Red Hat Enterprise Linux 7
Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Ella Deon Ballard

Tomáš Čapek

Aneta Petrová

Red Hat Enterprise Linux 7 Windows Integration Guide

Integrating Linux Systems with Active Directory Environments

Ella Deon Ballard
Red Hat Customer Content Services
dlackey@redhat.com

Tomáš Čapek
Red Hat Customer Content Services
tcapek@redhat.com

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

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 light-weight AD pass-through authentication to full-fledged Kerberos trusted realms.

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 username/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 and/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 a user synchronization capability, also available in IdM or Red Hat Directory Server (RHDS). The synchronization allows user accounts (and with RHDS also group accounts) to be synchronized from AD to IdM or RHDS. However, this approach has a set of 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 HBAC (Host-Based Access Control) in IdM and GPO (Group Policy Object) 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 benefitial 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 the following command:

```
# 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 mini-searches, each returning a subset of the results within a given range, until all matches are returned.

For example, when doing a search for the ***`member`*** attribute, each entry could have multiple values, and there can be multiple entries with that attribute. If there are 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 adserver.example.com
   nameserver 198.68.72.1
   ```

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. 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.

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

   a. Open the Samba configuration file.

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

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

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

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

   a. Obtain Kerberos credentials for a Windows administrative user.

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

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

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

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

   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@rhel-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 to 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, *ID Views and Migrating Existing Environments to Trust*.

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`** checkbox, and then click **`OK`**.
3. On the Linux client, add the Active Directory domain to the client's DNS configuration so that it can resolve the domain's SRV records.

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

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

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

      a. Open the Kerberos client configuration file.

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

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

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

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

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

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

      a. Open the Samba configuration file.

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

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

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

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

      a. Obtain Kerberos credentials for a Windows administrative user.

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

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

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

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

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

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

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

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

5. Install the **sssd-ad** package.

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

6. Start the SSSD service.

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

7. Open the SSSD configuration file.

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

8. Configure the Active Directory domain.

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

   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
```

9. 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
```

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

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

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

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

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

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

13. 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 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=*))
```

[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. It does not connect to the domain itself but configures underlying Linux system services, such as SSSD or Winbind, to connect to the domain.

3.1. About `realmd`

Chapter 2, *Using Active Directory as an Identity Provider for SSSD* describes how to use the System Security Services Daemon on a local system and Active Directory as a back-end identity provider. There are a number of different configuration parameters for each possible identity provider and for SSSD itself. All domain information must be available in advance and then properly formatted in the SSSD configuration for SSSD to integrate the local system with Active Directory. That can be a complex task and **`realmd`** simplifies that configuration; it can run a service discovery to identify available Active Directory and Red Hat Enterprise Linux Identity Management domains, and then join the domain and manage user access. SSSD as an underlying service supports multiple domains and **`realmd`** can therefore discover and support multiple domains as well.

3.1.1. Types of Domains

The **`realmd`** system can discover the following types of identity domains:

- Microsoft Active Directory
- Red Hat Enterprise Linux Identity Management

You can use **`realmd`** to join an Active Directory or Identity Management domain; **`realmd`** properly configures the required system configuration files and services.

3.1.2. Supported Domain Clients

The **`realmd`** system automatically configures the required client services that are used to connect to the given identity realm. There are two supported clients:

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

3.2. `realmd` Commands

The **`realmd`** system has two major task areas: managing system enrollment in a domain and setting which domain users are allowed to access the local system resources. The central utility in **`realmd`** is called **`realm`** – this utility mostly specifies an action and the realm for which to perform that action.

```
realm command arguments
```

For example:

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

Table 3.1. realmd Commands

Command	Description
Realm Commands	
discover	Run a discovery scan for domains on the network.
join	Add the system to the specified domain.
leave	Removes the system from the specified domain.
list	Lists all configured realms for the system or all discovered and configured realms.
Login Commands	
permit	Enables access for specified users or for all users within a configured realm to access the local system.
deny	Restricts access for specified users or for all users within a configured realm to access the local system.

3.3. Discovering and Joining Active Directory Domains

3.3.1. Discovering Domains

The discovery process is handled by the **`discover`** command. It returns complete realm configuration and a list of packages that must be installed for the system to be enrolled in the realm.

Note

Note that 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, specify the realm name in the command, for example, **`realm discover ad.example.com`**.

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

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

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

It is also possible to run a discovery for a specific domain, using the domain controller's host name or IP address:

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

3.3.2. Joining an Active Directory Domain

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

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

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

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

If Kerberos is properly configured on a Linux system, the join operation can also be performed with a Kerberos ticket for authentication. The **realmd** system can use the **-U** option to select, which principal to use, or can use the default credential cache or the **KRB5_CCACHE** variable.

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

During joining, the **realmd** system checks for the DNS SRV record:

```
_ldap._tcp.domain.example.com. // for IdM records
_ldap._tcp.dc._msdcs.domain.example.com. // for Active Directory records
```

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

The actual join command configures both the local system services and the entries in the Active Directory domain by performing these steps:

1. Runs a discovery scan for the specified realm.
2. Installs any required packages to join the system to the domain. This includes SSSD and the PAM home directory job packages. 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. You will be required to install them manually using the **yum** utility.

3. Attempts to join the Active Directory domain as the administrator unless a different user is specified with the **-U** option. The command first attempts to connect without credentials, but it prompts for a password if required.

Note

For Active Directory, the administrator account is called **Administrator**, for IdM, it is called **admin**.

4. Once it connects to the domain, it creates an account entry for the system in the directory.

5. Creates the **/etc/krb5.keytab** host keytab file.

6. Configures the domain in SSSD and restarts the service.

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

One of the attributes returned in the discovery search is **login-policy**, which shows if domain users are allowed to log in as soon as the join is complete. If logins are not allowed by default, then they can be manually allowed by the **permit** command. For details, see Section 3.4, “Managing User Logins from Active Directory”.

3.3.3. Removing a System from the Active Directory Domain

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

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

This command performs the removal as the default administrator account (admin in Identity Management; Administrator in Active Directory). The script may prompt for a password or, depending on how the system was joined to the domain, it may require performing the operation as a different user. A user can be specified with the **-U** option.

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

Note

When a client leaves a domain, the computer object is not deleted; the local client is only deconfigured. If you want to delete it, run the command with the **--remove** option specified.

3.3.4. Listing Domains

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

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

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

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

3.4. Managing User Logins from Active Directory

By default, login policies for domain users are defined in the domain itself. This can be overridden in the realm configuration so that client-side access control is used – that is, the local policies only define who is allowed to login. If a machine is joined to multiple domains, only one of them can apply domain access control; the other domains have to employ client-side access control.

The **`realmd`** command allows you to configure basic allow or deny access rules for users from a specific domain. You can specify these permissions only if you are applying the client-side access control.

Note

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

There are two commands that set access rules:

- The **`realm deny`** command simply prevents access to all users within the realm. Use this command with the **`--all`** option.
- The **`realm permit`** command, on the other hand, grants access to either all users by using **`--all`**, to only specified users, or it can withdraw permission from specified users by using **`-x`**.

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

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

Important

It is recommended that you use **`permit`** to allow access instead of **`permit -x`** to deny it. It is much safer to allow access to specifically selected users or groups than to deny access to some, thus enabling it to everyone else.

Because SSSD is currently not capable to inform **`realmd`** about available subdomains and user logins must contain the domain name, allowing access works for users in primary domains but not for users in trusted domains.

3.5. Adding Default User Configuration

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

- The user home directory
- A default user shell

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

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

For example:

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

3.6. Additional Configuration for the Active Directory Domain Entry

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

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

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

Table 3.2. Realm Configuration Options

Parameter	Description
computer-ou	Sets the directory location 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.

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

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

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

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

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

Chapter 4. Using Samba, Kerberos, and Winbind

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. Intergration 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 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 look-ups,
- 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 Windbind 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 (via 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 (e.g., *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@se 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@se yum install krb5-workstation
   ```

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

   ```
   [root@se vim /etc/krb5.conf

   [libdefaults]
   ...

     default_realm PLE.COM
   ```

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

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

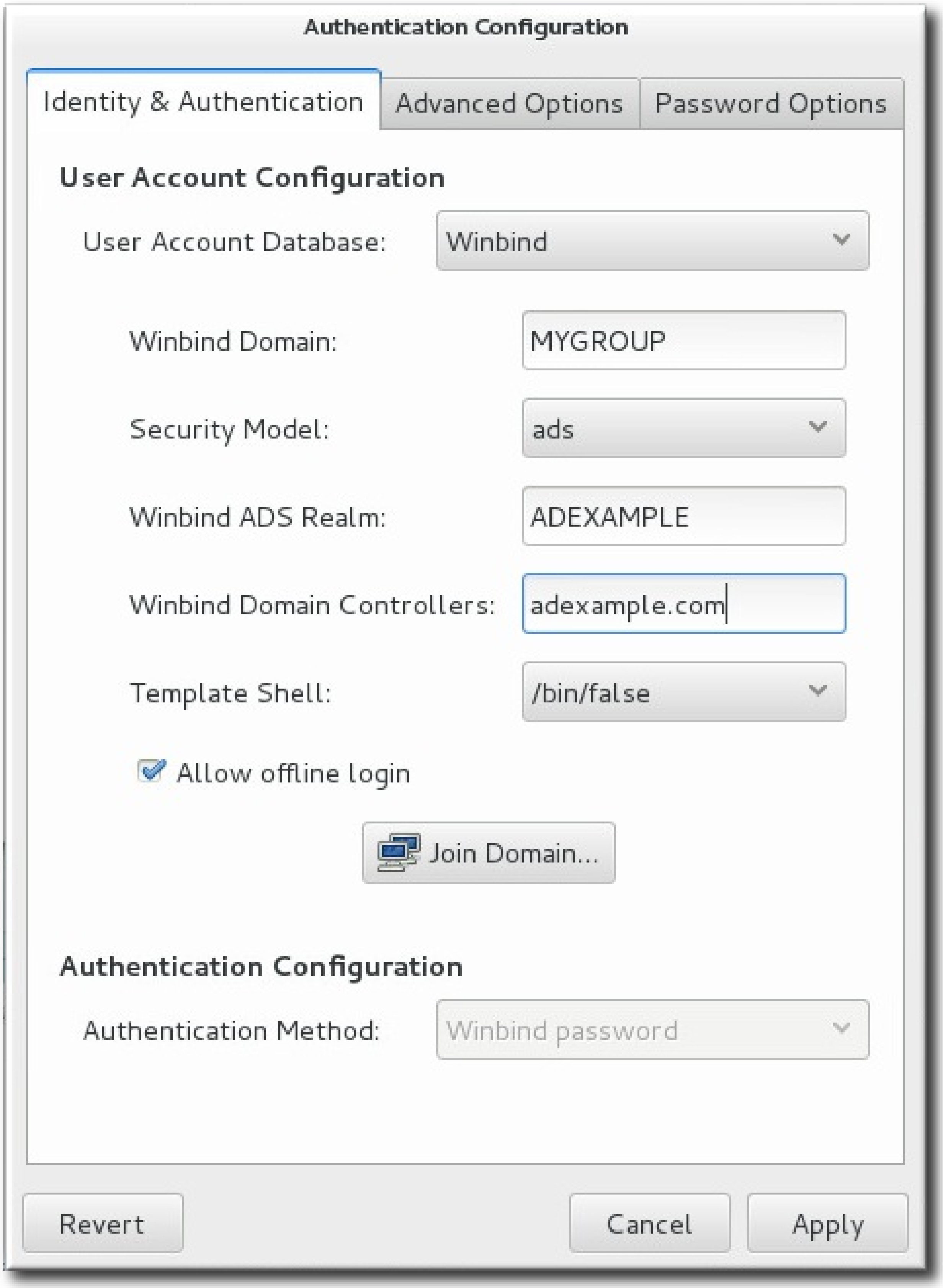
4. Edit the DNS configuration in the **`/etc/resolv.conf`** file to use the Active Directory domain as a name server and for search:

   ```
   nameserver 1.2.3
   search adexample
   ```

5. Open the Authentication Configuration Tool.

   ```
   [root2se authconfig-gtk
   ```

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


Authentication Configuration

Identity & Authentication | Advanced Options | Password Options

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

Join Domain...

