

The *trusted domain* contains users, and the *trusting domain* allows access to resources. In a one-way trust, trust flows only in one direction: users can access the trusting domain's resources but users in the trusting domain cannot access resources in the trusted domain. In Figure 5.1, “One-way Trust”, Domain A is trusted by Domain B, but Domain B is not trusted by Domain A.

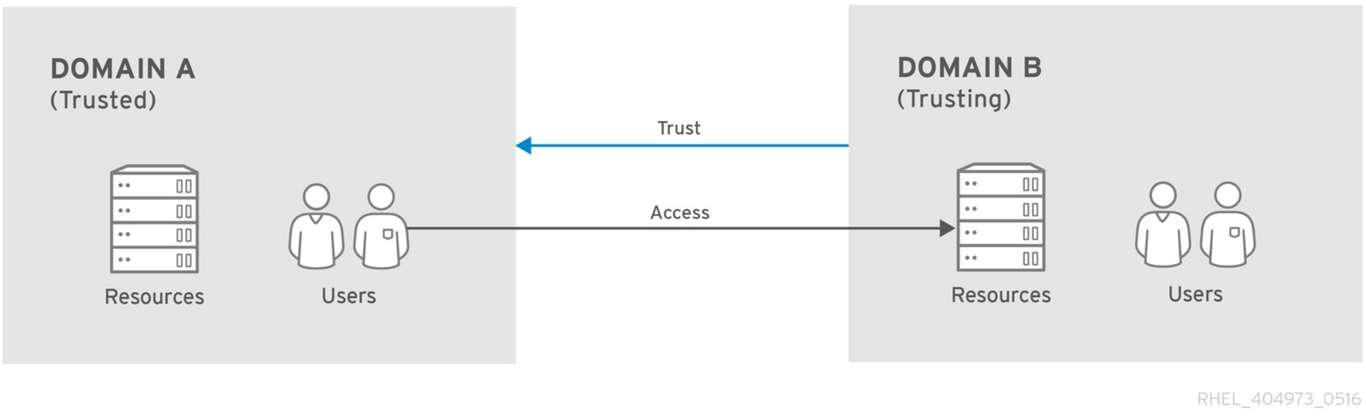


Figure 5.1. One-way Trust

IdM allows the administrator to configure both one-way and two-way trusts. For more information on how to achieve this, see Section 5.3.1, “One-Way and Two-Way Trusts”.

Transitive and Non-transitive Trusts

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

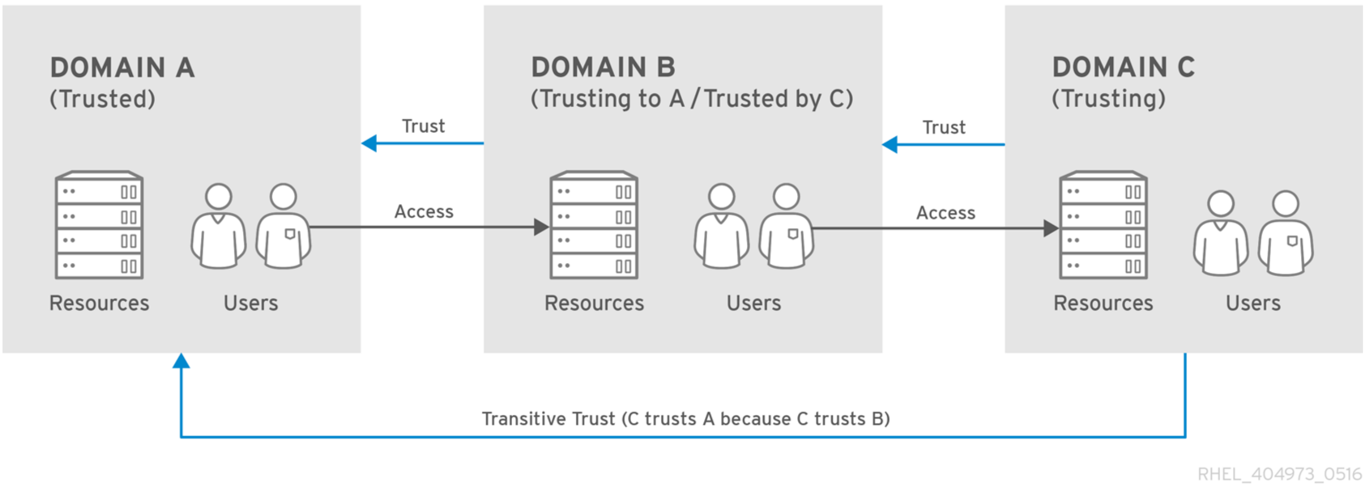


Figure 5.2. Transitive Trusts

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

Cross-forest Trust in Active Directory and Identity Management

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

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

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

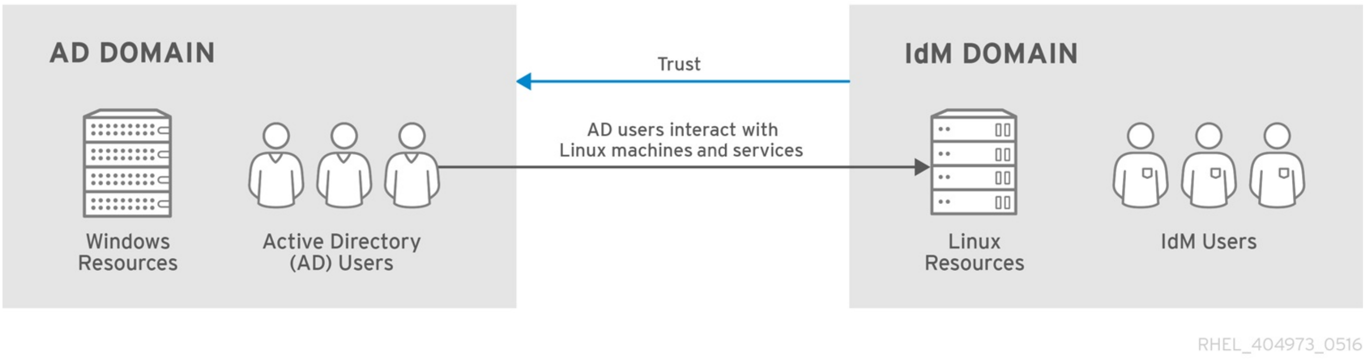


Figure 5.3. Trust Direction

5.1.2. Active Directory Security Objects and Trust

Active Directory Global Catalog

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

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

Global Catalog and POSIX Attributes

Active Directory does not replicate POSIX attributes with its default settings. If it is required to use POSIX attributes that are defined in AD Red Hat strongly recommends to replicate them to the global catalog service, as mentioned in Section 2.3.1.2, “About SSSD and POSIX Attributes”.

5.1.3. Trust Architecture in IdM

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

- SSSD, to perform identity lookups on Active Directory and to retrieve user and group security identifiers (SIDs) for authorization. SSSD also caches user, group, and ticket information for users and maps Kerberos and DNS domains,
- Identity Management (Linux domain management), to associate the Active Directory user with an IdM group for IdM policies and access.

Note

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

Trusts with Different Active Directory Forests

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

5.1.3.1. Active Directory PACs and IdM Tickets

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

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

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

1. The request for a service contains the PAC of the user. The IdM KDC analyzes the PAC by comparing the list of Active Directory groups to memberships in IdM groups.
2. For SIDs of the Kerberos principal defined in the MS-PAC, the IdM KDC evaluates external group memberships defined in the IdM LDAP. If additional mappings are available for an SID, the MS-PAC record is extended with other SIDs of the IdM groups to which the SID belongs. The resulting MS-PAC is signed by the IdM KDC.
3. The service ticket is returned to the user with the updated PAC signed by the IdM KDC. Users belonging to AD groups known to the IdM domain can now be recognized by SSSD running on the IdM clients based on the MS-PAC content of the service ticket. This allows to reduce identity traffic to discover group memberships by the IdM clients.

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

1. The Kerberos client libraries used in the evaluation process send the PAC data to the SSSD PAC responder.
2. The PAC responder verifies the group SIDs in the PAC and adds the user to the corresponding groups in the SSSD cache. SSSD stores multiple TGTs and tickets for each user as new services are accessed.
3. Users belonging to the verified groups can now access the required services on the IdM side.

5.1.3.2. Active Directory Users and Identity Management Groups

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

For a description of how to configure IdM groups for AD users, see Section 5.5, “Creating IdM Groups for Active Directory Users”.

Non-POSIX External Groups and SID Mapping

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

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

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

ID Ranges

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

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

Note

AD stores a subset of information for all objects within the forest in a *global catalog*. The global catalog includes every entry for every domain in the forest. If you want to use AD-defined POSIX attributes, Red Hat strongly recommends that you first replicate the attributes to the global catalog, as mentioned in Section 2.3.1.2, “About SSSD and POSIX Attributes”.

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

ipa-ad-trust

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

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

ipa-ad-trust-posix

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

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

For example:

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

5.1.3.3. Active Directory Users and IdM Policies and Configuration

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

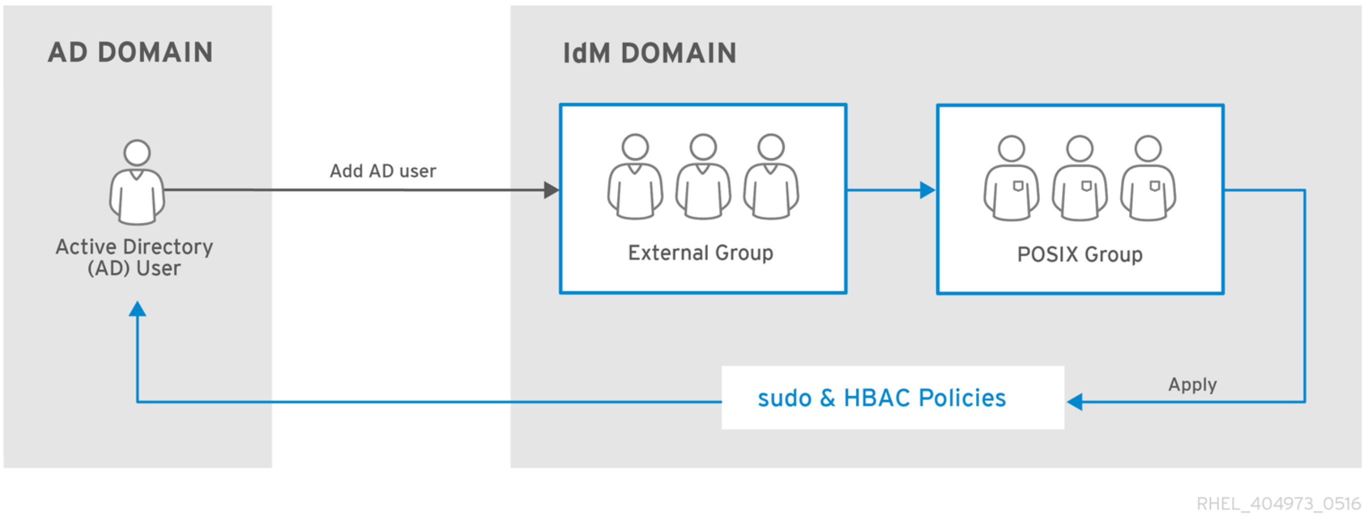


Figure 5.4. Active Directory Users and IdM Groups and Policies

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

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

5.1.4. User Principal Names in a Trusted Domains Environment

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

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

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

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

Verify that the alternative UPN was fetched, by running:

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

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

5.1.5. Potential Behavior Issues with Active Directory Trust

5.1.5.1. Active Directory Users and IdM Administration

Active Directory users exist only within the AD domain. Because they do not exist within IdM, they cannot be administrators for IdM. They also cannot use any IdM administrative tools, including the IdM web UI and command-line utilities.

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

5.1.5.2. Authenticating Deleted Active Directory Users

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

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

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

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

5.1.5.3. Credential Cache Collections and Selecting Active Directory Principals

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

1. service name
2. host name
3. realm name

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

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

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

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

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

This is set as the default principal in the Active Directory ticket cache. However, if any IdM user also has a Kerberos ticket (such as **admin**), then there is a separate IdM credentials cache, with an IdM default principal. That IdM default principal is selected for a host ticket if the Active Directory user uses SSH to connect to a resource.

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

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

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

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

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

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

5.1.5.4. Resolving Group SIDs

Losing Kerberos Tickets

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

Note

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

Cannot Verify Group Membership for Users

It is not possible to verify that a specific trusted user is associated with a specific IdM group, external or POSIX.

Cannot Display Remote Active Directory Group Memberships for an Active Directory User

Important

Note that this problem no longer occurs if the IdM server and client run on Red Hat Enterprise Linux 7.1 or later.

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

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

```
[root@ipaserver ~]# id ADDOMAIN\jsmith
```

```
uid=1921801107(jsmith@ad.example.com)
gid=1921801107(jsmith@ad.example.com)
groups=1921801107(jsmith@ad.example.com),129600004(ad_users),1921800513(do
main users@ad.example.com)
```

5.2. Environment and Machine Requirements to Set up Trusts

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

5.2.1. Supported Windows Platforms

Trust relationships can be configured with these Windows server versions:

- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2

5.2.2. DNS and Realm Settings

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

Unique primary DNS domains

Each system must have its own unique primary DNS domain configured; for example, **`ad.example.com`** for AD and **`idm.example.com`** for IdM. The most convenient management solution is an environment where each DNS domain is managed by integrated DNS servers, but it is possible to use any other standard-compliant DNS server as well.

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

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

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

DNS records resolvable from all DNS domains in the trust

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

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

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

No overlap between IdM and AD DNS Domains

Machines joined to IdM can be distributed over multiple DNS domains. DNS domains containing IdM clients must not overlap with DNS domains containing machines joined to AD. The primary IdM DNS domain must have proper SRV records to support AD trusts.

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

Verifying the DNS Configuration

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

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

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

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

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

   ```
   [root@ipaserver ~]# dig +short -t SRV
   _kerberos._udp.ipa.example.com.
   0 100 88 ipamaster1.ipa.example.com.

   [root@ipaserver ~]# dig +short -t SRV
   _ldap._tcp.ipa.example.com.
   0 100 389 ipamaster1.ipa.example.com.
   ```

   The commands are expected to list all IdM servers.