Authentication Configuration

Authentication Method: Winbind password

Revert | Cancel | Apply

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 inthis 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-Realm Trusts with Active Directory and Identity Management

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

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

5.1. Introduction to Trusts

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

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

5.1.1. The Architecture of a Trust Relationship

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

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

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

On a basic level, a trust flows only in one direction. The *trusted domain* contains users and the *trusting domain* allows access to resources. In a trust, 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, “Basic Unidirectional Trust”, Realm A is trusted by Realm B, but Realm B is not trusted by Realm A; the trust is unidirectional.

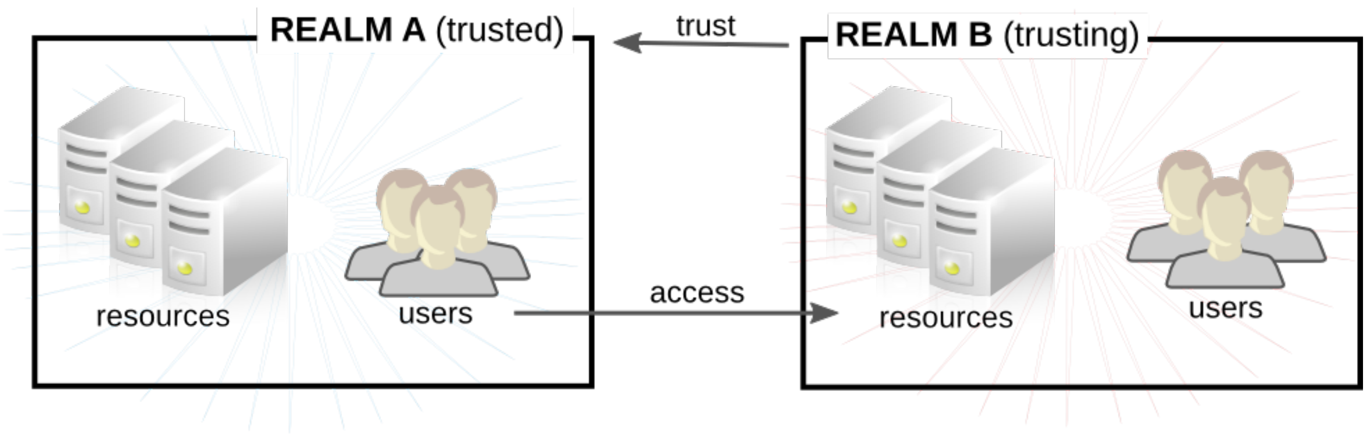


Figure 5.1. Basic Unidirectional Trust

Trusts can be *transitive* so that a domain trusts another domain and any other domain trusted by that second domain. Trusts can also be *non-transitive* which means the trust is limited only to the explicitly included domains.

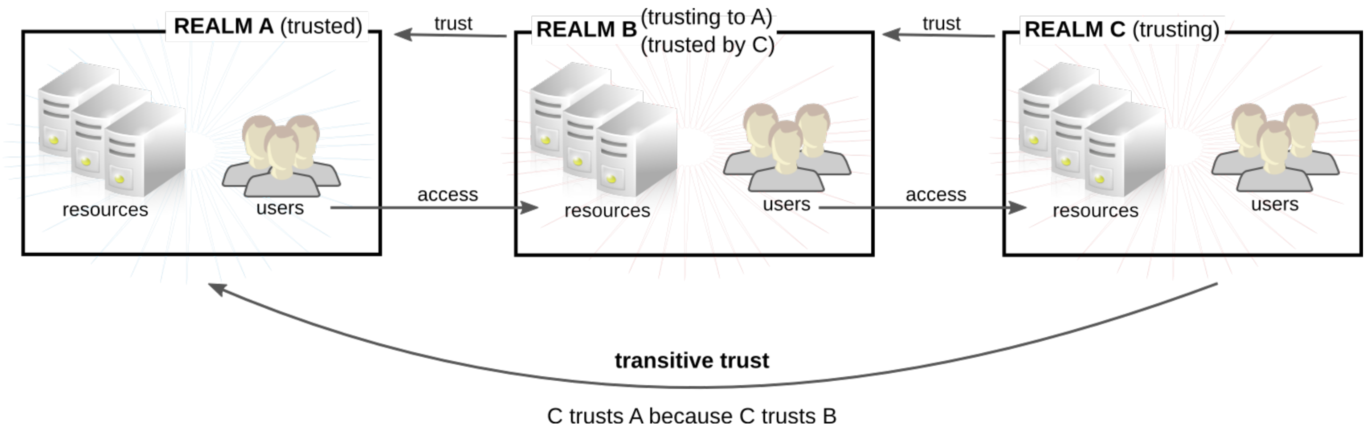


Figure 5.2. Transitive Trusts

Within a homogeneous Active Directory environment, most trusts are bidirectional and transitive by default. However, in a heterogeneous environment, Active Directory can establish a trust with a non-Active Directory domain, such as a Kerberos realm. This is a realm trust and it has a slightly different default configuration than an Active Directory domain trust:

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

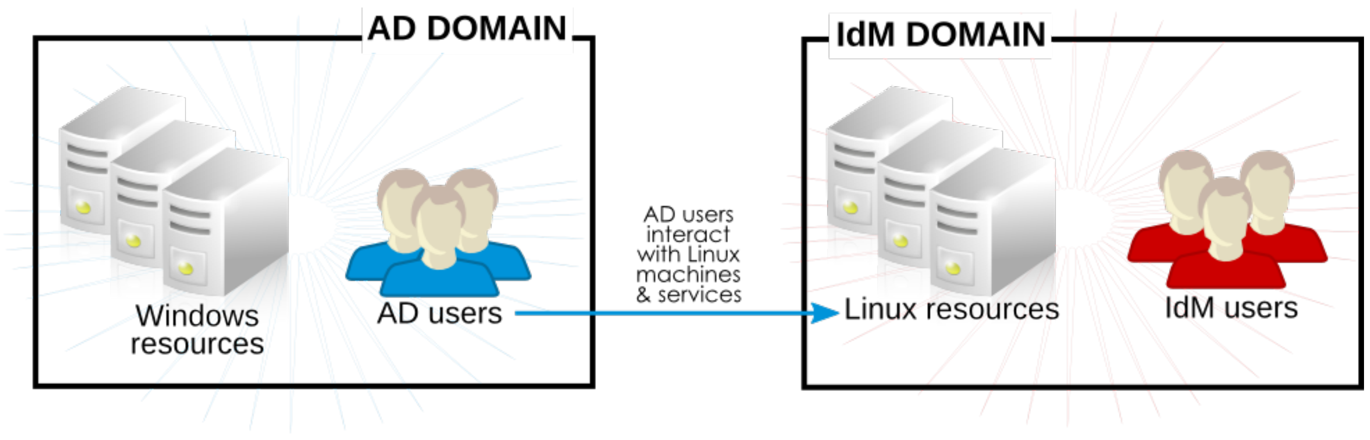


Figure 5.3. Trust Direction

5.1.2. Active Directory Principles, Security Objects, and Trust

In a trust, users access resources in external domains. A *trust path* is a sequence of domains through which secure communication flows in each individual case of getting access. Active Directory trusts use NTLM (NT LAN Manager) to evaluate identities. Realm trusts – including Identity Management trusts – use Kerberos to create, send, and verify privileged access certificates and create Kerberos tickets for external applications.

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

All Kerberos communication for both Active Directory and IdM for trusts uses GSS-API. Past the local security authority, there is the larger Active Directory configuration. The domain system container holds all security information, including principles, user, and group information for all objects in a domain. At the root of the domain is a global catalog of all users, groups, and objects within the entire forest. With a trust, information about Windows users can be retrieved from the system container or from the global catalog.

IdM can also be part of trust relationships with different Active Directory 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 a different unrelated Active Directory forest access to resources in the same shared IdM domain.

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.