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

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

3. After you execute the **`ipa-adtrust-install`** utility, as described in Section 5.3.4.1, “Preparing the IdM Server for Trust”, run a DNS query for the MS DC Kerberos over UDP and LDAP over TCP service records.

   ```
   [root@ipaserver ~]# dig +short -t SRV
   _kerberos._udp.dc._msdcs.ipa.example.com.
   ```

```
0 100 88 ipamaster1.ipa.example.com.

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

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

Verify that IdM is able to resolve service records for AD

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

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

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

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

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

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

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

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

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

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

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

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

```
> set type=TXT
> _kerberos.ipa.example.com.
_kerberos.ipa.example.com.        text =

    "IPA.EXAMPLE.COM"
```

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

4. After you execute the **ipa-adtrust-install** utility, as described in Section 5.3.4.1, “Preparing the IdM Server for Trust”, run a DNS query for the MS DC Kerberos over UDP and LDAP over TCP service records.

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

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

Verify that AD services are resolvable from the AD server

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

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

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

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

```
    priority = 0
    weight = 100
    port = 389
    svr hostname = addc1.ad.example.com
```

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

5.2.3. NetBIOS Names

The NetBIOS name is critical for identifying the AD 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.

Note

The NetBIOS name is usually the far-left component of the domain name. For example, if the domain is **`linux.example.com`**, the NetBIOS name is **`linux`**. If the domain name is **`example.com`**, the NetBIOS name is **`example`**.

The maximum length of the NetBIOS name is 15 characters.

5.2.4. Firewalls and Ports

IdM uses the following ports to communicate with its services: see the corresponding chapter in the Linux Domain Identity, Authentication, and Policy Guide.

For a trust relationship, additionally open the following ports on IdM and Active Directory:

Table 5.1. Ports Required for a Trust

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

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

Active Directory domain controllers and IdM masters must be able to communicate together using the services described in Table 5.1, “Ports Required for a Trust”, as well as using Kerberos, LDAP, and also DNS if the IdM master is running DNS. Ports required for these last three services are listed in in the Linux Domain Identity, Authentication, and Policy Guide.

Opening the Required Ports

Opening ports requires the **firewalld** service to be running. To start **firewalld** as well as to configure it to start automatically when the system boots:

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

Note

You can determine whether **firewalld** is currently running using the **systemctl status firewalld.service** command.

To open the ports using **firewalld**, assuming the default **firewalld** zone is used:

1. Open the ports to services required by IdM.

   ```
   [root@server ~]# firewall-cmd --permanent --add-port=
   {80/tcp,443/tcp,389/tcp,636/tcp,88/tcp,464/tcp,53/tcp,88/udp,464/u
   dp,53/udp,123/udp}
   ```

2. Open the ports to services required for a trust relationship.

   ```
   [root@server ~]# firewall-cmd --permanent --add-port=
   {135/tcp,138/tcp,139/tcp,445/tcp,138/udp,139/udp,389/udp,445/udp,10
   24-1300/tcp}
   ```

3. Reload the **firewalld** configuration, so that the change is applied immediately.

   ```
   [root@server ~]# firewall-cmd --reload
   ```

5.2.5. IPv6 Settings

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

5.2.6. Clock Settings

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

5.2.7. Supported User Name Formats

IdM performs user name mapping in the local SSSD client. The default user name format supported by SSSD is **name@domain**. Active Directory supports several different kinds of name formats: **username**, **username@domain.name**, and **DOMAIN\username**.

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

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

5.3. Creating Trusts

The following sections describe creating trusts in various configuration scenarios. Section 5.3.4, "Creating a Trust from the Command Line" contains the full procedure for configuring a trust from the command line. The other sections describe the steps which are different from this basic configuration scenario and reference the basic procedure for all other steps.

5.3.1. One-Way and Two-Way Trusts

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

One-way trust

One-way trust enables users and groups in Active Directory to access resources in IdM but not the other way around. The IdM domain trusts the Active Directory forest using the AD cross-forest trust feature, but the AD forest does not trust the IdM domain.

One-way trust is the default mode for creating a trust. Running the **`ipa trust-add`** command without the **`--two-way`** option automatically sets up a one-way trust.

Two-way trust

Two-way trust enables AD users and groups to access resources in IdM.

Note that the two-way trust in IdM does not give the users any additional rights compared to the one-way trust solution in AD. Both solutions are considered equally secure because of default cross-forest trust SID filtering settings.

To set up a two-way trust, pass the **`--two-way=true`** option with the **`ipa trust-add`** command.

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

5.3.2. External Trusts to Active Directory

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

External trusts are non-transitive. For this reason, users and groups from other Active Directory domains have no access to IdM resources. For further information, see Section 5.1.1, "Transitive and Non-transitive Trusts".

5.3.3. Trust Controllers and Trust Agents

To allow full flexibility for different use cases, Identity Management allows to choose from three types of master servers with regards to support of trust to Active Directory:

Normal IdM masters

Normal IdM master servers run the LDAP server, Kerberos KDC server, IdM management framework, SSSD, and a number of optional services, such as the CA, KRA, or DNS. These servers do not support trust to AD.

Trust agents

Trust agents are IdM masters allowed to perform identity lookups against AD domain controllers. Only IdM clients enrolled into trust agents can look up users and groups from trusted AD forests.

Trust agents are typically used to resolve AD users and groups but not to manage the trust.

Trust controllers

Trust controllers are implicitly trust agents as well, but in addition, they also run the Samba suite to allow AD domain controllers to communicate with IdM when trust to AD is established and verified.

Trust controllers can be used for trust management operations, such as adding trust agreements and enabling or disabling separate domains from a trusted forest to access IdM resources. Additionally, AD domain controllers contact trust controllers when validating the trust.

Red Hat recommends to reduce the number of trust controllers in a cross-forest trust deployment. Because trust controllers are required only for management purposes, trust agents should be used instead for enrolling IdM clients designed to run services accessible by users from trusted AD forests. Trust controllers run an increased amount of network-facing services compared to trust agents, and thus present a greater attack surface for potential intruders.

When setting up a trust to an AD forest, at least one IdM trust controller is required. To set up a normal IdM server as a trust controller, configure the trust by using the **`ipa-adtrust-install`** utility without adding the **`--add-agents`** option.

To turn a normal IdM master into a trust agent, run the **`ipa-adtrust-install --add-agents`** command on a trust controller. The command then enters interactive configuration session and prompts you for the information required to set up the agent. After setting up a trust agent, restart the LDAP service. For more information about the **`--add-agents`** option, see the ipa-adtrust-install(1) man page.

Note

IdM does not support downgrading trust controllers to trust agents. If all IdM servers in the topology are trust controllers, you cannot turn some of them into trust agents.

5.3.4. Creating a Trust from the Command Line

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

1. Preparing the IdM server for the trust, described in Section 5.3.4.1, “Preparing the IdM Server for Trust”
2. Creating a trust agreement, described in Section 5.3.4.2, “Creating a Trust Agreement”
3. Verifying the Kerberos configuration, described in Section 5.3.4.3, “Verifying the Kerberos Configuration”

5.3.4.1. Preparing the IdM Server for Trust

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

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

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

2. Configure the IdM server to enable trust services:

   a. Run the **`ipa-adtrust-install`** utility. The utility adds DNS service records required for AD trusts. These records are created automatically if IdM was installed with an integrated DNS server.

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

      Important

      Red Hat strongly recommends to verify the DNS configuration as described in Section 5.2.2, “Verifying the DNS Configuration” every time after running **`ipa-adtrust-install`**, especially if IdM or AD do not use integrated DNS servers.

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

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

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

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

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

3. Make sure that DNS is properly configured, as described in Section 5.2.2, “DNS and Realm Settings”.

4. Start the **smb** daemon and use the **`smbclient`** utility to verify that Samba responds to Kerberos authentication from the IdM side.

```
[root@ipaserver ~]# systemctl start smb

[root@ipaserver ~]# smbclient -L ipaserver.ipa.example.com -k
lp_load_ex: changing to config backend registry
Domain=[IDM] OS=[Windows 6.1] Server=[Samba 4.2.10]
    Sharename       Type      Comment
    ---------       ----      -------
 IPC$            IPC       IPC Service (Samba 4.2.10)
 Domain=[IDM] OS=[Windows 6.1] Server=[Samba 4.2.10]
```

```
        Server               Comment
        ---------            -------
        Workgroup            Master
        ---------            -------
```

5.3.4.2. Creating a Trust Agreement

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

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

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

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

To establish an external trust, pass the **`--external=true`** option to the **`ipa trust-add`** command.

5.3.4.3. Verifying the Kerberos Configuration

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

To verify a two-way trust:

1. Request a ticket for an IdM user:

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

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

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

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

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

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

   ```
   [root@ipaserver ]# klist
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: 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.ipa.example.com@IPA.DOMAIN
   06/15/12 12:13:23 06/16/12 12:12:55 krbtgt/AD.DOMAIN@IPA.DOMAIN
   06/15/12 12:14:58  06/15/12 22:14:58
   cifs/adserver.ad.example.com@AD.DOMAIN
   ```

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

1. Request a ticket for an Active Directory user:

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

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

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

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

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

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

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

Note

For more information about the plug-in, see Section 5.9.1, “Using SSH Without Passwords”.

5.3.5. Creating a Trust with a Shared Secret

A shared secret is a password that is known to trusted peers and can be used by other domains to join the trust. Trusts within Active Directory can be configured with a shared secret. In AD, the shared secret is stored as a *trusted domain object* (TDO) within the trust configuration.

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

To create a trust with a shared secret:

1. Prepare the IdM server for the trust, as described in Section 5.3.4.1, “Preparing the IdM Server for Trust”.

2. Configure a trust in the **`Active Directory Domains and Trusts`** console. Use these settings:

   - Right-click the appropriate domain, and choose **`Properties`**.
   - Navigate to the **`Trusts`** tab, and click the **`New Trust`** button.
   - Enter the IdM DNS name. For example, **`idm.example.com`**.
   - On the **`Trust Type`** page, select **`Forest trust`**.
   - On the **`Direction of Trust`** page, choose **`One-way: incoming`**.
   - On the **`Sides of Trust`** page, select **`This domain only`**.
   - Set the **`Trust Password`**.

   Note

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

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

3. Create a trust agreement, as described in Section 5.3.4.2, “Creating a Trust Agreement”. When running the **`ipa trust-add`** command, use the **`--type`** and **`--trust-secret`** options, and omit the **`--admin`** option. For example:

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

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

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

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

Note

Before running **`ipa trust-show`**, you might be required to run the **`ipa trust-fetch-domains`** ***`ad_domain`*** command to ensure you obtain a Common Internet File System (CIFS) ticket-granting ticket.

5. Verify the Kerberos configuration, as described in Section 5.3.4.3, “Verifying the Kerberos Configuration”.

5.3.6. Creating a Trust on an Existing IdM Instance

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

1. Prepare the IdM server for the trust, as described in Section 5.3.4.1, “Preparing the IdM Server for Trust”.
2. Create a trust agreement, as described in Section 5.3.4.2, “Creating a Trust Agreement”.
3. Generate SIDs for each IdM user.

Note

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

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

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

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

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

4. Verify the Kerberos configuration, as described in Section 5.3.4.3, “Verifying the Kerberos Configuration”.

5.3.7. Adding a Second Trust

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

1. Make sure that DNS is properly configured, as described in Section 5.2.2, “DNS and Realm Settings”.

2. Create a trust agreement, as described in Section 5.3.4.2, “Creating a Trust Agreement”.

5.3.8. Creating a Trust in the Web UI

Before creating a trust in the web UI, prepare the IdM server for the trust. This trust configuration is easiest to perform from the command line, as described in Section 5.3.4.1, “Preparing the IdM Server for Trust”.

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

1. Open the IdM web UI:

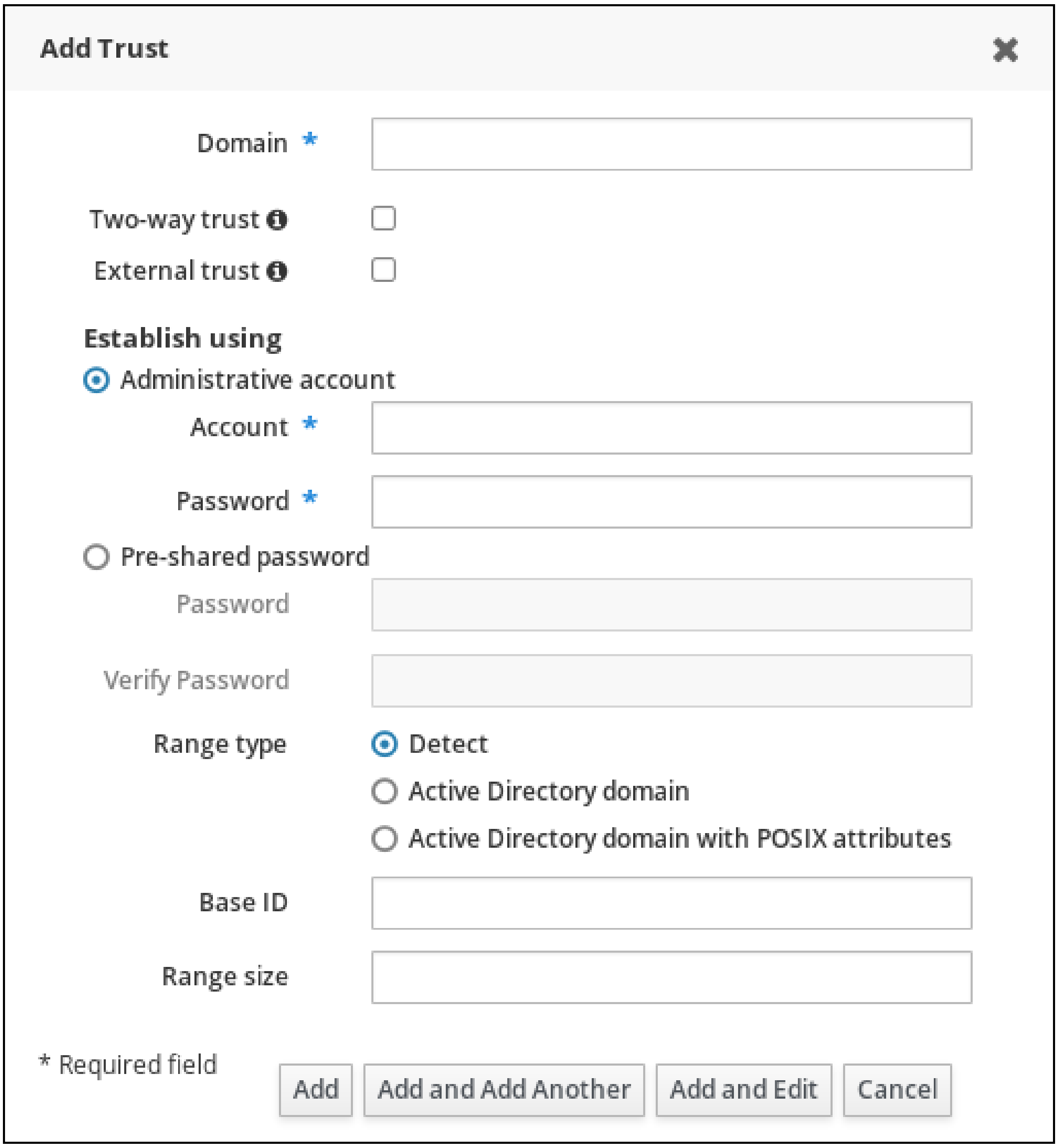
   ```
   https://ipaserver.example.com
   ```

2. Open the **IPA Server** main tab, and select the **Trusts** subtab.
3. In the **Trusts** subtab, click **Add** to open the new trust configuration window.
4. Fill in the required information about the trust:
   a. Provide the AD domain name in the **Domain** field.
   b. To set up the trust as two-way, select the **Two-way trust** check box. To set up the trust as one-way, leave **Two-way trust** unselected.

      For more information about one-way and two-way trusts, see Section 5.3.1, “One-Way and Two-Way Trusts”.
   c. To establish an external trust to a domain in another forest, select the **External Trust** check box.

      For more information, see Section 5.3.2, “External Trusts to Active Directory”.
   d. The **Establish using** section defines how the trust is to be established:
      - To establish the trust using the AD administrator's user name and password, select **Administrative account** and provide the required credentials.
      - Alternatively, to establish the trust with a shared password, select **Pre-shared password** and provide the trust password.
   e. Define the ID configuration for the trust:
      - The **Range type** option allows you to choose the ID range type. If you want IdM to automatically detect what kind of ID range to use, select **Detect**.
      - To define the starting ID of the ID range, use the **Base ID** field. To define the size of the ID range, use the **Range size** field. If you want IdM to use default values for the ID range, do not specify these options.

      For more information about ID ranges, see Section 5.1.3.2, “ID Ranges”.



- Domain *:
- Two-way trust: ☐
- External trust: ☐
- Establish using: [✓ Administrative account] Pre-shared password
  - Account *:
  - Password *:
  - Password:
  - Verify Password:
- Range type: [✓ Detect] Active Directory domain / Active Directory domain with POSIX attributes
- Base ID:
- Range size:

* Required field

Figure 5.5. Adding a Trust in the Web UI

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

After this, verify the Kerberos configuration, as described in Section 5.3.4.3, “Verifying the Kerberos Configuration”.

5.4. IdM clients in an Active Directory DNS Domain

In some environments with trusts between IdM and Active Directory, you can configure users to access an IdM client using a host name from the Active Directory DNS domain, while the client itself is joined to IdM to benefit from its Linux-focused features.

Important

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

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

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

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

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

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

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

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

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

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

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

Note

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

Handling of SSL certificates

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

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

```
[root@idm-client.ad.example.com ~]# ipa-getcert request -r \
```

```
         -f /etc/httpd/alias/server.crt \
         -k /etc/httpd/alias/server.key \
         -N CN=ipa-client.ad.example.com \
         -D ipa-client.ad.example.com \
         -K host/idm-client.ad.example.com@IDM.EXAMPLE.COM \
         -U id-kp-serverAuth
```

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

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

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

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

```
ignore_acceptor_hostname = true
```

Handling of SSL certificates

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

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

1. Create a new host object:

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

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

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

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

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

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

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

5.5. Creating IdM Groups for Active Directory Users

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

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

Note

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

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

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

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

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

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

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

   For example, for an AD group:

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

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

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

5.6. Maintaining Trusts

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

5.6.1. Editing the Global Trust Configuration

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

The global trust configuration contains five attributes:

- A Windows-style security ID (SID); this attribute is autogenerated and cannot be modified
- A domain GUID; this attribute is autogenerated and cannot be modified
- A Kerberos domain name; this attribute comes from the IdM configuration and cannot be modified
- The default group to which to add IdM users; this attribute can be modified
- The NetBIOS name; it is not recommended to modify this attribute

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

5.6.1.1. Changing the NetBIOS Name

Important

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

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

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

5.6.1.2. Changing the Default Group for Windows Users

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

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

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

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

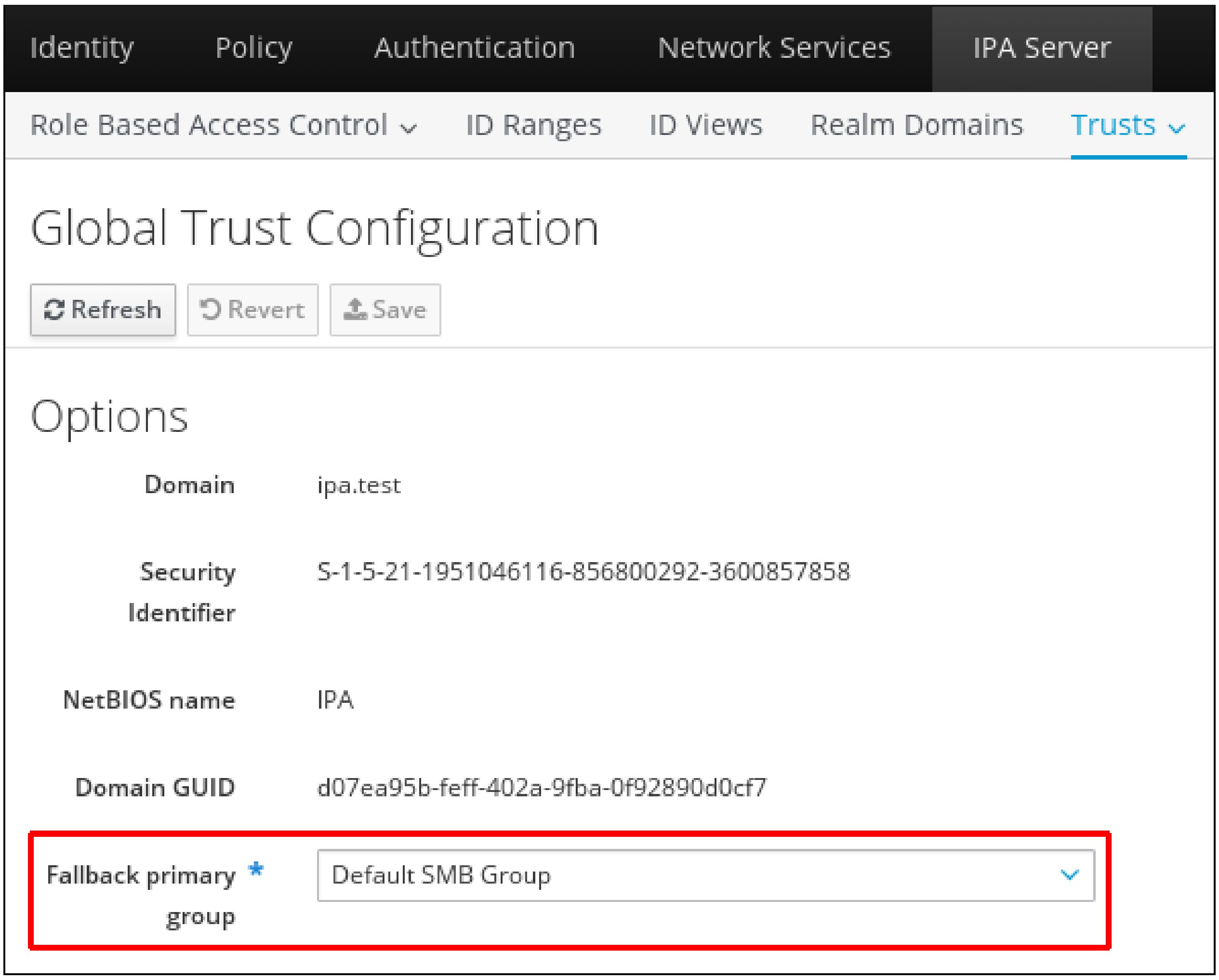
To set the default group from the IdM web UI:

1. Open the IdM web UI.

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

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

3. Select a new group from all of the IdM groups in the **Fallback primary group** drop-down list.



Global Trust Configuration

Options

Domain: ipa.test

Security Identifier: S-1-5-21-1951046116-856800292-3600857858

NetBIOS name: IPA

Domain GUID: d07ea95b-feff-402a-9fba-0f92890d0cf7

Fallback primary group *: Default SMB Group

Figure 5.6. Configuring the Default Group for Windows Users

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

5.6.2. Discovering, Enabling, and Disabling Trust Domains

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

IdM has a trust with the root domain in a forest, and due to transitivity, all of its 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 or beneficial to retrieve that topology manually. That is done with the **trust-fetch-domains** command:

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

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

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

Note

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

Once the topology is retrieved (through automatic or manual discovery), individual domains and 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.ad.example.com
------------------------------------------
Disabled trust domain "test.ad.example.com"
------------------------------------------
```

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

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

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

5.6.3. 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=ad.example.com
 Domain: ipa.example.org, ipa.example.com, example.com, ad.example.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,ad.example.com}
```

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

Creating ID ranges at the time when a trust is originally configured is described in Section 5.1.3.2, “ID Ranges”. To add an ID range later, use the **`ipa idrange-add`** command with the following options:

- the **`--base-id`** option sets the base ID for the POSIX range, which is the starting number
- the **`--range-size`** option sets the size of the range
- the **`--rid-base`** option sets the starting number of the RID, which is the far-right number in the SID; the value represents the range to add to the base ID to prevent conflicts
- the **`--dom-sid`** option sets the domain SID, because there can be multiple domains configured for trusts

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

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

Important

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

5.6.5. Kerberos Flags for Services and Hosts

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

For more information and how to set Kerberos flags, see the corresponding chapter in the Linux Domain Identity, Authentication, and Policy Guide.

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.

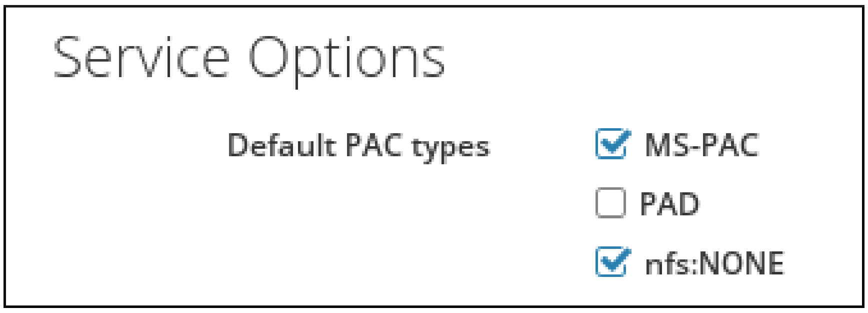
Group information in Active Directory is stored in a list of identifiers in each Kerberos ticket for Active Directory users in a special data set called *privileged access certificates* or MS-PAC. The group information in the PAC has to be mapped to the Active Directory groups and then to the corresponding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

IdM services can be configured to generate PACs 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.


Service Options

Default PAC types: ✓ MS-PAC, ☐ PAD, ✓ nfs:NONE

Figure 5.7. The `Service Options` Area

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

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

Note

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

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

5.7.2. 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.

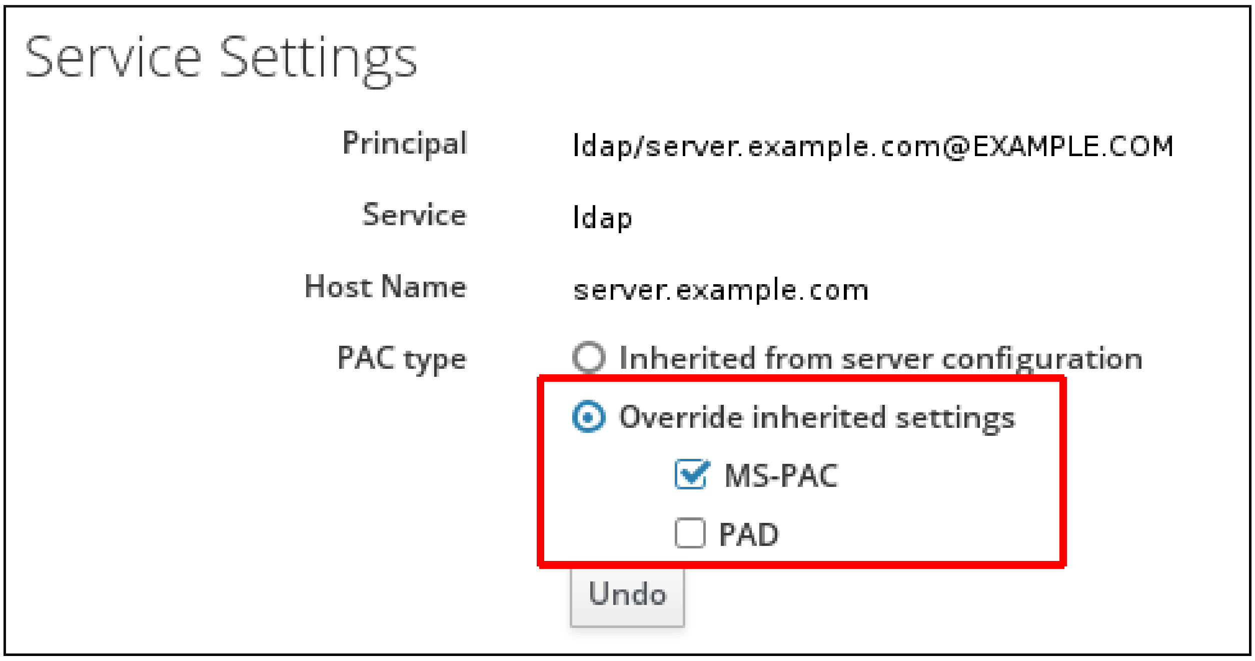
To change the PAC setting from the command line, use the **ipa service-mod** command with the **--pac-type** option. For information on how to use the command, run it with the **--help** option added:

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

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

To change the PAC setting in the web UI:

1. Open the **Identity** tab, and select the **Services** subtab.
2. Click the name of the service to edit.
3. In the **Service Settings** area, check the **Override inherited settings** option and then select the **MS-PAC** check box to add a certificate that can be used by AD services.


Service Settings

Principal	ldap/server.example.com@EXAMPLE.COM
Service	ldap
Host Name	server.example.com
PAC type	○ Inherited from server configuration
	● Override inherited settings
	✓ MS-PAC
	☐ PAD

Figure 5.8. The Service Settings Area

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

Note

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

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

5.8. Transferring POSIX Attributes from Active Directory

Important

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

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

- the **loginShell** attribute, which specifies the AD user's shell
- the **unixHomeDirectory** attribute, which specifies the AD user's home directory

When a custom shell or home directory value is defined on the AD server using these attributes, the custom value is then displayed to the IdM client for the AD user. Therefore, the same user shell is displayed for the AD user both on the AD side and on the IdM side.

Note that to display the AD user's home directory to the IdM client, the ***subdomain_homedir*** option in the **[domain]** section of the **/etc/sssd/sssd.conf** file on the IdM server must be set to **%o**. The **%o** value represents the home directory retrieved from the identity provider. For example:

```
[domain/example.com]
subdomain_homedir = %o
```

If the AD administrator modifies **loginShell** or **unixHomeDirectory** on the AD side, the change is automatically reflected on the IdM side as well. If the attributes are not defined on the AD server, SSSD uses a template default value. This default value is then displayed to the IdM client.

5.9. Using SSH from Active Directory Machines for IdM Resources

When a trust is configured, Active Directory users can access machines, services, and files on IdM hosts using SSH and their AD credentials.

5.9.1. Using SSH Without Passwords

The **localauth** Kerberos plug-in for local authorization ensures that Kerberos principals are automatically mapped to local SSSD user names. With **localauth**, Windows users from a trusted AD domain are not prompted for a password when logging in using Kerberos and can therefore use SSH without passwords.

The plug-in provides a reliable mapping mechanism across multiple realms and trusts: when **sssd** connects to the Kerberos library to map the principal to a local POSIX identity, the SSSD plug-in maps them according to the trust agreements defined in IdM.

Kerberos Authentication for AD Users on Red Hat Enterprise Linux 7.1 and newer Systems

In Red Hat Enterprise Linux 7.1 and newer systems, SSSD automatically configures the **localauth** Kerberos plug-in.

SSSD allows user names in the format **user@AD.DOMAIN**, **ad.domain\user** and **AD\user**.

Note

On systems with **localauth**, it is not required to set the ***auth_to_local*** option in the **/etc/krb5.conf** file or list Kerberos principals in the **.k5login** file. The **localauth** plug-in makes this previously used configuration for logins without passwords obsolete.

Manual Configuration of Kerberos Authentication for AD Users

On systems where the **localauth** plug-in is not present, SSH prompts for a user password for Active Directory domain users even if the user obtains a proper Kerberos ticket.

To enable Active Directory users to use Kerberos for authentication in this situation, configure the ***auth_to_local*** option in the **/etc/krb5.conf** file or list the user Kerberos principals in the **.k5login** file in the home directory of the user.

Configuring /etc/krb5.conf

The following procedure describes how to configure realm mapping in the Kerberos configuration.

1. Open the **/etc/krb5.conf** file.
2. In the **[realms]** section, identify the IdM realm by name, and then add two ***auth_to_local*** lines to define the Kerberos principal name mapping:
   - In one rule, include a rule to map different Active Directory user name formats and the specific Active Directory domain.
   - In the other rule, set the value of **DEFAULT**, for standard Unix user names.

   For example:

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

3. Restart the KDC service.

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

Note that if you configure Kerberos authentication using the ***auth_to_local*** option, the user name used for SSH access must meet the following criteria:

- The user name must have the format **ad_user@ad_domain**.
- The domain name must be lowercase.
- The case of the user name must match the case of the user name in Active Directory. For example, **user** and **User** are considered different users because of the different cases.

For more information about setting ***auth_to_local***, see the krb5.conf(5) man page.

Configuring .k5login

The following procedure configures the system to find the Kerberos principal name for a local user name.

1. Create the **.k5login** file in the user's home directory.
2. List the Kerberos principals used by the user in the file.

If the authenticating user matches the principal in an existing Kerberos ticket, the user is allowed to log in using the ticket and is not prompted for a password.

Note that if you configure Kerberos authentication using the **.k5login** configuration, the user name used for SSH access must have the format **ad_user@ad_domain**.

For more information about configuring the **.k5login** file, see the .k5login(5) man page.

Either one of these configuration procedures results in AD users being able to log in using Kerberos.

5.10. Using a Trust with Kerberos-enabled Web Applications

Any existing web application can be configured to use Kerberos authentication, which references the trusted Active Directory and IdM Kerberos realms. For the full Kerberos configuration directives, see the Configuration page for the mod_auth_kerb module.

Note

After changing the Apache application configuration, restart the Apache service:

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

For example, for an Apache server, there are several options that define how the Apache server connects to the IdM Kerberos realm:

KrbAuthRealms

The ***KrbAuthRealms*** option gives the application location to the name of the IdM domain. This is required.

Krb5Keytab

The ***Krb5Keytab*** option gives the location for the IdM server keytab. This is required.

KrbServiceName

The ***KrbServiceName*** option sets the Kerberos service name used for the keytab (HTTP). This is recommended.

KrbMethodK5Passwd **and** ***KrbMethodNegotiate***

The ***KrbMethodK5Passwd*** Kerberos method option enables password-based authentication for valid users. The ***KrbMethodNegotiate*** option enables single sign-on (SSO) if a valid Kerberos ticket is available.

These options are recommended for ease of use for many users.

KrbLocalUserMapping

The ***KrbLocalUserMapping*** option enables normal web logins (which are usually the UID or common name of the account) to be mapped to the fully-qualified user name (which has a format of *user@REALM.COM*).

This option is strongly recommended. Without the domain name/login name mapping, the web login appears to be a different user account than the domain user. This means that users cannot see their expected data.

For information on supported user name formats, see Section 5.2.7, “Supported User Name Formats”.

Example 5.1. Kerberos Configuration in an Apache Web Application

```
<Location "/mywebapp">
   AuthType Kerberos
   AuthName "IPA Kerberos authentication"
   KrbMethodNegotiate on
   KrbMethodK5Passwd on
   KrbServiceName HTTP
   KrbAuthRealms IDM_DOMAIN
   Krb5Keytab /etc/httpd/conf/ipa.keytab
   KrbLocalUserMapping on
   KrbSaveCredentials off
   Require valid-user
</Location>
```

5.11. Active Directory Trust for Legacy Linux Clients

Linux clients running Red Hat Enterprise Linux with SSSD version 1.8 or earlier (*legacy clients*) do not provide native support for IdM cross-forest trusts with Active Directory. Therefore, for AD users to be able to access services provided by the IdM server, the legacy Linux clients and the IdM server have to be properly configured.

Instead of using SSSD version 1.9 or later to communicate with the IdM server to obtain LDAP information, legacy clients use other utilities for this purpose, for example **`nss_ldap`**, **`nss-pam-ldapd`**, or SSSD version 1.8 or earlier. Clients running the following versions of Red Hat Enterprise Linux do not use SSSD 1.9 and are therefore considered to be legacy clients:

- Red Hat Enterprise Linux 5.7 or later

- Red Hat Enterprise Linux 6.0 – 6.3

Important

Do not use the configuration described in this section for non-legacy clients, that is, clients running SSSD version 1.9 or later. SSSD 1.9 or later provides native support for IdM cross-forest trusts with AD, meaning AD users can properly access services on IdM clients without any additional configuration.

When a legacy client joins the domain of an IdM server in a trust relationship with AD, a *compat LDAP tree* provides the required user and group data to AD users. However, the compat tree enables the AD users to access only a limited number of IdM services.

Legacy clients *do not* provide access to the following services:

- Kerberos authentication
- host-based access control (HBAC)
- SELinux user mapping
- **`sudo`** rules

Access to the following services *is provided* even in case of legacy clients:

- information look-up
- password authentication

5.11.1. Server-side Configuration for AD Trust for Legacy Clients

Make sure the IdM server meets the following configuration requirements:

- The *ipa-server* package for IdM and the *ipa-server-trust-ad* package for the IdM trust add-on have been installed.
- The **`ipa-server-install`** utility has been run to set up the IdM server.
- The **`ipa-adtrust-install --enable-compat`** command has been run, which ensures that the IdM server supports trusts with AD domains and that the compat LDAP tree is available.

  If you have already run **`ipa-adtrust-install`** without the **`--enable-compat`** option in the past, run it again, this time adding **`--enable-compat`**.
- The **`ipa trust-add`** ***`ad.example.org`*** command has been run to establish the AD trust.

If the host-based access control (HBAC) **`allow_all`** rule is disabled, enable the **`system-auth`** service on the IdM server, which allows authentication of the AD users.

You can determine the current status of **`allow_all`** directly from the command line using the **`ipa hbacrule-show`** command. If the rule is disabled, **`Enabled: FALSE`** is displayed in the output:

```
[user@server ~]$ kinit admin
[user@server ~]$ ipa hbacrule-show allow_all
  Rule name: allow_all
  User category: all
  Host category: all
  Service category: all
  Description: Allow all users to access any host from any host
  Enabled: FALSE
```

Note

For information on disabling and enabling HBAC rules, see the Linux Domain Identity, Authentication, and Policy Guide.

To enable **system-auth** on the IdM server, create an HBAC service named **system-auth** and add an HBAC rule using this service to grant access to IdM masters. Adding HBAC services and rules is described in the Linux Domain Identity, Authentication, and Policy Guide. Note that HBAC services are PAM service names; if you add a new PAM service, make sure to create an HBAC service with the same name and then grant access to this service through HBAC rules.

5.11.2. Client-side Configuration Using the `ipa-advise` Utility

The **ipa-advise** utility provides the configuration instructions to set up a legacy client for an AD trust.

To display the complete list of scenarios for which **ipa-advise** can provide configuration instructions, run **ipa-advise** without any options. Running **ipa-advise** prints the names of all available sets of configuration instructions along with the descriptions of what each set does and when it is recommended to be used.

```
[root@server ~]# ipa-advise
config-redhat-nss-ldap  : Instructions for configuring a system
      with nss-ldap as a IPA client.
      This set of instructions is targeted
      for platforms that include the
      authconfig utility, which are all
      Red Hat based platforms.
config-redhat-nss-pam-ldapd : Instructions for configuring a system
(...)
```

To display a set of instructions, run the **ipa-advise** utility with an instruction set as a parameter:

```
[root@server ~]# ipa-advise config-redhat-nss-ldap
#!/bin/sh
# ----------------------------------------------------------------------
# Instructions for configuring a system with nss-ldap as a IPA client.
# This set of instructions is targeted for platforms that include the
# authconfig utility, which are all Red Hat based platforms.
# ----------------------------------------------------------------------
# Schema Compatibility plugin has not been configured on this server. To
# configure it, run "ipa-adtrust-install --enable-compat"
# Install required packages via yum
yum install -y wget openssl nss_ldap authconfig