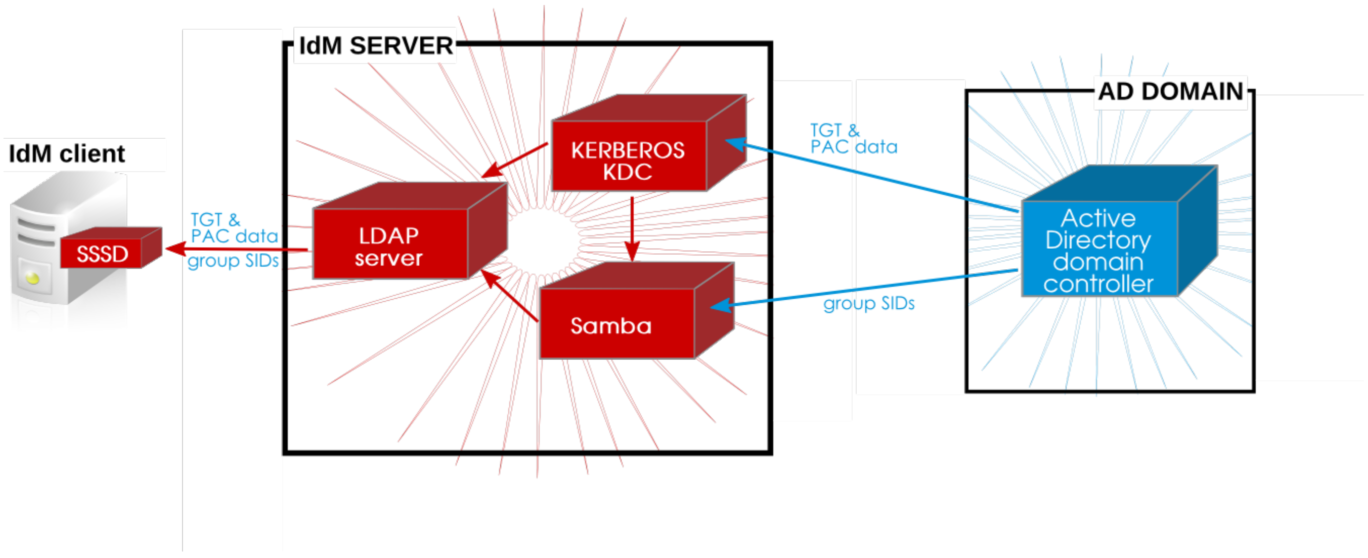


Figure 5.4. Applications and Services for Trust

5.1.3.1. Active Directory PACs and IdM Tickets

Group information in Active Directory is stored in a list of identifiers in each Kerberos ticket for Active Directory users in a special data set called *privileged access certificates*, or MS-PAC. The group information in the PAC has to be mapped to the Active Directory groups and then to the corresponding IdM groups to help determine access. A PAC is essentially an account usability extension an it is embedded in a Kerberos ticket as a way of identifying the entity to other Windows clients and servers in the Windows domain.

On IdM resources, if an Active Directory user requests a ticket for a service, then IdM forwards the request to Active Directory to retrieve the user information. The PAC information sent back by Active Directory is embedded in the Kerberos ticket.

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

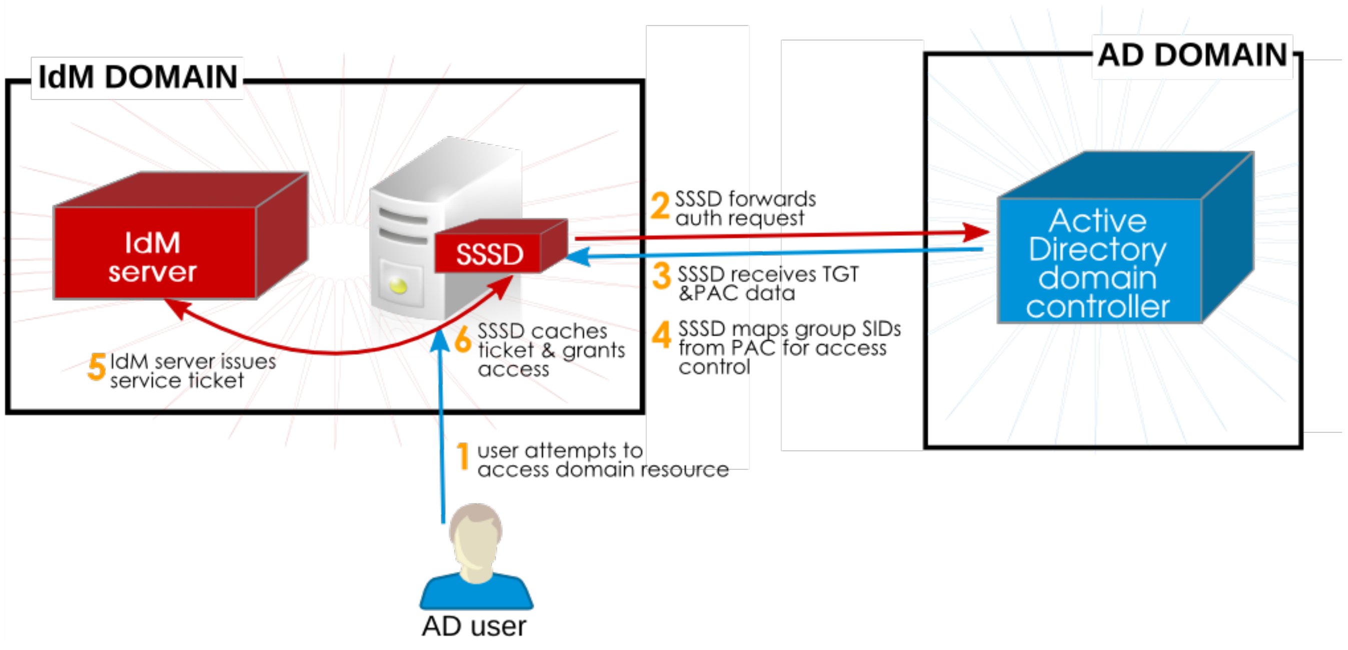

IdM DOMAIN
AD DOMAIN
IdM server
SSSD
Active Directory domain controller
2 SSSD forwards auth request
3 SSSD receives TGT &PAC data
4 SSSD maps group SIDs from PAC for access control
6 SSSD caches ticket & grants access
5 IdM server issues service ticket
1 user attempts to access domain resource
AD user

Figure 5.5. IdM, SSSD, and Active Directory

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

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

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

5.1.3.2. Active Directory Users and IdM Groups

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

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

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

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

Actually configuring groups for Active Directory users is described in Section 5.3, “Creating IdM Groups for Active Directory Users”.

5.1.3.3. Mapping User IDs and Using POSIX Attributes

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

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

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

NOTE

To use Active Directory-defined POSIX attributes, those attributes must be published to the global catalog. Otherwise, AD users with POSIX attributes cannot utilize IdM resources.

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

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

Table 5.1. Types of Ranges

Range Option	Description
ipa-ad-trust	For IDs set with SID - username mapping.
ipa-ad-trust-posix	For IDs defined in POSIX attributes in the Active Directory entry.

5.1.3.4. Active Directory Users and IdM Policies and Configuration

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

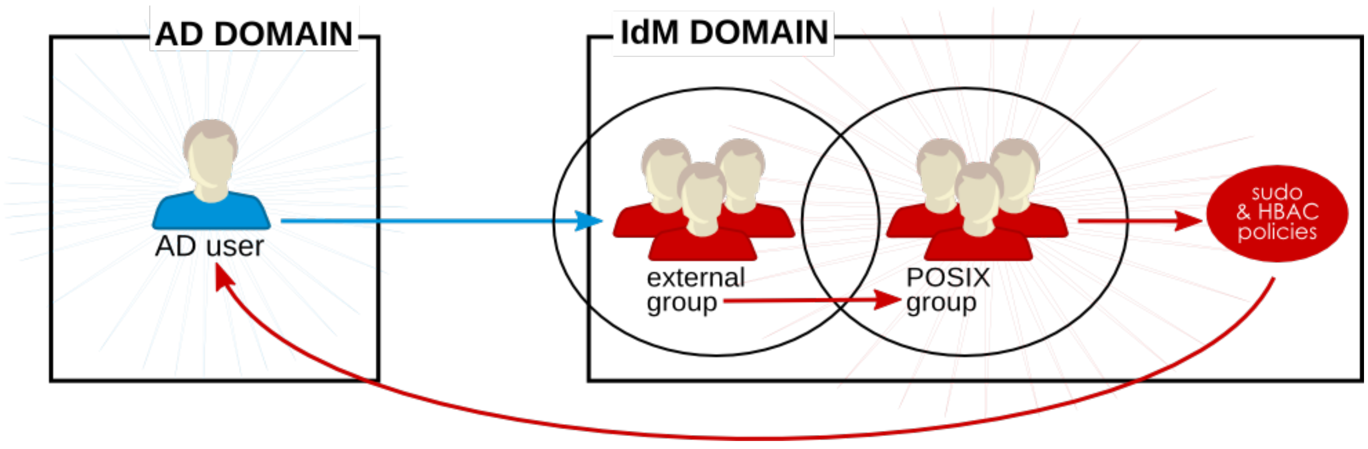


Figure 5.6. Active Directory Users and IdM Groups and Policies

Active Directory users are external to the IdM domain, but they can still be added as group members to IdM groups, as long as those groups are configured as external groups. External groups in IdM are non-POSIX groups. The external group is then added as a member of an IdM group (a POSIX group).

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

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

NOTE

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

5.1.4. Different DNS-Trust Environments

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

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

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

IMPORTANT

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

Required: Identical Records in Superior and Subdomains

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

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

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

Configuration Option 1: Separate DNS Domains

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

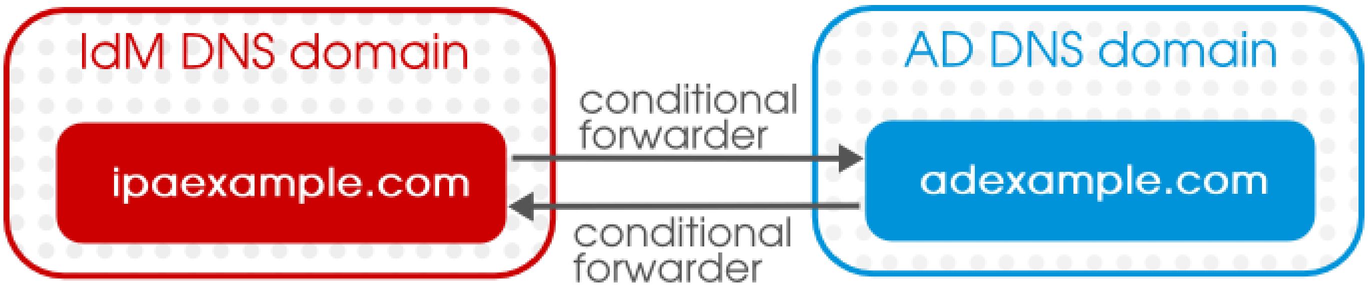


Figure 5.7. Trust with Separate DNS Domains

Configuration Option 2: Separate DNS Subdomains

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

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

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

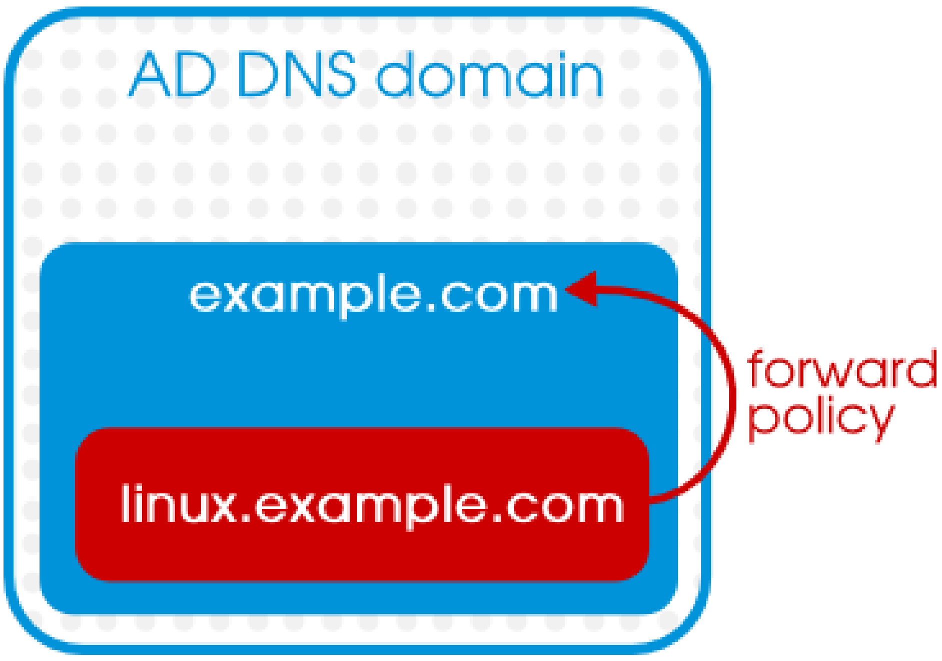


Figure 5.8. Trust with IdM as a DNS Subdomain of Active Directory

5.1.5. Identity Management Interactions with Multiple Domains

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

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

5.1.6. Potential Behavior Issues with Active Directory Trust

5.1.6.1. Active Directory Users and IdM Administration

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

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

5.1.6.2. Authenticating Deleted Active Directory Users

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

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

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

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

5.1.6.3. Credential Cache and Selecting Active Directory Principals

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

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

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

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

Valid starting       Expires              Service principal
27.11.2013 11:25:23  27.11.2013 21:25:23
krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
 renew until 28.11.2013 11:25:16
```

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

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

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

Valid starting       Expires              Service principal
27.11.2013 11:25:23  27.11.2013 21:25:23
krbtgt/ADEXAMPLE.ADREALM@ADEXAMPLE.ADREALM
 renew until 28.11.2013 11:25:16

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

```
Valid starting       Expires              Service principal
27.11.2013 11:25:18  28.11.2013 11:25:16  krbtgt/EXAMPLE.COM@EXAMPLE.COM
27.11.2013 11:25:48 28.11.2013 11:25:16
host/ipaclient.example.com@EXAMPLE.COM >>>>> host principal
```

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

5.1.6.4. Resolving Group SIDs

Losing Kerberos Tickets

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

Cannot Verify Group Membership for Users

There is no way to verify that a specific trusted user is associated with a specific IdM group, external or POSIX.

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

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

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

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

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

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

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

TIP

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

```
[root@ipaserver ~]# id ADDOMAIN\jsmith
uid=1921801107(jsmith@adexample.com)
gid=1921801107(jsmith@adexample.com)
groups=1921801107(jsmith@adexample.com),129600004(ad_users),192180051
3(domain users@adexample.com)
```

5.2. Environment and Machine Requirements to Set up Trusts

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

5.2.1. Supported Windows Platforms

Trust relationships can be configured with these Windows server versions:

- Windows Server 2008 R2
- Windows Server 2012 R2

5.2.2. Domain and Realm Names

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

5.2.3. NetBIOS Names

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

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

5.2.4. Integrated DNS

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

5.2.5. Integrated Certificate Authorities

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

5.2.6. Firewalls and Ports

For a trust relationship, the Active Directory server and IdM server must have the system ports required for an IdM server installation open.

The IdM backend LDAP server *must not be reachable* by any Active Directory domain controller. Make sure the associated ports (389 and 636) on the IdM server host are shut down for the Active Directory domain controller. Note that the 389 and 636 ports must be shut down only for the domain controllers; otherwise, they are required to remain open.

For a list of the ports required by IdM, see the corresponding chapter in the Linux Domain Identity, Authentication, and Policy Guide.

The ports required for a trust relationship to work are listed in Table 5.2, “Ports Required for a Trust”.

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

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

The **firewalld** configuration is required to allow access to the necessary IdM ports and, for each Active Directory host, to reject access to the IdM LDAP ports. To set the **firewalld** configuration in this way, assuming the used **firewalld** zone is **public**:

1. Add the rules to restrict access to LDAP ports for each Active Directory host.

   ```
   [root@server ~]# firewall-cmd --permanent --zone=public --add-rich-rule='rule family="ipv4" source address="ad_ip_address" service name="ldap" reject'
    [root@server ~]# firewall-cmd --permanent --zone=public --add-rich-rule='rule family="ipv4" source address="ad_ip_address" service name="ldaps" reject'
   ```

2. Open the ports to services required by IdM.

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

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

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

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

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

Table 5.2. Ports Required for a Trust

Service	Ports	Type
NetBIOS-DGM	138	TCP and UDP
NetBIOS-SSN	139	TCP and UDP
LDAP	389	UDP
Microsoft-DS	445	TCP and UDP

5.2.7. IPv6 Settings

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

5.2.8. Clock Settings

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

5.2.9. Supported Username Formats

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

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

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

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

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

This supports usernames in multiple formats:

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

TIP

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

5.3. Creating IdM Groups for Active Directory Users

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

As described in Section 5.1.2, “Active Directory Principles, Security Objects, and Trust”, Active Directory users are added to the IdM domain in a kind of daisy chain. They are to an IdM external group (meaning, a non-POSIX group), and then that external group is added to a local

POSIX group as a member. The IdM POSIX group can then be used for user/role management of Active Directory users.

TIP

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

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

2. Create an *external* group in the IdM domain for the Active Directory users. Using the **--external** argument indicates that this group will contain members from outside the IdM domain. For example:

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

3. Create the *POSIX* group for actually administering the IdM policies.

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

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

   For example, for an Active Directory group:

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

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

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

5.4. Maintaining Trusts

There are several layers to the trust configuration. There is the immediate trust agreement with IdM and its peer Active Directory. There is also a substantial backend configuration in IdM. When IdM is configured to support trusts, it creates a number of different kinds of configuration areas:

- Global trust configuration that is used to identify IdM within Windows domains (such as a SID)
- Identified DNS domains in Active Directory, which are pulled into the IdM DNS zone configuration (realm domains)
- The Kerberos trust configuration (the individual trust agreements, in trust domains)
- Assigned available ID ranges, per IdM server, to use to assign UID and GID numbers to Windows users as they enter the IdM domain

5.4.1. Editing the Global Trust Configuration

When the **`ipa-adtrust-install`** command is run, it automatically configures background information for the IdM domain which is required to create a trust with the Active Directory domain. Even for external trusts, the Active Directory domain assumes that its trusted peer has certain configuration attributes, such as a security ID and domain ID. The attributes are created for the IdM server, so that it is Active Directory-compliant.

The global trust configuration contains five attributes:

- A Windows-style security ID
- A domain GUID
- A Kerberos domain name
- The default group to which to add Windows users

Only some of those attributes (the NetBIOS name and default group) can be edited. The GUID and SID are autogenerated, and the Kerberos realm name is from the IdM configuration.

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

5.4.1.1. Changing the NetBIOS Name

The NetBIOS name is the far-left component of the domain name; this is a key identifier for the host system of a domain controller. When IdM is enabled for trust, a NetBIOS name is set for the IdM server so that it is compatible within an Active Directory topology. This is configured in the **`ipa-adtrust-install`** command. To change it, rerun the **`ipa-adtrust-install`** command.

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

The **`-a`** option gives the IdM administrator password.

5.4.1.2. Changing the Default Group for Windows Users

IdM has a feature, *automembership*, which adds new users to specific groups automatically. There is a default automembership rule to add Windows users automatically to the *Default SMB Group* (a group created as part of the IdM trust configuration). This is a fallback group used if no other automembership rules apply to the Windows users.

The default group can be changed, which can be particularly useful if there are different external groups added for Windows users (Section 5.3, “Creating IdM Groups for Active Directory Users”). **This group is a fallback or default group which is used globally for all Windows users; other rules can be set to apply specific groups to different Windows users, rather than using the default.**

The default group can be set using the **`trustconfig-mod`** command:

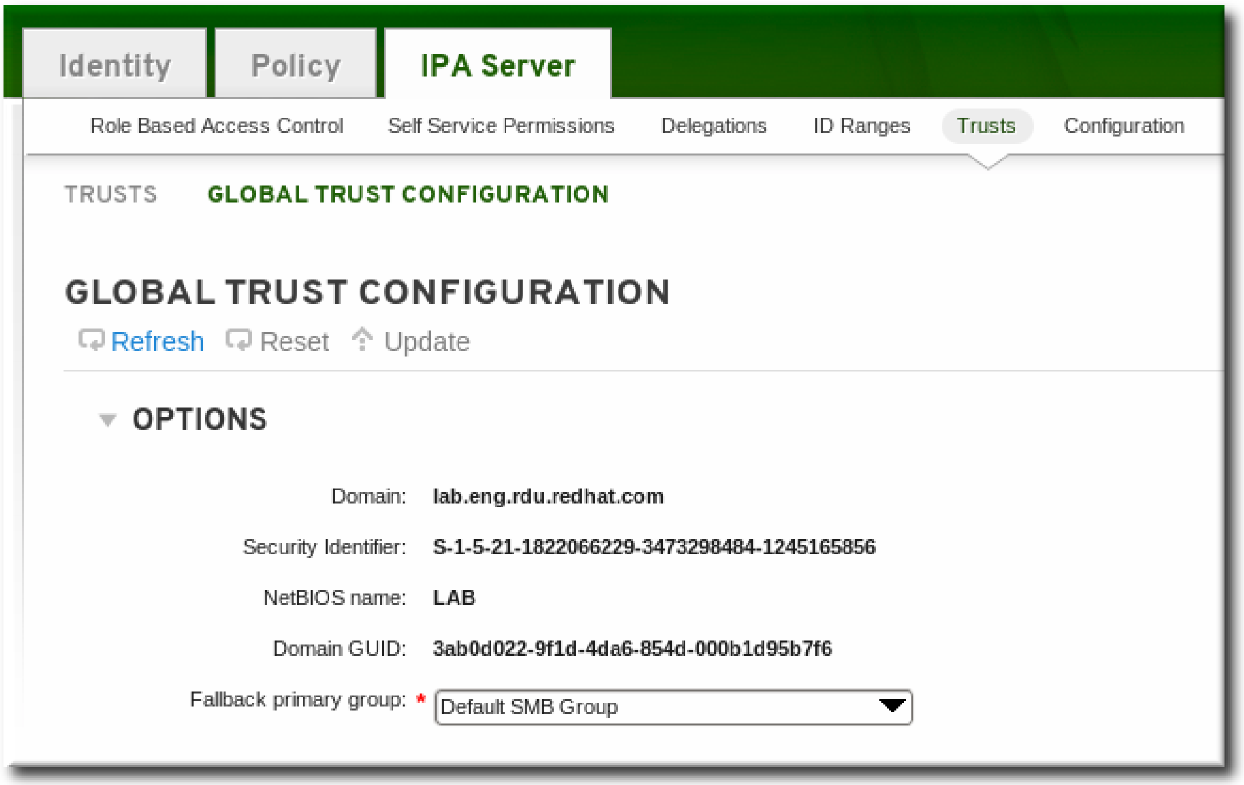
```
[root@server ~]# kinit admin
[root@server ~]# ipa trustconfig-mod --fallback-primary-group="Example
Windows Group"
```

This can also be modified through the IdM web UI.

1. Open the IdM web UI.

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

2. Open the **`IPA Server`** main tab, and then select the **`Trusts`** subtab.
3. In the **`Global Configuration`** subtab, select a new group from all of the IdM groups listed in the **Fallback primary group**.



GLOBAL TRUST CONFIGURATION

Domain:	lab.eng.rdu.redhat.com
Security Identifier:	S-1-5-21-1822066229-3473298484-1245165856
NetBIOS name:	LAB
Domain GUID:	3ab0d022-9f1d-4da6-854d-000b1d95b7f6
Fallback primary group: *	Default SMB Group

4. Click the **Update** link to save the new configuration.

5.4.2. Discovering, Enabling, and Disabling Trust Domains

A transitive trust means that the trust path can follow a chain of domains. If Domain A trusts Domain B, and Domain B trusts Domain C, then Domain A implicitly trusts Domain C. The trust it has with Domain B is transitive — it follows the trust path to Domain C.

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

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

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

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

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

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.adexample.com
------------------------------------------
Disabled trust domain "test.adexample.com"
------------------------------------------
```