# NOTE: IPA certificate uses the SHA-256 hash function. SHA-256 was
# introduced in RHEL5.2. Therefore, clients older than RHEL5.2 will not
# be able to interoperate with IPA server 3.x.
# Please note that this script assumes /etc/openldap/cacerts as the
# default CA certificate location. If this value is different on your
# system the script needs to be modified accordingly.
# Download the CA certificate of the IPA server
mkdir -p -m 755 /etc/openldap/cacerts
wget http://idm.example.com/ipa/config/ca.crt -O
/etc/openldap/cacerts/ca.crt
(...)
```

You can configure a Linux client using the **ipa-advise** utility by running the displayed instructions as a shell script or by executing the instructions manually.

To run the instructions as a shell script:

1. Create the script file.

   ```
   [root@server ~]# ipa-advise config-redhat-nss-ldap >
   setup_script.sh
   ```

2. Add execute permissions to the file using the **chmod** utility.

   ```
   [root@server ~]# chmod +x setup_script.sh
   ```

3. Copy the script to the client using the **scp** utility.

   ```
   [root@server ~]# scp setup_script.sh root@client
   ```

4. Run the script on the client.

   ```
   [root@client ~]# ./setup_script.sh
   ```

Important

Always read and review the script file carefully before you run it on the client.

To configure the client manually, follow and execute the instructions displayed by **ipa-advise** from the command line.

5.12. Smart Card Certificates in a Trusted Active Directory Environment

For Active Directory users in a trusted environment, there are two options where you can store the smart card certificate:

In Active Directory

If the smart card certificate is stored in a trusted Active Directory, IdM automatically reads the **userCertificate** attribute from Active Directory user objects. No further actions are required.

In IdM

If you cannot place the smart card certificate in the Active Directory for some reason, you can instead store it in IdM. Use the **`Default Trust View`** to override the certificate on all hosts in the IdM domain or create a new ID view to override it on specific hosts.

For more details about ID views and how to use them, see the corresponding chapter in the Linux Domain Identity, Authentication, and Policy Guide.

For more information how to use smart cards see the corresponding chapter in the Linux Domain Identity, Authentication, and Policy Guide.

Chapter 6. Synchronizing Active Directory and Identity Management Users

Red Hat Enterprise Linux Identity Management uses *synchronization* to combine the user data stored in an Active Directory domain and the user data stored in the IdM domain. Critical user attributes, including passwords, are copied and synchronized between the services.

Entry synchronization is performed through a process similar to replication, which uses hooks to connect to and retrieve directory data from the Windows server.

Password synchronization is performed through a Windows service which is installed on the Windows server and then communicates to the Identity Management server.

6.1. Supported Windows Platforms

Synchronization is supported with these Windows servers:

- Windows Server 2008
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2

PassSync 1.1.5 or later is compatible with all supported Windows Server versions.

6.2. About Active Directory and Identity Management

Within the IdM domain, information is shared among servers and replicas by copying that information, reliably and predictably, between data masters (servers and replicas). This process is *replication*.

A similar process can be used to share data between the IdM domain and a Microsoft Active Directory domain. This is *synchronization*.

Synchronization is the process of copying user data back and forth between Active Directory and Identity Management.

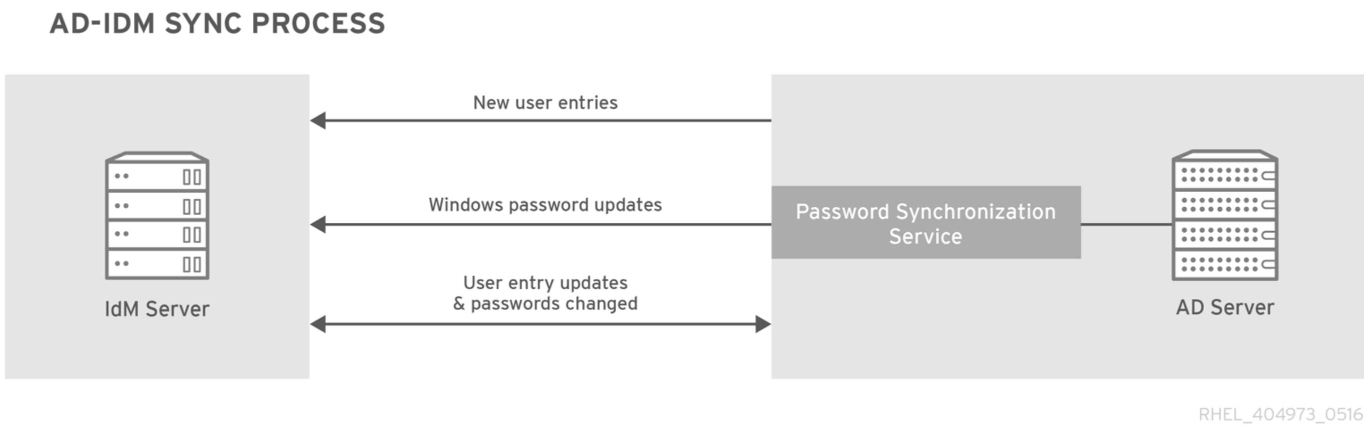


Figure 6.1. Active Directory and IdM Synchronization

Synchronization is defined in an *agreement* between an IdM server and an Active Directory domain controller. The agreement defines all of the information required to identify user entries that can be synchronized, such as the subtree to synchronize, as well as defining how account attributes are handled. The synchronization agreements are created with default values which can be tweaked to meet the needs of a specific domain. When two servers are involved in synchronization, they are called *peers*.

Table 6.1. Information in a Synchronization Agreement

Windows Information	IdM Information
» User subtree (**`cn=Users,$SUFFIX`**) » Connection information ▪ Active Directory administrator user name and password ▪ Password Synchronization Service password ▪ CA certificate	» User subtree (**`ou=People,$SUFFIX`**)

Synchronization is most commonly *bidirectional*. Information is sent back and forth between the IdM and the Windows domain in a process that is very similar to how IdM servers and replicas share information among themselves. An exception are new user entries, which are only added from the Windows domain to the IdM domain. It is possible to configure synchronization to only synchronize one way. That is *unidirectional* synchronization.

To prevent the risk of data conflicts, only one directory should originate or remove user entries. This is typically the Windows directory, which is the primary identity store in the IT environment, and then new accounts or account deletions are synchronized to the Identity Management peer. Either directory can modify entries.

Synchronization, then, is configured between one Identity Management server and one Active Directory domain controller. The Identity Management server propagates throughout to the IdM domain, while the domain controller propagates changes throughout the Windows domain.

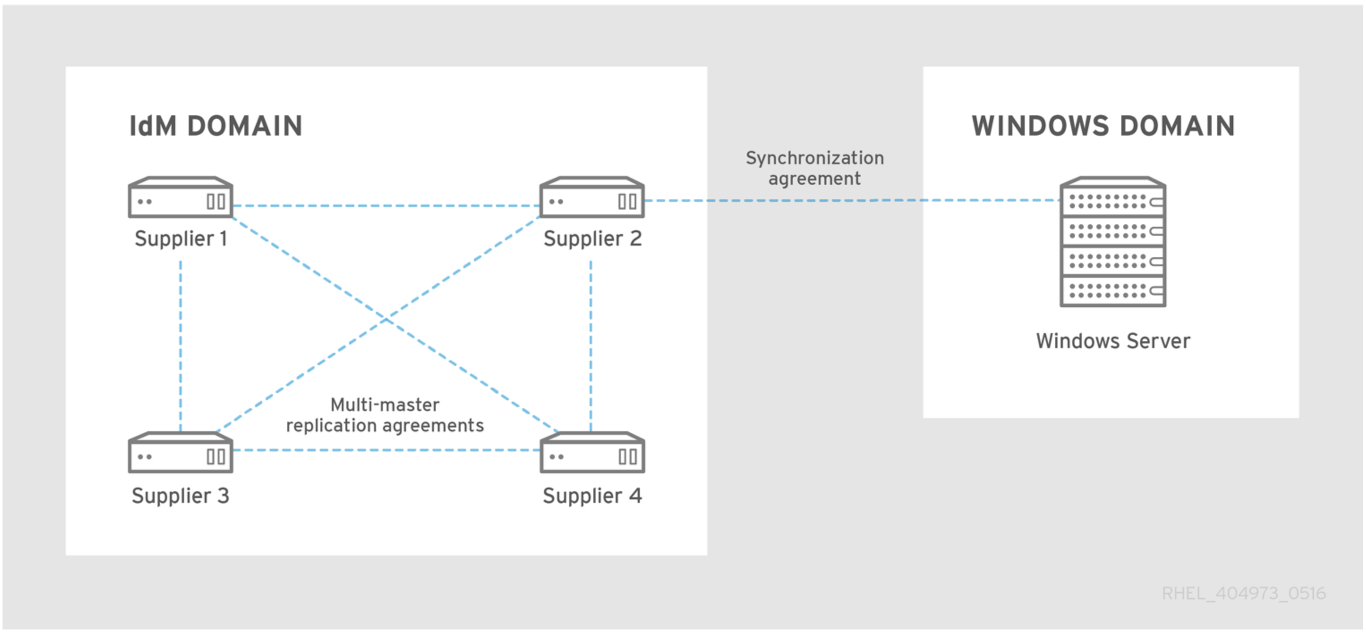


Figure 6.2. Synchronization Topology

There are some key features to IdM synchronization:

- A synchronization operation runs every five minutes. To modify the frequency, set the **`winSyncInterval`** attribute in the Active Directory peers DN:

  ```
  cn=meTowinserver.ad.example.com,cn=replica,cn=dc\3Didm\,dc\3Dexample\,
  dc\3Dcom,cn=mapping tree,cn=config
  ```

- Synchronization can only be configured with one Active Directory domain.
- Synchronization can only be configured with *one* Active Directory domain controller.
- Only user information is synchronized; group information is not.
- Both user attributes and passwords can be synchronized.
- While modifications are bidirectional (going both from Active Directory to IdM and from IdM to Active Directory), creating accounts is only unidirectional, from Active Directory to Identity Management. New accounts created in Active Directory are synchronized over to IdM automatically. However, user accounts created in IdM must also be created in Active Directory before they will be synchronized. In this situation, the synchronization process tries to find a matching account with the same value for the **`uid`** attribute in IdM than for the **`sAMAccountName`** attribute in Active Directory. If a match is found, the IdM **`ntUserDomainId`** attribute is set to the Active Directory **`objectGUID`** value. These attributes are globally unique and immutable, and entries stay synchronized, even if they are moved or renamed.
- Account lock information is synchronized by default, so a user account which is disabled in one domain is disabled in the other.
- Password synchronization changes take effect immediately. If a user password is added or changed on one peer, that change is immediately propagated to the other peer server.

  The Password Synchronization client synchronizes new passwords or password updates.

  Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

- While there can only be one agreement, the PassSync service must be installed on every Active Directory server.

When Active Directory users are synchronized over to IdM, certain attributes (including Kerberos and POSIX attributes) will have IPA attributes automatically added to the user entries. These attributes are used by IdM within its domain. They are not synchronized back over the corresponding Active Directory user entry.

Some of the data in synchronization can be modified as part of the synchronization process. For examples, certain attributes can be automatically added to Active Directory user accounts when they are synced over to the IdM domain. These attribute changes are defined as part of the synchronization agreement and are described in Section 6.5.2, “Changing the Behavior for Synchronizing User Account Attributes”.

6.3. About Synchronized Attributes

Identity Management synchronizes a subset of user attributes between IdM and Active Directory user entries. Any other attributes present in the entry, either in Identity Management or in Active Directory, are ignored by synchronization.

Note

Most POSIX attributes are not synchronized.

Although there are significant schema differences between the Active Directory LDAP schema and the 389 Directory Server LDAP schema used by Identity Management, there are many attributes that are the same. These attributes are simply synchronized between the Active Directory and IdM user entries, with no changes to the attribute name or value format.

User Schema That Are the Same in Identity Management and Windows Servers

- cn [5]
- physicalDeliveryOfficeName
- description
- postOfficeBox
- destinationIndicator
- postalAddress
- facsimileTelephoneNumber
- postalCode
- givenname
- registeredAddress
- homePhone
- sn
- homePostalAddress
- st
- initials
- street
- l
- telephoneNumber
- mail
- teletexTerminalIdentifier
- mobile

- telexNumber
- o
- title
- ou
- userCertificate
- pager
- x121Address

Some attributes have different names but still have direct parity between IdM (which uses 389 Directory Server) and Active Directory. These attributes are *mapped* by the synchronization process.

Table 6.2. User Schema Mapped between Identity Management and Active Directory

Identity Management	Active Directory
cn [a]	name
nsAccountLock	userAccountControl
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

[a] The ***cn*** is mapped directly (***cn*** to ***cn***) when synchronizing from Identity Management to Active Directory. When synchronizing from Active Directory ***cn*** is mapped from the ***name*** attribute in Active Directory to the ***cn*** attribute in Identity Management.

6.3.1. User Schema Differences between Identity Management and Active Directory

Even though attributes may be successfully synchronized between Active Directory and IdM, there may still be differences in how Active Directory and Identity Management define the underlying X.500 object classes. This could lead to differences in how the data are handled in the different LDAP services.

This section describes the differences in how Active Directory and Identity Management handle some of the attributes which can be synchronized between the two domains.

6.3.1.1. Values for cn Attributes

In 389 Directory Server, the ***cn*** attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the Identity Management ***cn*** attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that,potentially, if a ***cn*** value is added to an Active Directory entry and that value is not one of the values for ***cn*** in Identity Management, then all of the Identity Management ***cn*** values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the ***cn*** attribute as its naming attribute, where Identity Management uses ***uid***. This means that there is the potential to rename the entry entirely (and accidentally) if the ***cn*** attribute is edited in the Identity Management.

6.3.1.2. Values for street and streetAddress

Active Directory uses the attribute ***streetAddress*** for a user's postal address; this is the way that 389 Directory Server uses the ***street*** attribute. There are two important differences in the way that Active Directory and Identity Management use the ***streetAddress*** and ***street*** attributes, respectively:

- In 389 Directory Server, ***streetAddress*** is an alias for ***street***. Active Directory also has the ***street*** attribute, but it is a separate attribute that can hold an independent value, not an alias for ***streetAddress***.
- Active Directory defines both ***streetAddress*** and ***street*** as single-valued attributes, while 389 Directory Server defines ***street*** as a multi-valued attribute, as specified in RFC 4519.

Because of the different ways that 389 Directory Server and Active Directory handle ***streetAddress*** and ***street*** attributes, there are two rules to follow when setting address attributes in Active Directory and Identity Management:

- The synchronization process maps ***streetAddress*** in the Active Directory entry to ***street*** in Identity Management. To avoid conflicts, the ***street*** attribute should not be used in Active Directory.
- Only one Identity Management ***street*** attribute value is synchronized to Active Directory. If the ***streetAddress*** attribute is changed in Active Directory and the new value does not already exist in Identity Management, then all ***street*** attribute values in Identity Management are replaced with the new, single Active Directory value.

6.3.1.3. Constraints on the initials Attribute

For the ***initials*** attribute, Active Directory imposes a maximum length constraint of six characters, but 389 Directory Server does not have a length limit. If an ***initials*** attribute longer than six characters is added to Identity Management, the value is trimmed when it is synchronized with the Active Directory entry.

6.3.1.4. Requiring the surname (sn) Attribute

Active Directory allows **person** entries to be created without a surname attribute. However, RFC 4519 defines the **person** object class as requiring a surname attribute, and this is the definition used in Directory Server.

If an Active Directory **person** entry is created without a surname attribute, that entry will not be synchronized over to IdM since it fails with an object class violation.

6.3.2. Active Directory Entries and POSIX Attributes

When users are synchronized between Active Directory and Identity Management the directory synchronization (DirSync) LDAP server extension control is used to search a directory for objects that have changed.

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 synchronized 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.

6.4. Setting up Active Directory for Synchronization

Synchronizing user accounts is enabled within IdM. It is only necessary to set up a synchronization agreement (Section 6.5.1, “Creating Synchronization Agreements”). However, the Active Directory does need to be configured in a way that allows the Identity Management server to connect to it.

6.4.1. Creating an Active Directory User for Synchronization

On the Windows server, it is necessary to create the user that the IdM server will use to connect to the Active Directory domain.

The process for creating a user in Active Directory is covered in the Windows server documentation at http://technet.microsoft.com/en-us/library/cc732336.aspx. The new user account must have the proper permissions:

- Grant the synchronization user account **Replicating directory changes** rights to the synchronized Active Directory subtree. Replicator rights are required for the synchronization user to perform synchronization operations.

  Replicator rights are described in http://support.microsoft.com/kb/303972.

- Add the synchronization user as a member of the **Account Operators** and **Enterprise Read-only Domain Controllers** groups. It is not necessary for the user to belong to the **Domain Admins** group.

6.4.2. Setting up an Active Directory Certificate Authority

The Identity Management server connects to the Active Directory server using a secure connection. This requires that the Active Directory server have an available CA certificate or CA certificate chain available, which can be imported into the Identity Management security databases, so that the Windows server is a trusted peer.

While this could technically be done with an external (to Active Directory) CA, most deployments should use the Certificate Services available with Active Directory.

The procedure for setting up and configuring certificate services on Active Directory is covered in the Microsoft documentation at http://technet.microsoft.com/en-us/library/cc772393(v=WS.10).aspx.

6.5. Managing Synchronization Agreements

6.5.1. Creating Synchronization Agreements

Synchronization agreements are created on the IdM server using the **`ipa-replica-manage connect`** command because it creates a *connection* to the Active Directory domain. To establish an encrypted connection to Active Directory, IdM must to trust the Windows CA certificate.

1. Copy the root certificate authority (CA) certificate to the IdM server:

   a. If your Active Directory CA certificate is self-signed:

      i. Export the Active Directory CA certificate on the Windows server.

         A. Press the **`Super key`**+**`R`** combination to open the **`Run`** dialog.

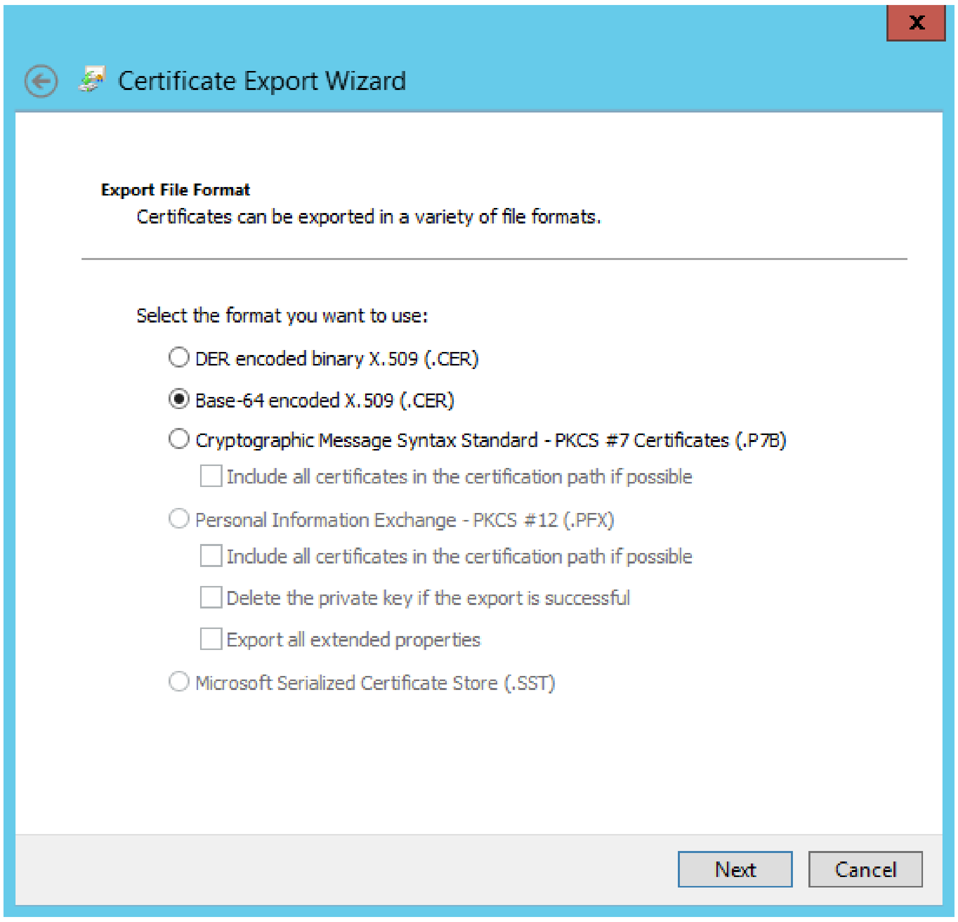
         B. Enter **`certsrv.msc`** and click **`OK`**.

         C. Right-click on the name of the local Certificate Authority and choose **`Properties`**.

         D. On the **`General`** tab, select the certificate to export in the **`CA certificates`** field and click **`View Certificate`**.

         E. On the **`Details`** tab, click **`Copy to File`** to start the **`Certificate Export Wizard`**.

         F. Click **`Next`**, and then select **`Base-64 encoded X.509 (.CER)`**.



Certificate Export Wizard

Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:

- ○ DER encoded binary X.509 (.CER)
- ● Base-64 encoded X.509 (.CER)
- ○ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - ☐ Include all certificates in the certification path if possible
- ○ Personal Information Exchange - PKCS #12 (.PFX)
  - ☐ Include all certificates in the certification path if possible
  - ☐ Delete the private key if the export is successful
  - ☐ Export all extended properties
- ○ Microsoft Serialized Certificate Store (.SST)

         G. Specify a suitable directory and file name for the exported file. Click **`Next`** to export the certificate, and then click **`Finish`**.

H. Copy the exported certificate to the IdM server machine.

b. If your Active Directory CA certificate is signed by an external CA:

i. To find out what certificate is the CA root certificate, display the certificate chain:

```
# openssl s_client -connect adserver.example.com:636
CONNECTED(00000003)
depth=1 C = US, O = Demo Company, OU = IT, CN = Demo
CA-28
verify error:num=20:unable to get local issuer
certificate
verify return:0
---
Certificate chain
 0 s:/C=US/O=Demo Company/OU=IT/CN=adserver.example.com
   i:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
 1 s:/C=US/O=Demo Company/OU=IT/CN=Demo CA-1
   i:/C=US/O=Demo Company/OU=IT/CN=Demo Root CA 2
```

The previous example shows that the Active Directory server's CA certificate is signed by **`CN=Demo CA-1`**, which is signed by **`CN=Demo Root CA 2`**. This means that **`CN=Demo Root CA 2`** is the root CA.

ii. Copy the CA certificate to the IdM server.

2. Remove any existing Kerberos credentials on the IdM server.

```
$ kdestroy
```

3. Use the **`ipa-replica-manage`** command to create a Windows synchronization agreement. This requires the **`--winsync`** option. If passwords will be synchronized as well as user accounts, then also use the **`--passsync`** option and set a password to use for Password Synchronization.

The **`--binddn`** and **`--bindpw`** options give the user name and password of the system account on the Active Directory server that IdM will use to connect to the Active Directory server.

```
$ ipa-replica-manage connect --winsync \
 --binddn cn=administrator,cn=users,dc=example,dc=com \
 --bindpw Windows-secret \
 --passsync secretpwd \
 --cacert /etc/openldap/cacerts/windows.cer \
 adserver.example.com -v
```

- **`--winsync`**: Identifies this as a Windows synchronization agreement.
- **`--binddn`**: IdM uses this DN of an Active Directory account to bind to the remote directory and synchronize attributes.
- **`--bindpw`**: Password for the synchronization account.
- **`--cacert`**: Full path and file name to the:
  - Active Directory CA certificate, if the CA was self-signed.

    - external CA certificate, if the Active Directory CA was signed by an external CA.

  - **`--win-subtree`**: DN of the Windows directory subtree containing the users to synchronize. The default value is **`cn=Users,$SUFFIX`**.

  - **`AD_server_name`**: Fully qualified domain name (FQDN) of the Active Directory domain controller.

4. When prompted, enter the Directory Manager password.

5. *Optional.* Configure Password Synchronization, as in Section 6.6.2, “Setting up Password Synchronization”. Without the Password Synchronization client, user attributes are synchronized between the peer servers, but passwords are not.

   Note

   The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.

   Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

6.5.2. Changing the Behavior for Synchronizing User Account Attributes

When the synchronization agreement is created, it has certain default behaviors defined for how the synchronization process handles the user account attributes during synchronization. The types of behaviors are things like how to handle lockout attributes or how to handle different DN formats. This behavior can be changed by editing the synchronization agreement.

The synchronization agreement exists as a special plug-in entry in the LDAP server and each attribute behavior is set through an LDAP attribute. To change the synchronization behavior, use the **`ldapmodify`** command to modify the LDAP server entry directly.

For example, account lockout attributes are synchronized between IdM and Active Directory by default, but this can be disabled by editing the ***`ipaWinSyncAcctDisable`*** attribute. (Changing this means that if an account is disabled in Active Directory, it is still active in IdM and vice versa.)

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password

dn: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
replace: ipaWinSyncAcctDisable
ipaWinSyncAcctDisable: none

modifying entry "cn=ipa-winsync,cn=plugins,cn=config"
```

The following is an overview of synchronization settings attributes:

General User Account Parameters

- **`ipaWinSyncNewEntryFilter`**: Sets the search filter to use to find the entry which contains the list of object classes to add to new user entries.

Default value: **`(cn=ipaConfig)`**

- **`ipaWinSyncNewUserOCAttr`**: Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.

  Default value: **`ipauserobjectclasses`**

- **`ipaWinSyncHomeDirAttr`**: Identifies which attribute in the entry contains the default location of the POSIX home directory.

  Default value: **`ipaHomesRootDir`**

- **`ipaWinSyncUserAttr`**: Sets an additional attribute with a specific value to add to Active Directory users when they are synchronized over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and the synchronization process adds all of the values to the entry.

  Example: **`ipaWinSyncUserAttr: attributeName attributeValue`**

  Note

  This only sets the attribute value if the entry does not already have that attribute present. If the attribute is present, then the entry's value is used when the Active Directory entry is synchronized over.

- **`ipaWinSyncForceSync`**: Sets whether existing IdM users that match existing AD users should be forced to be synchronized. When set to **`true`**, such IdM users are automatically edited so that they are synchronized.

  Possible values: **`true | false`**

  If an IdM user account has a ***`uid`*** parameter which is identical to the ***`sAMAccountName`*** in an existing Active Directory user, then that account is *not* synchronized by default. This attribute tells the synchronization service to add the ***`ntUser`*** and ***`ntUserDomainId`*** to the IdM user entries automatically, which allows them to be synchronized.

User Account Lock Parameters

- **`ipaWinSyncAcctDisable`**: Sets which way to synchronize account lockout attributes. It is possible to control which account lockout settings are in effect. For example, **`to_ad`** means that when account lockout attribute is set in IdM, its value is synchronized over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synchronized from both domains.

  Possible values: **`both`** (default), **`to_ad`**, **`to_ds`**, **`none`**

- **`ipaWinSyncInactivatedFilter`**: Sets the search filter to use to find the DN of the group used to hold inactivated (disabled) users. This does not need to be changed in most deployments.

  Default value: **`(&(cn=inactivated)(objectclass=groupOfNames))`**

Group Parameters

- **`ipaWinSyncDefaultGroupAttr`**: Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the ***`gidNumber`*** for the user account.

  Default value: **`ipaDefaultPrimaryGroup`**

- **`ipaWinSyncDefaultGroupFilter`**: Sets the attribute in the new user account to reference to see what the default group for the user is. The group name in the entry is then used to find the ***`gidNumber`*** for the user account.

  Default value: **`ipaDefaultPrimaryGroup`**

Realm Parameters

- **`ipaWinSyncRealmAttr`**: Sets the attribute which contains the realm name in the realm entry.

  Default value: ***`cn`***

- **`ipaWinSyncRealmFilter`**: Sets the search filter to use to find the entry which contains the IdM realm name.

  Default value: **`(objectclass=krbRealmContainer)`**

6.5.3. Changing the Synchronized Windows Subtree

Creating a synchronization agreement automatically sets the two subtrees to use as the synchronized user database. In IdM, the default is **`cn=users,cn=accounts,$SUFFIX`**, and for Active Directory, the default is **`CN=Users,$SUFFIX`**.

The value for the Active Directory subtree can be set to a non-default value when the synchronization agreement is created by using the **`--win-subtree`** option. After the agreement is created, the Active Directory subtree can be changed by using the **`ldapmodify`** command to edit the ***`nsds7WindowsReplicaSubtree`*** value in the synchronization agreement entry.

1. Get the name of the synchronization agreement, using **`ldapsearch`**. This search returns only the values for the ***`dn`*** and ***`nsds7WindowsReplicaSubtree`*** attributes instead of the entire entry.

```
[jsmith@ipaserver ~]$ ldapsearch -xLLL -D "cn=directory manager" -w
password -p 389 -h ipaserver.example.com -b cn=config
objectclass=nsdswindowsreplicationagreement dn
nsds7WindowsReplicaSubtree

dn:
cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dco
m,cn=mapping tree,cn=config
nsds7WindowsReplicaSubtree: cn=users,dc=example,dc=com

... 8< ...
```

2. Modify the synchronization agreement

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -W -p
389 -h ipaserver.example.com <<EOF
 dn:
cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dco
m,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\3Dc
om,cn=mapping tree,cn=config"
```

The new subtree setting takes effect immediately. If a synchronization operation is currently running, then it takes effect as soon as the current operation completes.

6.5.4. Configuring Uni-directional Synchronization

By default, all modifications and deletions are bidirectional. A change in Active Directory is synchronized over to Identity Management, and a change to an entry in Identity Management is synchronized over to Active Directory. This is essentially an equitable, multi-master relationship, where both Active Directory and Identity Management are equal peers in synchronization and are both data masters.

However, there can be some data structure or IT designs where only one domain should be a data master and the other domain should accept updates. This changes the synchronization relationship from a multi-master relationship (where the peer servers are equal) to a master consumer relationship.

This is done by setting the ***oneWaySync*** parameter on the synchronization agreement. The possible values are ***fromWindows*** (for Active Directory to Identity Management synchronization) and ***toWindows*** (for Identity Management to Active Directory synchronization).

For example, to synchronize changes from Active Directory to Identity Management:

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password
-p 389 -h ipaserver.example.com

dn:
cn=meToWindowsBox.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=m
apping tree,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```

Important

Enabling unidirectional synchronization does *not* automatically prevent changes on the unsynchronized server, and this can lead to inconsistencies between the synchronization peers between synchronization updates. For example, unidirectional synchronization is configured to go from Active Directory to Identity Management, so Active Directory is (in essence) the data master. If an entry is modified or even deleted on the Identity Management, then the Identity Management information is different then the information and those changes are never carried over to Active Directory. During the next synchronization update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

6.5.5. Deleting Synchronization Agreements

Synchronization can be stopped by deleting the synchronization agreement which *disconnects* the IdM and Active Directory servers. In the inverse of creating a synchronization agreement, deleting a synchronization agreement uses the **`ipa-replica-manage disconnect`** command and then the host name of the Active Directory server.

1. Delete the synchronization agreement.