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

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

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

5.4.3. Viewing and Managing DNS Realms

When a trust is created, the Active Directory DNS configuration is added to the IdM DNS configuration, with each realm being added as a special *realm domain*. Each domain is stored in the **cn=Realm Domains,cn=ipa,cn=etc,dc=example,dc=com** subtree in the IdM directory.

Since these realm domains are added automatically, the DNS zones do not generally need to be added or modified. The list of configured realm domains can be displayed (instead of listing all DNS zones configured in IdM) using the **realmdomains-show** command.

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

If a single realm domain should be added to the configuration, this can be done with the **`--add-domain`** option.

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

A single domain can be removed using the **`--del-domain`** option.

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

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

5.4.4. Adding Ranges for UID/GID Numbers in a Transitive Trust

Windows systems treat ID numbers differently than Linux systems. When a user is created in Linux, it is assigned a user ID number, and a private group is created for that user. The private group UI number is the same as the user ID number. In Linux, there is no conflict in that. On Windows, however, the security ID is unique for every object in the domain, so there is a conflict if a user and a group have the same ID.

If the POSIX attributes (including ***`uidNumber`*** and ***`gidNumber`***) are pulled from the global catalog in Active Directory, then the ID numbers are unique, because they are unique within the Active Directory environment. This is an *ipa-ad-trust-posix* range type, which is set in the **`trust-add`** command when the trust is created. Essentially, no ID validation or range is required.

However, if instead the SIDs will be generated in IdM, using SID/username mapping, then the ID ranges for both the Windows identities and the IdM users and groups need to have unique, non-overlapping ranges available. This is *ipa-ad-trust* range type.

A unique ID range is created for each Active Directory domain automatically when it is added in a trust to IdM. However, Active Directory and IdM can work in a *transitive* trust. A transitive trust is a daisy change, where if Realm A trusts Realm B, and Realm B trusts Realms C, then Realm A also trusts Realm C. When the trust is configured, a range is only added for the domain specified in the trust agreement. Ranges for the transitively-trusted domains need to be added manually.

To add an ID range, set the base ID for the POSIX range (the starting number), the starting number of the RID (the far-right number in the SID), the size of the range, and the domain SID (since there can be multiple domains configured for trusts).

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

The base ID is the starting POSIX ID number. The RID is a range to add to the base ID to prevent conflicts. If the base ID is 1200000 and the RID is 1000, then the resulting ID number is 1201000.

5.5. Verifying That IdM Machines Have Resolvable Names

As Section 5.1.4, “Different DNS-Trust Environments” explains, regardless of the DNS configuration, all hostnames within both the Identity Management and Active Directory DNS domains must be fully-resolvable for trusted services to function reliably.

After configuring trust, verify that the Identity Management servers are resolvable in both the IdM and Active Directory realms.

First, verify that the IdM-hosted services are resolvable from the IdM domain.

1. Run a DNS query for the Kerberos record over UDP.

```
[root@ipaserver ~]# dig +short -t SRV @10.1.1.1
_kerberos._udp.ipa.example.com.
0 100 88 ipamaster1.ipa.example.com.
```

2. Run a DNS query for the LDAP record over TCP.

```
[root@ipaserver ~]# dig +short -t SRV @10.1.1.1
_ldap._tcp.ipa.example.com.
0 100 389 ipamaster1.ipa.example.com.
```

3. Run a DNS query for the TXT record with the Kerberos realm name. This must match the Kerberos realm for the Identity Management servers.

```
[root@ipaserver ~]# dig +short -t TXT @10.1.1.1
_kerberos.ipa.example.com.
```

On the Active Directory server, verify that all of the IdM-hosted servers and services are resolvable.

Active Directory has a utility called **nslookup.exe** which can query the DNS configuration.

1. Set the **nslookup.exe** utility to look up service records.

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

2. Enter the name of the service and (optionally) the IP address of the IdM name server.

```
> _ldap._tcp.ipa.example.com 10.1.1.1
Server:   [10.1.1.1]
Address:   10.1.1.1

_ldap._tcp.ipa.example.com       SRV service location:
          priority                = 0
          weight                  = 100
          port                    = 389
          svr hostname   = ipaserver.ipa.example.com
ipaserver.ipa.example.com      internet address = 10.1.1.1
```

3. Change the service type to TXT to check the IdM Kerberos realm configuration [5]

```
> set type=TXT
```

4. Query the Kerberos records.

```
> _kerberos.ipa.example.com. 10.1.1.1
```

Active Directory caches the results of DNS lookups. The current cache can be viewed by running **`ipconfig /displaydns`**, and the cache can be deleted by running **`ipconfig /flushdns`**.

5.6. Setting PAC Types for Services

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

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

POSIX systems have a similar data set called a *POSIX authorization data* element. PADs, like PACs, contain group-based authorization data for a user. The access data is returned in response to the initial authentication request, so there is no additional cross-realm communication to retrieve group data.

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

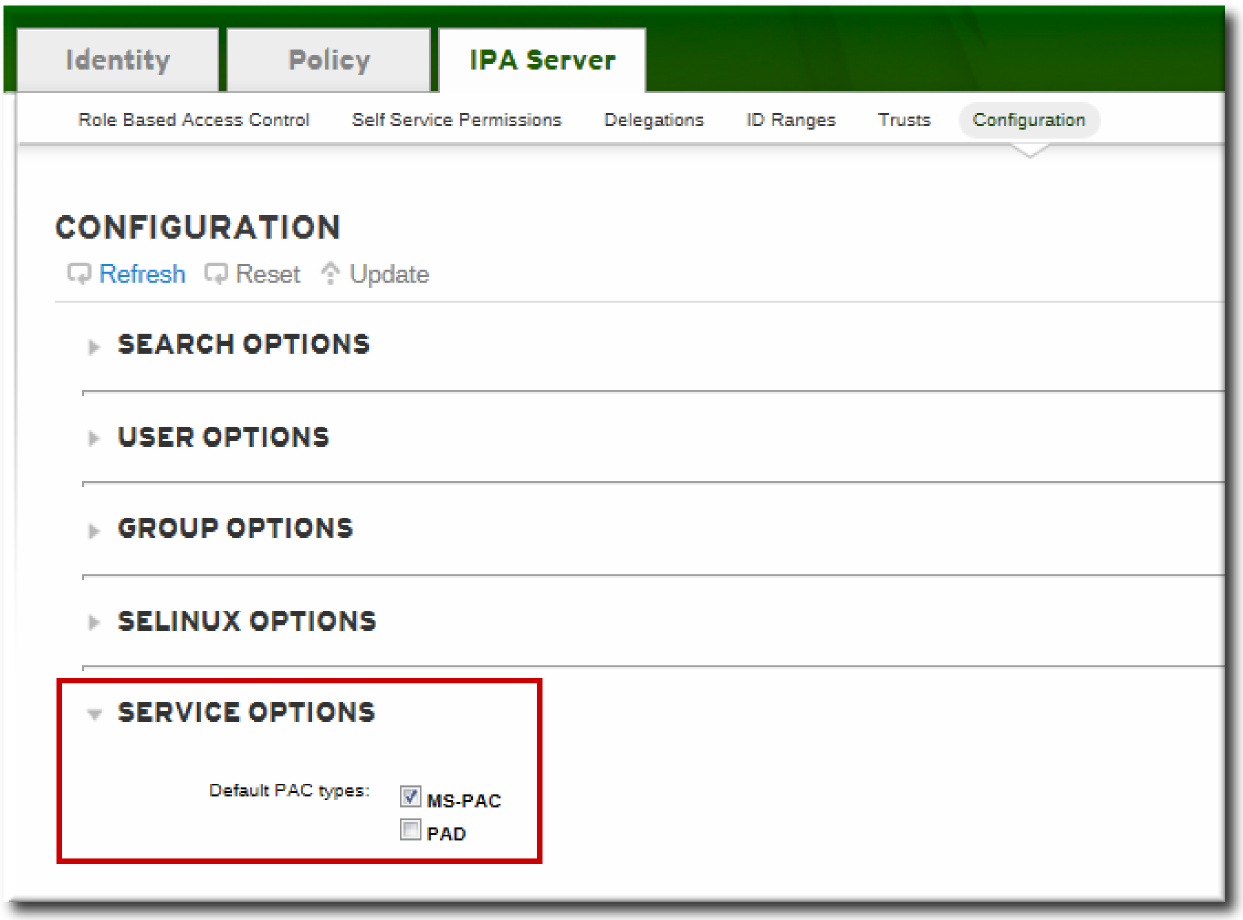
5.6.1. Setting Default PAC Types

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.



Default PAC types: ☑ MS-PAC ☐ PAD

4. Select the check boxes by the PAC types to use. If both PAC types are selected, then both are added to the Kerberos ticket.

   - *MS-PAC* adds a certificate that can be used by Active Directory services.
   - *PAD* adds a certificate that can be used by POSIX (non-Windows) systems.
   - If no checkbox is selected, then no PACs are added to Kerberos tickets.

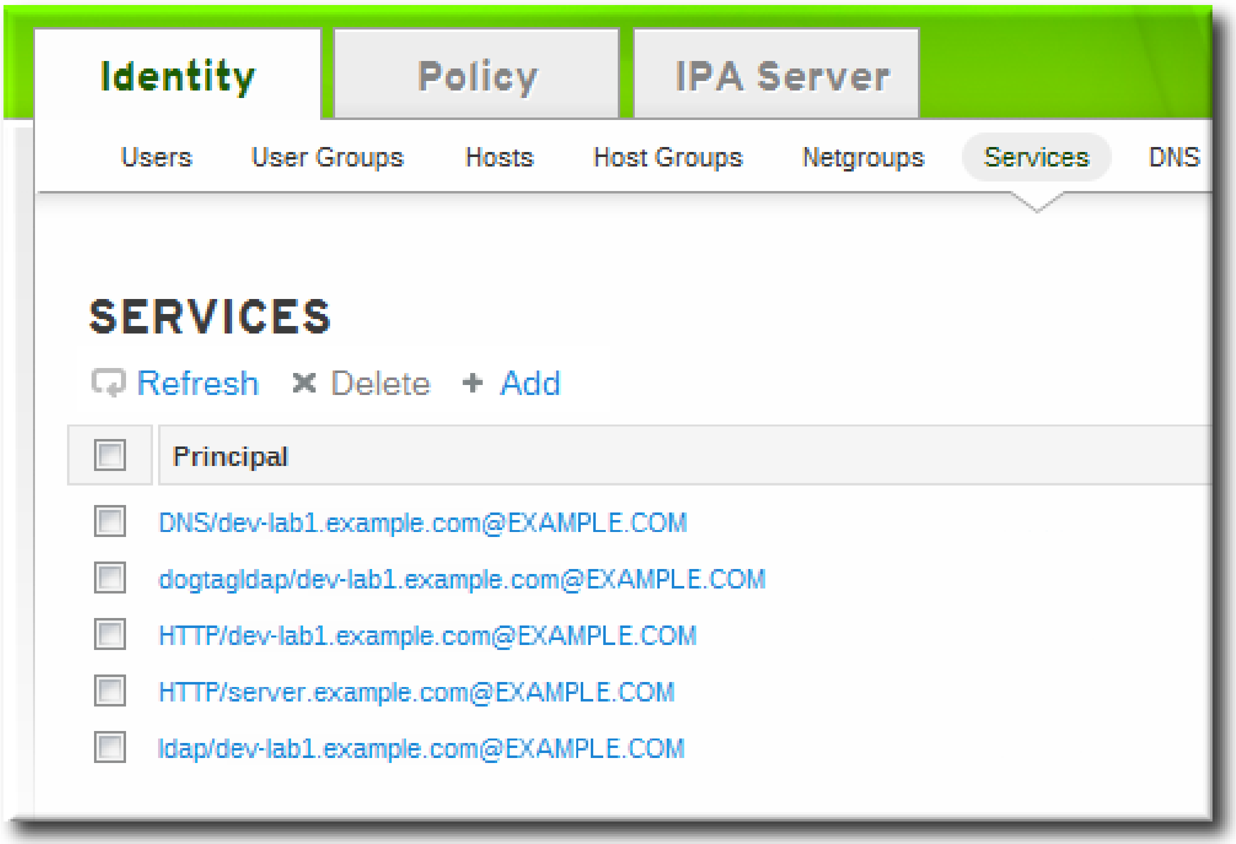
5. Click the **`Update`** link at the top of the page to save the changes.

5.6.2. Setting PAC Types for a Service

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

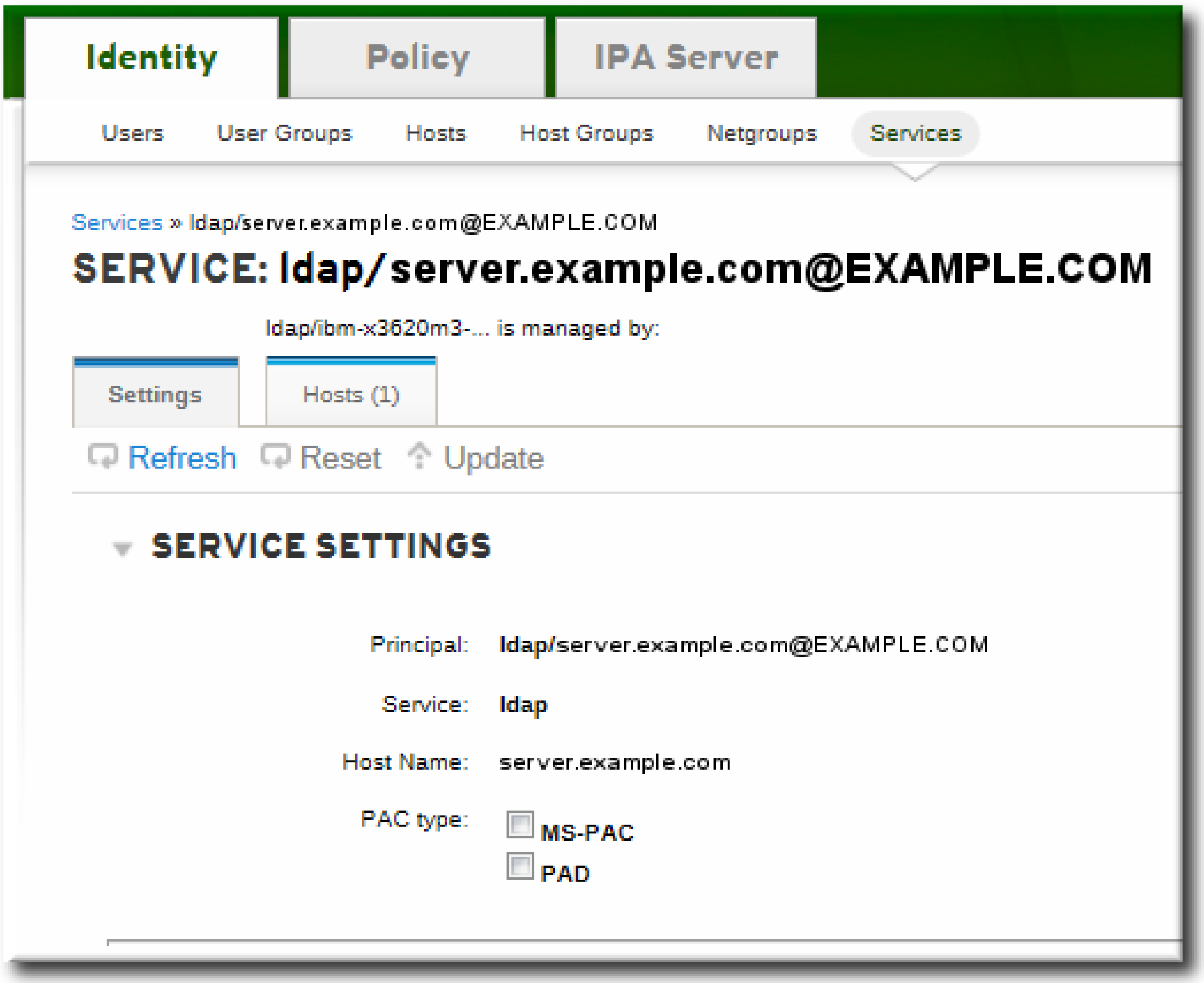
1. Open the **`Identity`** tab, and select the **`Services`** subtab.

2. Click the name of the service to edit.



DNS/dev-lab1.example.com@EXAMPLE.COM
dogtagldap/dev-lab1.example.com@EXAMPLE.COM
HTTP/dev-lab1.example.com@EXAMPLE.COM
HTTP/server.example.com@EXAMPLE.COM
ldap/dev-lab1.example.com@EXAMPLE.COM

3. In the **`Service Settings`** area, select the check boxes by the PAC types to use. If both PAC types are selected, then both are added to the Kerberos ticket.


Identity
Policy
IPA Server
Users
User Groups
Hosts
Host Groups
Netgroups
Services
Services » ldap/server.example.com@EXAMPLE.COM
SERVICE: ldap/server.example.com@EXAMPLE.COM
ldap/ibm-x3620m3-... is managed by:
Settings
Hosts (1)
Refresh
Reset
Update
SERVICE SETTINGS

Principal: ldap/server.example.com@EXAMPLE.COM
Service: ldap
Host Name: server.example.com
PAC type: ☐ MS-PAC ☐ PAD

   - *MS-PAC* adds a certificate that can be used by Active Directory services.
   - *PAD* adds a certificate that can be used by POSIX (non-Windows) systems.
   - If no checkbox is selected, then no PACs are added to Kerberos tickets.

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

5.7. Using SSH from Active Directory Machines for IdM Resources

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

NOTE

When using PuTTY on the Windows machine, make sure that GSS-API credential delegation is enabled.

5.7.1. Username Requirements for SSH

One critical factor when using SSH is the username. The username must meet several criteria:

- The username must have the format *ad_user@ad_domain*.
- The domain name itself must be lower-case. This is required for Kerberos principal mapping.
- The case of the username must match, exactly, the case of the username in Active Directory. *jsmith* and *JSmith* are considered different users because of the different cases.

5.7.2. Using SSH Without Passwords

Even if a proper Kerberos ticket is obtained, using SSH still prompts for a user password for Active Directory domain users. SSH specifies the username with the **`-l`**, but the Kerberos ticket contains the Kerberos principal, not the username. The system requires a way to compare the offered, local username to a principal name to see if the user has a ticket. The **`.k5login`** file offers a simple way to map a local user to a Kerberos principal. The file is in a local user's home directory (and the user is identified by the **`-l`** option with SSH), and it lists Kerberos principals for that user. If the authenticating user matches the principal in an existing Kerberos ticket, then the user is allowed to log in using the ticket for authentication, rather than requiring a password.

To change to Kerberos authentication (meaning, to use passwordless SSH authentication), each Active Directory user should have a **`.k5login`** file in their Linux home directory. The only contents in the file are a list of Kerberos principals used by the user. The principals can be any of the formats in Section 5.2.9, "Supported Username Formats", such as *user@REALM.COM*, *AD.domain\user*, or *AD\user*.

For example, for the user *jsmith* in an Active Directory realm named *ENGINEERING.ADREALM.COM*, the **`.k5login`** is put in the home directory:

```
/home/engineering.adrealm/jsmith/.k5login
```

The contents of the file contain two different principal names:

```
jsmith@ENGINEERING.ADREALM
ENGINEERING.ADREALM.COM\jsmith
```

The **`.k5login`** file is case-sensitive, so listing multiple principals in different formats and different cases can be useful.