```
# ipa-replica-manage disconnect adserver.ad.example.com
```

2. List the certificates in the IdM directory certificate database:

```
# certutil -L -d /etc/dirsrv/slapd-IDM-EXAMPLE-COM/
Certificate Nickname                               Trust Attributes
                                                   SSL,S/MIME,JAR/XPI

IDM.EXAMPLE.COM IPA CA                             CT,C,C
CN=adserver,DC=ad,DC=example,DC=com                  C,,
Server-Cert                                        u,u,u
```

3. Remove the Active Directory CA certificate from the IdM server database:

```
# certutil -D -d /etc/dirsrv/slapd-IDM-EXAMPLE-COM/ -n
"CN=adserver,DC=ad,DC=example,DC=com"
```

6.5.6. Winsync Agreement Failures

Creating the synchronization agreement fails because it cannot connect to the Active Directory server.

One of the most common synchronization agreement failures is that the IdM server cannot connect to the Active Directory server:

```
"Update failed! Status: [81  - LDAP error: Can't contact LDAP server]
```

This can occur if the wrong Active Directory CA certificate was specified when the agreement was created. This creates duplicate certificates in the IdM LDAP database (in the **`/etc/dirsrv/slapd-DOMAIN/`** directory) with the name *Imported CA*. This can be checked using **`certutil`**:

```
$ certutil -L -d /etc/dirsrv/slapd-DOMAIN/

Certificate Nickname                                              Trust
Attributes
SSL,S/MIME,JAR/XPI

CA certificate                                                    CTu,u,Cu
Imported CA                                                       CT,,C
Server-Cert                                                       u,u,u
Imported CA                                                       CT,,C
```

To resolve this issue, remove the CA certificate from the certificate database:

```
# certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"
```

There are errors saying passwords are not being synchronized because it says the entry exists

For some entries in the user database, there may be an informational error message that the password is not being reset because the entry already exists:

```
"Windows PassSync entry exists, not resetting password"
```

This is not an error. This message occurs when an exempt user, the Password Synchronization user, is not being changed. The Password Synchronization user is the operational user which is used by the service to change the passwords in IdM.

6.6. Managing Password Synchronization

Synchronizing user entries is configured with the synchronization agreement. However, passwords in both Active Directory and Identity Management are not part of the normal user synchronization process. A separate client must be installed on the Active Directory servers to capture passwords as user accounts are created or passwords are changed, and then to forward that password information with the synchronized updates.

Note

The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.

Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

6.6.1. Setting up the Windows Server for Password Synchronization

Synchronizing passwords requires these things:

- Active Directory must be running in SSL.

  Note

  Install the Microsoft Certificate System in Enterprise Root Mode. Active Directory will then automatically enroll to retrieve its SSL server certificate.

- The Password Synchronization Service must be installed on *each* Active Directory domain controller. To synchronize a password from Windows, the PassSync service requires access to the unencrypted password to synchronize it over a secure connection to IdM. Because users can change their passwords on every domain controller, the installation of the PassSync service on each domain controller is necessary.

- The password policy must be set similar on IdM and Active Directory side. When the synchronization destination receives an updated password, it was only validated to match the policy on the source. It is not re-validated on the synchronization destination.

To verify that the Active Directory password complexity policy is enabled, run on an Active Directory domain controller:

```
> dsquery * -scope base -attr pwdProperties
    pwdProperties
    1
```

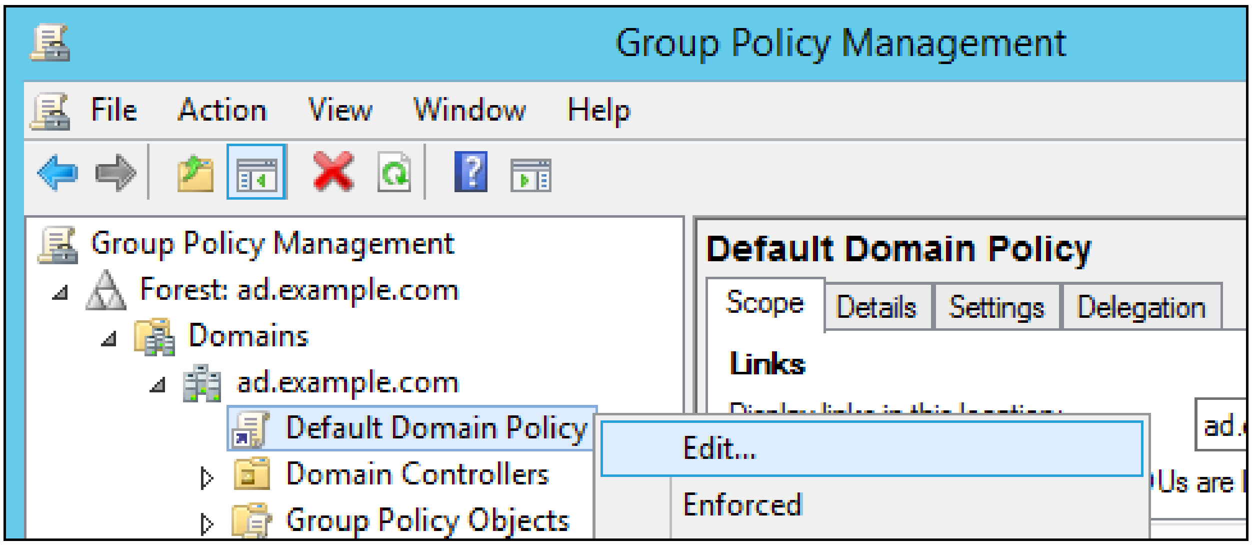
If the value of the attribute **`pwdProperties`** is set to **`1`**, the password complexity policy is enabled for the domain.

Note

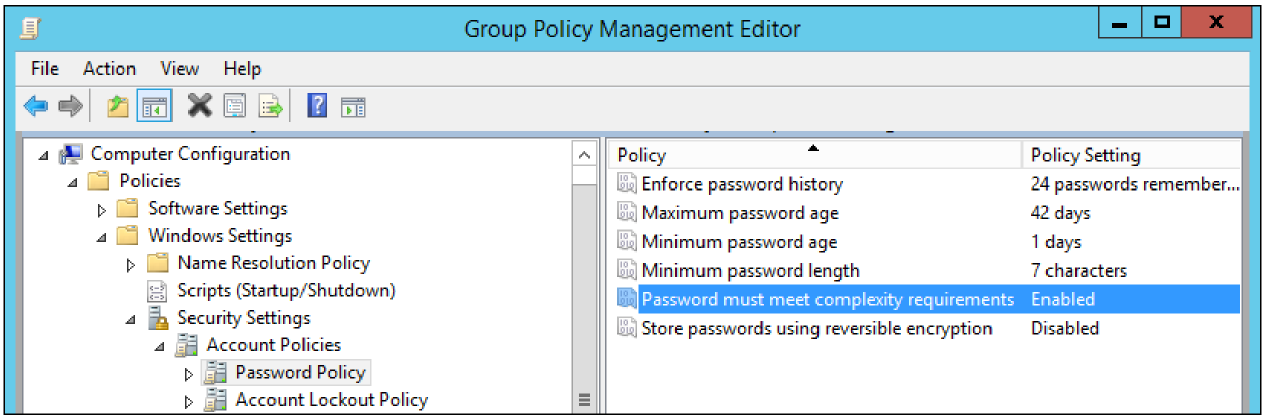
If you are unsure if group policies define deviating password settings for Organizational Units (ou), ask your group policy administrator.

To enable the Active Directory password complexity setting for the whole domain:

1. Run **`gpmc.msc`** from the command line.
2. Select **Group Policy Management**.
3. Open **Forest: *ad.example.com* → Domains → *ad.example.com***.
4. Right-click the **Default Domain Policy** entry and select **Edit**.



5. The **`Group Policy Management Editor`** opens automatically.
6. Open **Computer Configuration → Policies → Windows Settings → Security Settings → Account Policies → Password Policy**.
7. Enable the **`Password must meet complexity requirements`** option and save.



Computer Configuration
Policies
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Security Settings
Account Policies
Password Policy
Account Lockout Policy

Policy	Policy Setting
Enforce password history	24 passwords remember...
Maximum password age	42 days
Minimum password age	1 days
Minimum password length	7 characters
Password must meet complexity requirements	Enabled
Store passwords using reversible encryption	Disabled

6.6.2. Setting up Password Synchronization

Install the Password Synchronization Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the **`PassSync.msi`** file to the Active Directory machine.

   a. Log in to the Customer Portal.

   b. Click the **`Downloads`** link in the upper left corner.

   c. Select **`Red Hat Enterprise Linux`**.

   d. Select the most recent version of Red Hat Enterprise Linux 6 or Red Hat Enterprise Linux 7 and architecture.

   e. Download **`PassSync Installer`** by clicking on the 'Download Now' button.

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 Synchronization MSI file to install it.

3. The **Password Synchronrization Setup** window appears. Hit **Next** to begin installing.

4. Fill in the information to establish the connection to the IdM server.

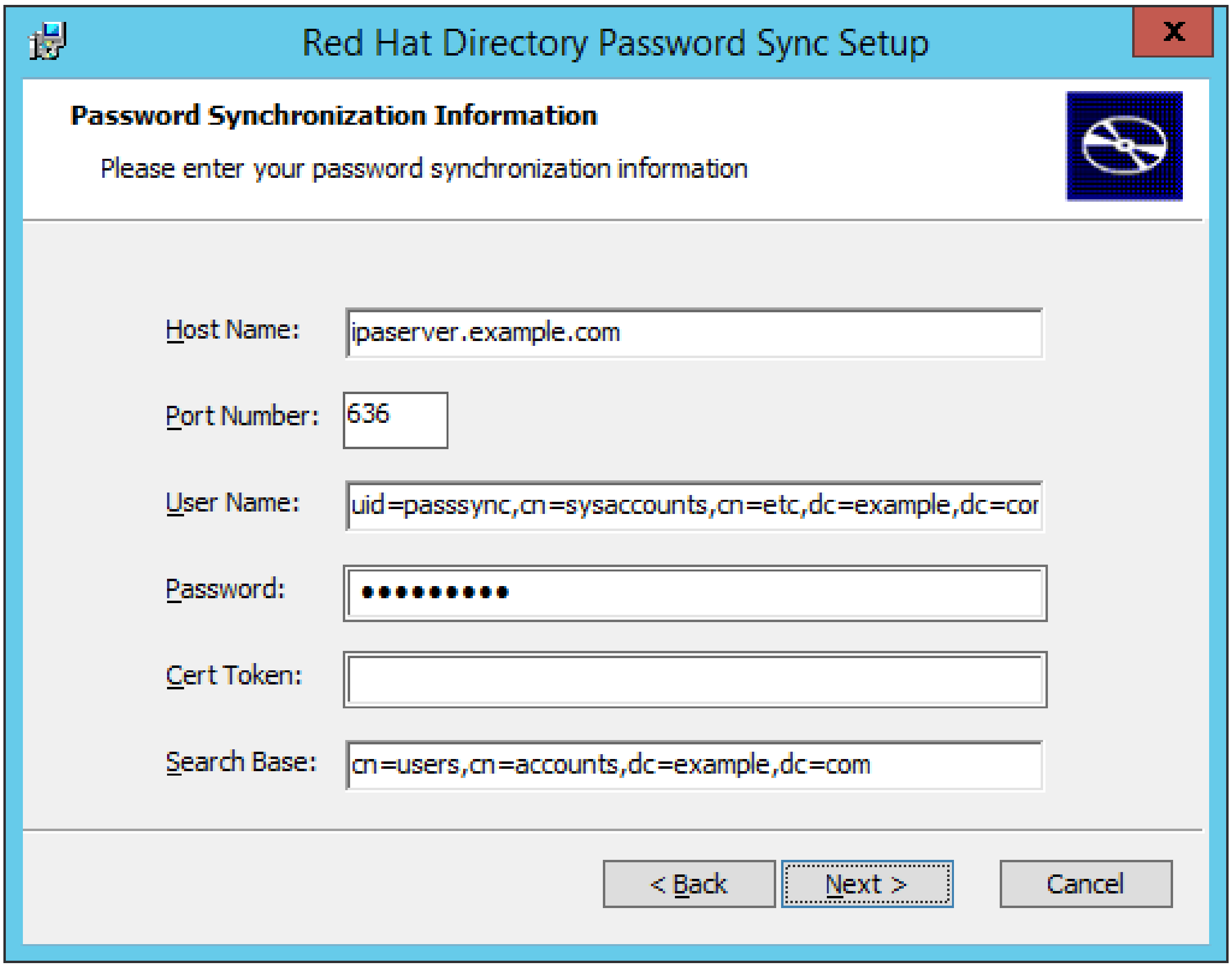
   - The IdM server connection information, including the host name and secure port number.

   - The user name of the system user which Active Directory uses to connect to the IdM machine. This account is configured automatically when synchronization is configured on the IdM server. The default account is **`uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com`**.

   - The password set in the **`--passsync`** option when the synchronization 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: ipaserver.example.com

Port Number: 636

User Name: uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=cor

Password: ●●●●●●●●●●

Cert Token:

Search Base: cn=users,cn=accounts,dc=example,dc=com

Hit **Next**, then **Finish** to install Password Synchronization.

5. Import the IdM server's CA certificate into the PassSync 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 Synchronization database. For example:

```
cd "C:\Program Files\Red Hat Directory Password
Synchronization"

certutil.exe -d . -A -n "IPASERVER.EXAMPLE.COM IPA CA" -t
CT,, -a -i ipaca.crt
```

6. Reboot the Windows machine to start Password Synchronization.

Note

The Windows machine must be rebooted. Without the rebooting, **PasswordHook.dll** is not enabled, and password synchronization will not function.

7. If passwords for existing accounts should be synchronized, reset the user passwords.

> **Note**
>
> The Password Synchronization client captures password changes and then synchronizes them between Active Directory and IdM. This means that it synchronizes new passwords or password updates.
>
> Existing passwords, which are stored in a hashed form in both IdM and Active Directory, cannot be decrypted or synchronized when the Password Synchronization client is installed, so existing passwords are not synchronized. User passwords must be changed to initiate synchronization between the peer servers.

The first attempt to synchronize passwords, which happened when the Password Synchronization application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory synchronization peers. The tools to create the certificate and key databases is installed with the **`.msi`**.

The password synchronization client cannot synchronize passwords for members of the IdM **`admin`** group. This is an intended behavior to prevent, for example, password synchronization agents or low level user administrators to change passwords of top level administrators.

> **Note**
>
> Passwords are only validated on the synchronization source to match the password policies. To verify and enable the Active Directory password complexity policy, see Section 6.6.1, "Setting up the Windows Server for Password Synchronization".

6.6.3. 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. Listed accounts in this attribute also bypass the password policy, and therefore no strength or history enforcement is applied.

> **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 DNs 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.

[5] The ***cn*** is treated differently than other synchronized attributes. It is mapped directly (***cn*** to ***cn***) when synchronizing from Identity Management to Active Directory. When synchronizing from Active Directory to Identity Management, however, ***cn*** is mapped from the ***name*** attribute on Windows to the ***cn*** attribute in Identity Management.

Chapter 7. Migrating Existing Environments from Synchronization to Trust

Synchronization and *trust* are two possible approaches to indirect integration. Synchronization is generally discouraged, and Red Hat recommends to use the approach based on Active Directory (AD) trust instead. See Section 1.3, “Indirect Integration” for details.

This chapter describes how to migrate an existing synchronization-based setup to AD trust. The following migrating options are available in IdM:

- Section 7.1, “Migrate from Synchronization to Trust Automatically Using **`ipa-winsync-migrate`**”
- Section 7.2, “Migrate from Synchronization to Trust Manually Using ID Views”

7.1. Migrate from Synchronization to Trust Automatically Using `ipa-winsync-migrate`

Important

The **`ipa-winsync-migrate`** utility is only available on systems running Red Hat Enterprise Linux 7.2 or later.

7.1.1. How Migration Using `ipa-winsync-migrate` Works

The **`ipa-winsync-migrate`** utility migrates all synchronized users from an AD forest, while preserving the existing configuration in the Winsync environment and transferring it into the AD trust. For each AD user created by the Winsync agreement, **`ipa-winsync-migrate`** creates an ID override in the Default Trust View (see Section 8.1, “Active Directory Default Trust View”).

After the migration completes:

- The ID overrides for the AD users have the following attributes copied from the original entry in Winsync:
  - Login name (**`uid`**)
  - UID number (**`uidnumber`**)
  - GID number (**`gidnumber`**)
  - Home directory (**`homedirectory`**)
  - GECOS entry (**`gecos`**)
- The user accounts in the AD trust keep their original configuration in IdM, which includes:
  - POSIX attributes
  - User groups
  - Role-based access control rules
  - Host-based access control rules

- SELinux membership
- **sudo** rules

- The new AD users are added as members of an external IdM group.
- The original Winsync replication agreement, the original synchronized user accounts, and all local copies of the user accounts are removed.

7.1.2. How to Migrate Using `ipa-winsync-migrate`

Before you begin:

- Back up your IdM setup using the **ipa-backup** utility. See Backing Up and Restoring Identity Management in the *Linux Domain Identity, Authentication, and Policy Guide*.

  Reason: The migration affects a significant part of the IdM configuration and many user accounts. Creating a backup enables you to restore your original setup if necessary.

To migrate:

1. Create a trust with the synchronized domain. See Chapter 5, *Creating Cross-forest Trusts with Active Directory and Identity Management*.
2. Run **ipa-winsync-migrate** and specify the AD realm name:

   ```
   # ipa-winsync-migrate --ad-realm AD.EXAMPLE.COM
   ```

   If a conflict occurs in the overrides created by **ipa-winsync-migrate**, information about the conflict is displayed, but the migration continues.

   See the ipa-winsync-migrate(1) man page for more details about the utility.

7.2. Migrate from Synchronization to Trust Manually Using ID Views

You can use ID views to manually change the POSIX attributes that AD previously generated for AD users.

1. Create a backup of the original synchronized user or group entries.
2. Create a trust with the synchronized domain. For information about creating trusts, see Chapter 5, *Creating Cross-forest Trusts with Active Directory and Identity Management*.
3. For every synchronized user or group, preserve the UID and GIDs generated by IdM by doing one of the following:

   - Individually create an ID view applied to the specific host and add user ID overrides to the view.
   - Create user ID overrides in the Default Trust View.

   For details, see Defining a Different Attribute Value for a User Account on Different Hosts.

Note

Only IdM users can manage ID views. AD users cannot.

4. Delete the original synchronized user or group entries.

For general information on using ID views in Active Directory environments, see Chapter 8, *Using ID Views in Active Directory Environments*.

Chapter 8. Using ID Views in Active Directory Environments

ID views enable you to specify new values for POSIX user or group attributes, as well as to define on which client host or hosts the new values will apply.

Integration systems other than Identity Management (IdM) sometimes generate UID and GID values based on an algorithm different than the algorithm used in IdM. By overriding the previously generated values to make them compliant with the values used in IdM, a client that used to be a member of another integration system can be fully integrated with IdM.

Note

This chapter only describes ID views functionality related to Active Directory (AD). For general information about ID views, see the Linux Domain Identity, Authentication, and Policy Guide.

You can use ID views in AD environments for the following purposes:

Overriding AD User Attributes, such as POSIX Attributes or SSH Login Details

See Section 8.3, “Using ID Views to Define AD User Attributes” for details.

Overriding smart card certificates for AD Users

See Section 8.4, “Overriding Smart Card Certificates for AD Users” for details.

Migrating from synchronization-based to trust-based integration

See Section 7.2, “Migrate from Synchronization to Trust Manually Using ID Views” for details.

Performing per-host group override of the IdM user attributes

See Section 8.5, “Migrating NIS Domains to IdM” for details.

8.1. Active Directory Default Trust View

The Default Trust View is the default ID view always applied to AD users and groups in trust-based setups. It is created automatically when you establish the trust using **`ipa-adtrust-install`** and cannot be deleted.

Using the Default Trust View, you can define custom POSIX attributes for AD users and groups, thus overriding the values defined in AD.

Table 8.1. Applying the Default Trust View

	Values in AD	Default Trust View		Result
Login	ad_user	ad_user	→	ad_user
UID	111	222	→	222
GID	111	(no value)	→	111

Note

The Default Trust View only accepts overrides for AD users and groups, not for IdM users and groups. It is applied on the IdM server and clients and therefore only need to provide overrides for Active Directory users and groups.

Overriding Default Trust View with Other ID Views

If another ID view applied to the host overrides the attribute values in the Default Trust View, IdM applies the values from the host-specific ID view on top of the Default Trust View.

- If an attribute is defined in the host-specific ID view, IdM applies the value from this view.
- If an attribute is not defined in the host-specific ID view, IdM applies the value from the Default Trust View.

The Default Trust View is always applied to IdM servers and replicas as well as to AD users and groups. You cannot assign a different ID view to them: they always apply the values from the Default Trust View.

Table 8.2. Applying a Host-Specific ID View on Top of the Default Trust View

	Values in AD	Default Trust View	Host-Specific View		Result
Login	ad_user	ad_user	(no value)	→	ad_user
UID	111	222	333	→	333
GID	111	(no value)	333	→	333

Default Trust View on Clients with Earlier Versions of IdM

Clients running IdM versions earlier than Red Hat Enterprise Linux 7.1 only see the Default Trust View, because ID views are applied on the client side. If you need a client to apply a different ID view, update SSSD on the client to a version with ID views support or have the client use the compat LDAP tree.

Note

The compat LDAP tree offers a simplified LDAP tree with user and group data for legacy clients.

8.2. Fixing ID Conflicts

IdM uses *ID ranges* to avoid collisions of POSIX IDs from different domains. For details on ID ranges, see ID Ranges in the *Linux Domain Identity, Authentication, and Policy Guide*.

POSIX IDs in ID views do not use a special range type, because IdM must allow overlaps with other kinds of ID ranges. For example, AD users created through synchronization have POSIX IDs from the same ID range as IdM users.

POSIX IDs are managed manually in ID views on the IdM side. Therefore, if an ID collision occurs, fix it by changing the conflicting IDs.

8.3. Using ID Views to Define AD User Attributes

With ID views, you can change the user attribute values defined in AD. For a complete list of the attributes, see Attributes an ID View Can Override.

For example: If you are managing a mixed Linux-Windows environment and want to manually define POSIX attributes or SSH login attributes for an AD user, but the AD policy does not allow it, you can use ID views to override the attribute values. When the AD user authenticates to clients running SSSD or authenticates using a compat LDAP tree, the new values are used in the authentication process.

Note

Only IdM users can manage ID views. AD users cannot.

The process for overriding the attribute values follows these steps:

1. Create a new ID view.
2. Add a user ID override in the ID view, and specify the require attribute value.
3. Apply the ID view to a specific host.

For details on how to perform these steps, see Defining a Different Attribute Value for a User Account on Different Hosts in the *Linux Domain Identity, Authentication, and Policy Guide*.

8.4. Overriding Smart Card Certificates for AD Users

In a trusted environment, the administrator wants to store the smart card certificates for Active Directory users in IdM. The administrator uses ID views to override the attribute values from Active Directory.

The process for overriding the **`userCertificate`** attribute values follows these steps:

1. Use the automatically created **`Default Trust View`** that is applied to all IdM clients, or add a new, custom ID view and apply it to one or more hosts.
2. Add a user ID override in the ID view, and specify the certificate to override.

For details on how to perform these steps, see Defining a Different Attribute Value for a User Account on Different Hosts in the *Linux Domain Identity, Authentication, and Policy Guide*.

8.5. Migrating NIS Domains to IdM

If you are managing a Linux environment and want to migrate disparate NIS domains with different UIDs and GIDs into a modern identity management solution, you can use ID views to set host specific UIDs and GIDs for existing hosts to prevent changing the permissions on existing files and directories.

The process for the migration follows these steps:

1. Create the users and groups in the IdM domain. For details, see
   - Adding Stage or Active Users
   - Creating User Groups
2. Use ID views for existing hosts to override the IDs IdM generated during the user creation:
   a. Create an individual ID view.
   b. Add ID overrides for the users and groups to the ID view.
   c. Assign the ID view to the specific hosts.

   For details, see Defining a Different Attribute Value for a User Account on Different Hosts.
3. Configure the Linux System as an IdM Client.
4. Decommission the NIS domains.

Appendix A. Revision History

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Enterprise Linux.

Revision 7.0-26	**Wed Nov 23 2016**	**Aneta Šteflová Petrová**	Added ipa-winsync-migrate. Minor fixes for the trust, SSSD, and synchronization chapters.
Revision 7.0-25	**Tue Oct 18 2016**	**Aneta Šteflová Petrová**	Version for 7.3 GA publication.
Revision 7.0-24	**Thu Jul 28 2016**	**Marc Muehlfeld**	Updated diagrams, added Kerberos flags for services and hosts, other minor fixes.
Revision 7.0-23	**Thu Jun 09 2016**	**Marc Muehlfeld**	Updated the synchronization chapter. Removed the Kerberos chapter. Other minor fixes.
Revision 7.0-22	**Tue Feb 09 2016**	**Aneta Petrová**	Updated realmd, removed index, moved a part of ID views to the Linux Domain Identity guide, other minor updates.
Revision 7.0-21	**Fri Nov 13 2015**	**Aneta Petrová**	Version for 7.2 GA release with minor updates.
Revision 7.0-20	**Thu Nov 12 2015**	**Aneta Petrová**	Version for 7.2 GA release.
Revision 7.0-19	**Fri Sep 18 2015**	**Tomáš Čapek**	Updated the splash page sort order.
Revision 7.0-18	**Thu Sep 10 2015**	**Aneta Petrová**	Updated the output format.
Revision 7.0-17	**Mon Jul 27 2015**	**Aneta Petrová**	Added GPO-based access control, a number of other minor changes.
Revision 7.0-16	**Thu Apr 02 2015**	**Tomáš Čapek**	Added ipa-advise, extended CIFS share with SSSD, admonition for the Identity Management for UNI extension.
Revision 7.0-15	**Fri Mar 13 2015**	**Tomáš Čapek**	Async update with last-minute edits for 7.1.
Revision 7.0-13	**Wed Feb 25 2015**	**Tomáš Čapek**	Version for 7.1 GA release.
Revision 7.0-11	**Fri Dec 05 2014**	**Tomáš Čapek**	Rebuild to update the sort order on the splash page.
Revision 7.0-7	**Mon Sep 15 2014**	**Tomáš Čapek**	

Section 5.3 Creating Trusts temporarily removed for content updates.

Revision 7.0-5	**June 27, 2014**	**Ella Deon Ballard**

Improving Samba+Kerberos+Winbind chapters.

Revision 7.0-4	**June 13, 2014**	**Ella Deon Ballard**

Adding Kerberos realm chapter.

Revision 7.0-3	**June 11, 2014**	**Ella Deon Ballard**

Initial release.