Because there are two different principals, either string can be used with **`kinit`** to obtain a ticket.

The **`.k5login`** manpage has more details.

5.8. Using Trust with Kerberized Web Applications

Any existing web application can be configured to use Kerberos authentication, which references the trusted Active Directory and IdM Kerberos realms. The full Kerberos configuration directives are covered in the mod_auth_kerb module man pages.

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

- The ***`KrbAuthRealms`*** directive gives the application location to the name of the IdM domain. This is required.
- The ***`Krb5Keytab`*** gives the location for the IdM server keytab. This is required.
- The ***`KrbServiceName`*** sets the Kerberos service name used for the keytab (HTTP). This is recommended.
- The Kerberos methods directives (***`KrbMethodNegotiate`*** and ***`KrbMethodK5Passwd`***) enables password-based authentication for valid users. This is recommended for ease of use for many users.
- The ***`KrbLocalUserMapping`*** directive enables normal web logins (which are usually the UID or common name of the account) to be mapped to the fully-qualified username (which has a format of *user@REALM.COM*).

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

  Section 5.2.9, “Supported Username Formats” discusses different supported username formats.

Example 5.1. 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>
```

NOTE

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

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

5.9. 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-realm 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 – 5.10
- 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-realm 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

5.9.1. 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`** command 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`** utility. 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.

5.9.2. 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** command 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://vm-093.idm.lab.eng.brq.redhat.com/ipa/config/ca.crt -O
/etc/openldap/cacerts/ipa.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] There are usually no TXT records for Active Directory domains.

Chapter 6. Setting up Kerberos Cross-Realm Authentication

Kerberos v5 creates a realm of clients. That realm can be trust and integrate with an Active Directory domain, as well as other Kerberos domains. Kerberos itself is system-agnostic, so it can work in a number of different environments, systems, and applications.

A lot of Linux environments (and mixed environments) will already have a Kerberos realm deployed for single sign-on, application authentication, and user management. That makes Kerberos a potentially common integration path for mixed Windows-Linux environments, particularly if the Linux environment is not using a more structured domain configuration like Identity Management.

6.1. A Trust Relationship

A *trust* means that the users within one realm are trusted to access the resources in another domain *as if they belonged to that realm*. This is done by creating a shared key for a single principal that is held in common by both domains.

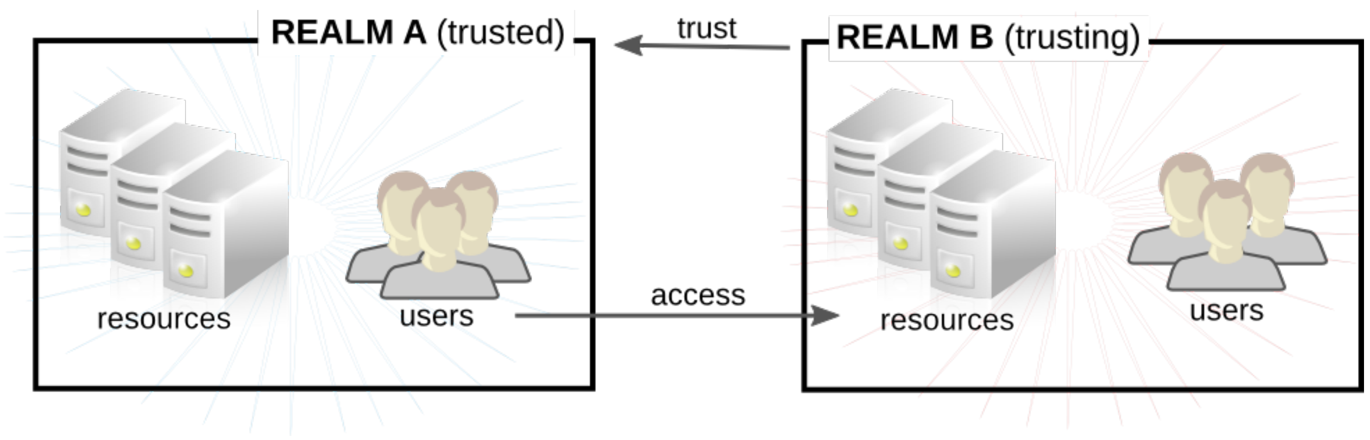


Figure 6.1. Basic Trust

In Figure 6.1, “Basic Trust”, the shared principal would belong to Domain B (**`krbtgt/B.EXAMPLE.COM@A.EXAMPLE.COM`**). When that principal is also added to Domain A, then the clients in Domain A can access the resources in Domain B. The configured principal exists in both realms. That shared principal has three characteristics:

- It exists in both realms.
- When a key is created, the same password is used in both realms.
- The key has the same key version number (kvno).

A cross-realm trust is unidirectional by default. This trust is not automatically reciprocated so that the **`B.EXAMPLE.COM`** realm are trusted to authenticate to services in the **`A.EXAMPLE.COM`** realm. To establish trust in the other direction, both realms would need to share keys for the **`krbtgt/A.EXAMPLE.COM@B.EXAMPLE.COM`** service — an entry with a reverse mapping from the previous example.

A realm can have multiple trusts, both realms that it trusts and realms it is trusted by. With Kerberos trusts, the trust can flow in a chain. If Realm A trusts Realm B and Realm B trusts Realm C, Realm A implicity trusts Realm C, as well. The trust flows along realms; this is a *transitive* trust.

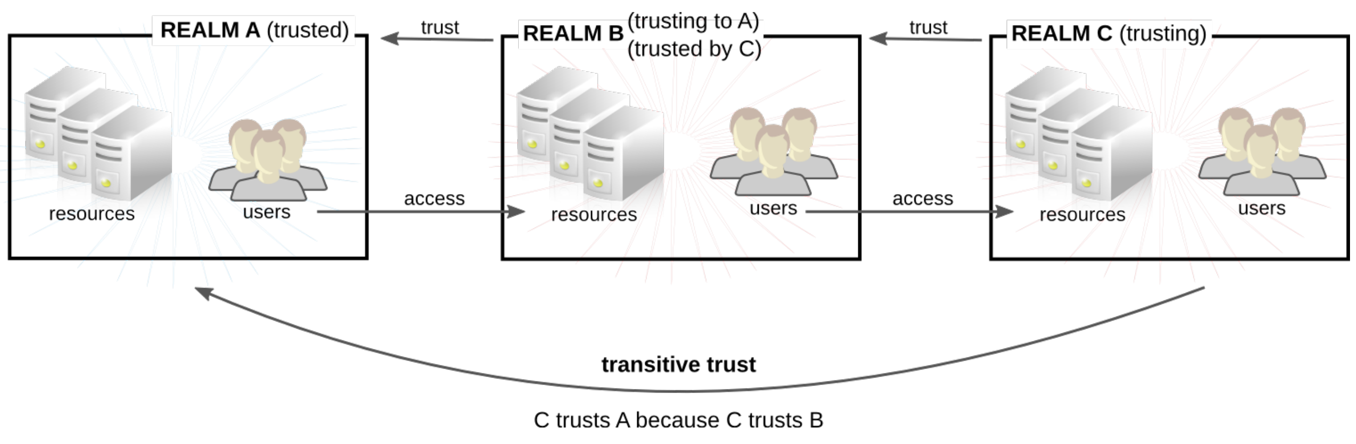


Figure 6.2. Transitive Trust

NOTE

While a Kerberos trust is transitive by default, that is not necessarily true for a Kerberos trust with a Windows domain. In Windows, trusts with other Windows domains is transitive, but trusts with external (meaning, non-Windows) realms are non-transitive by default. They can be configured to be transitive.

The direction of a transitive trust is the *trust flow*. The trust flow has to be defined, first by recognizing to what realm a service belongs and then by identifying what realms a client must contact to access that service.

A Kerberos principal name is structured in the format *service/hostname@REALM*. The *service* is generally a protocol, such as LDAP, IMAP, HTTP, or host. The *hostname* is the fully-qualified domain name of the host system, and the *REALM* is the Kerberos realm to which it belongs. Clients usually map the hostname or DNS domain name to the realm. The realm, then, somewhat related to the DNS domain name (uncless a realm is explicitly defined in the **`domain_realm`** section of **`/etc/krb5.conf`**).

When traversing a trust, Kerberos assumes that each realm is structured like a hierarchical DNS domain, with a root domain and subdomains. This means that the trust flows up to a shared root. Each step, or *hop*, has a shared key. In Figure 6.3, “Trusts in the Same Domain”, A shares a key with EXAMPLE.COM, and EXAMPLE.COM shares a key with B.

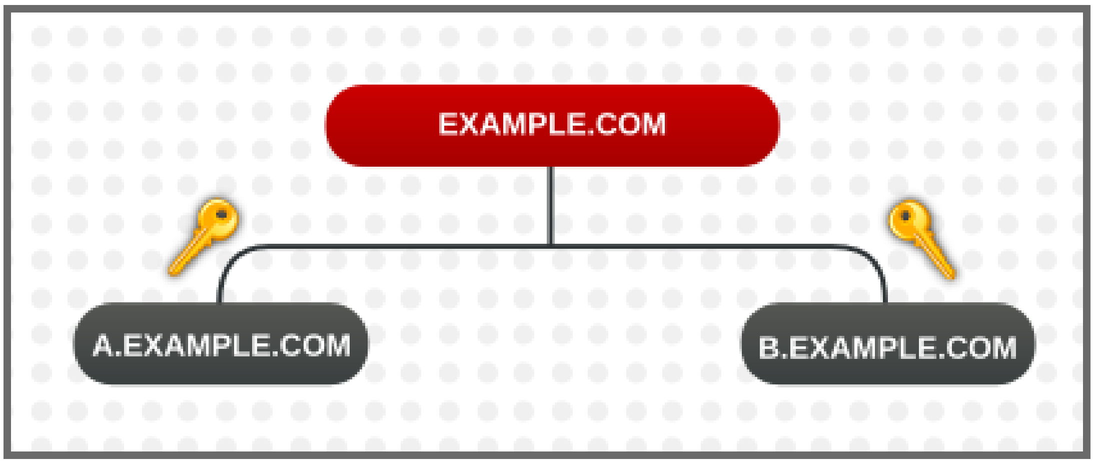


Figure 6.3. Trusts in the Same Domain

The client treats the relam name as a DNS name, and it determines its trust path by stripping off elements of its own realm name until it reaches the root name. It then begins prepending names until it reaches the service's realm.

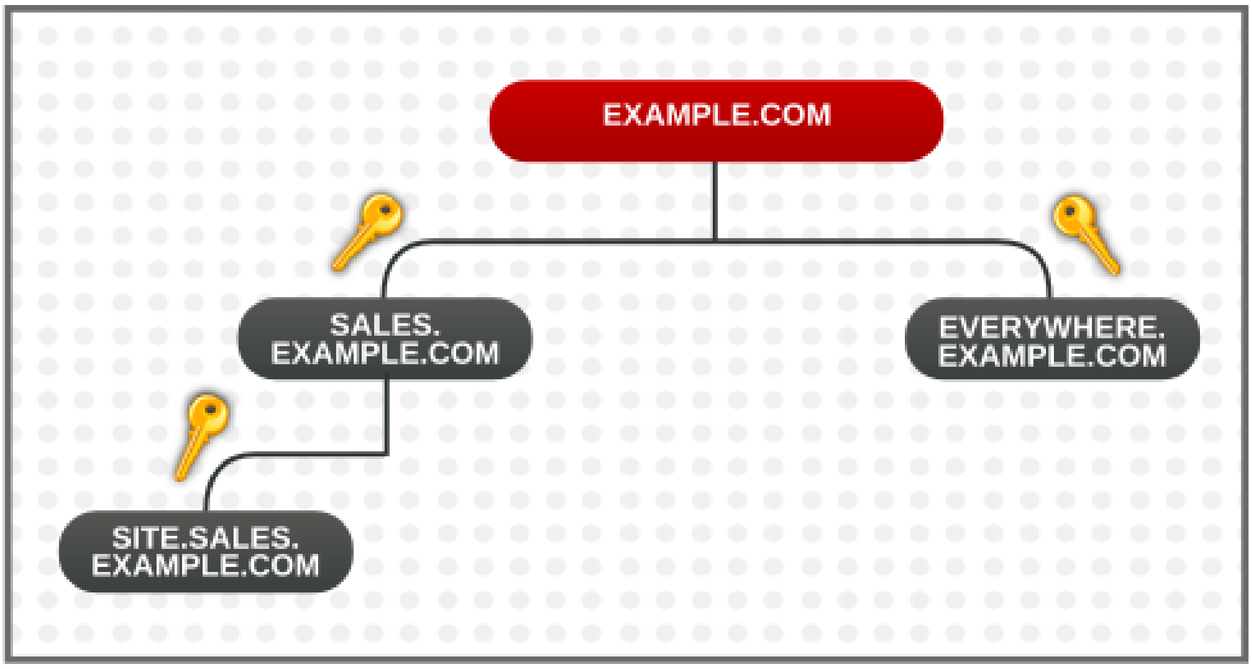


Figure 6.4. Child/Parent Trusts in the Same Domain

This is a nature of trusts being transitive. SITE.SALES.EXAMPLE.COM only has a single shared key, with SALES.EXAMPLE.COM. But because of a series of small trusts, there is a large trust flow that allows trust to go from SITE.SALES.EXAMPLE.COM to EVERYWHERE.EXAMPLE.COM.

That trust flow can even go between completely different domains by creating a shared key at the domain level, where the sites share no common suffix.

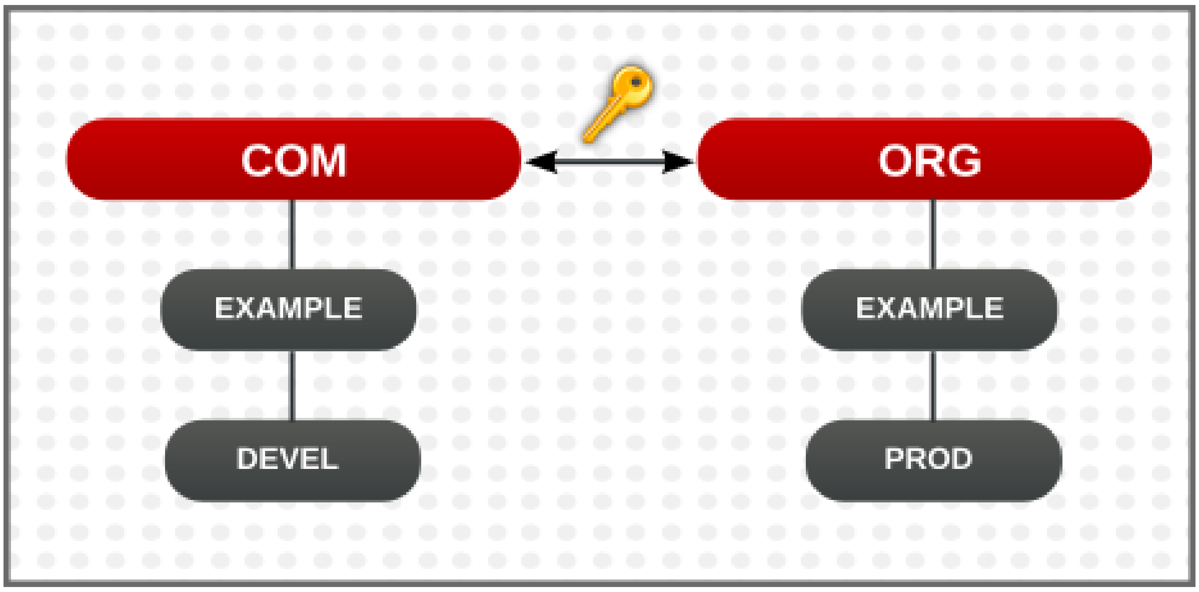


Figure 6.5. Trusts in Different Domains

It is also possible to reduce the number of hops and represent very complex trust flows by explicitly defining the flow. The **capaths** section of the **/etc/krb5.conf** file defines the trust flow between different realms.

The format of the **capaths** section is relatively straightforward: there is a main entry for each realm where a client has a principal, and then inside each realm section is a list of intermediate realms from which the client must obtain credentials.

For example, this has the realm of **A.EXAMPLE.COM**, and a set of hops from A to D. A client in Realm A must obtain credentials first from Realm B (the **.** means that it can obtain credentials directly, without any intermmediate hops; otherwise, it would attempt to gain credentials by going through a hierarchy). It must then use the B credentials to obtain credentials from C, and then use the C credentials to obtain credentials for D.

```
[capaths]
A.EXAMPLE.COM = {
B.EXAMPLE.COM = .
C.EXAMPLE.COM = B.EXAMPLE.COM
D.EXAMPLE.COM = C.EXAMPLE.COM
}
```

6.2. Setting up a Realm Trust

In this example, the Kerberos realm is **KRB.EXAMPLE.COM**, and the Active Directory realm is **AD.EXAMPLE.COM**.

1. Create the entry for the shared principal in Kerberos, using **kadmin**.

   ```
   [root@server ~]# kadmin -r KRB.EXAMPLE.COM
   kadmin: add_principal krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM
   Enter password for principal
   "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM":
   Re-enter password for principal
   "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM":
   Principal "krbtgt/AD.EXAMPLE.COM@KRB.EXAMPLE.COM" created.
   quit
   ```

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

   - The **Trust Type** is **Realm**.
   - The **Transitivity of Trust** can be either transitive or non-transitive.
   - The **Direction of Trust** is **One-way: incoming**. This trusts Active Directory users in the Kerberos realm.

     This creates a unidirectional trust, where Active Directory users are trusted in the Kerberos realm. To create a two-way trust, set the direction of trust to two-way. This is described in the Microsoft TechNet documentation.
   - The **Sides of Trust** is **This domain only**.
   - The **Trust Password** can be anything. This must be used when configuring the trust in Kerberos.

Chapter 7. Synchronizing Active Directory and Identity Management Users

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

Entry synchronization is performed through a process similar to replication, which uses hooks to connect to and retrieve directory data from the Windows server. This is functionality is available immediately in Identity Management, with no additional configuration in the Active Directory domain.

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

7.1. Supported Windows Platforms

Synchronization are supported with these Windows servers:

- Windows Server 2008 R2
- Windows Server 2012 R2

The version of the password sync service which works with Windows is 1.1.5. This is available in the Red Hat Directory Server downloads part of Red Hat Network.

7.2. About Active Directory and Identity Management

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

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

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

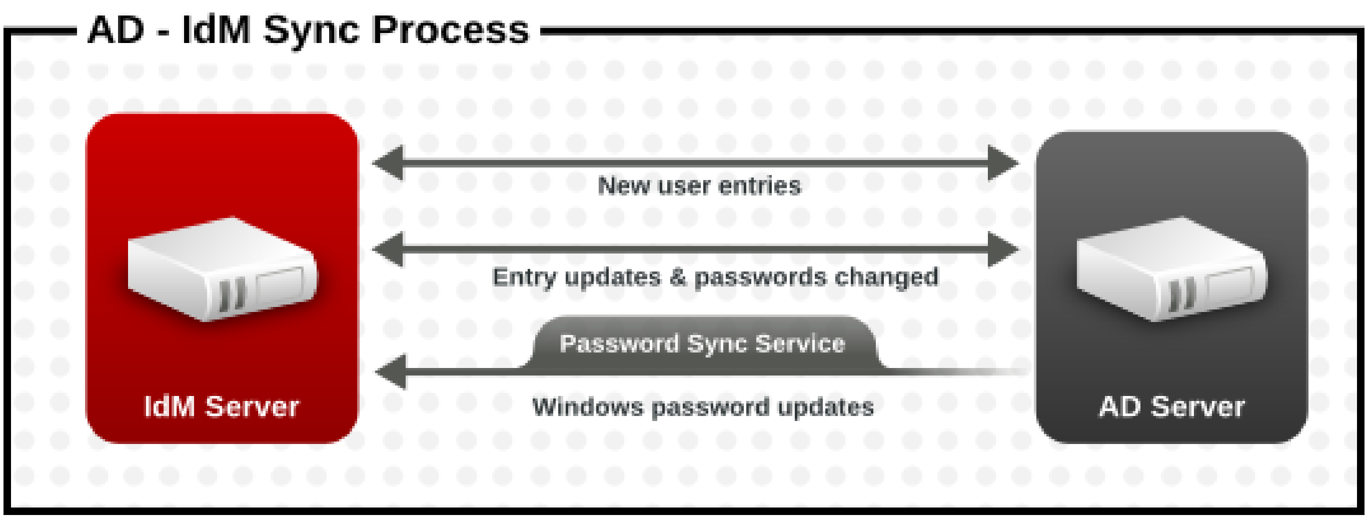


Figure 7.1. Active Directory and IdM Synchronization

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

Table 7.1. Information in a Sync Agreement

Windows Information	IdM Information
• User subtree (**`cn=Users,$SUFFIX`**) • Connection information ◦ Active Directory administrator username and password ◦ Password Sync Service password ◦ CA certificate	• User subtree (**`ou=People,$SUFFIX`**)

Synchronization is most commonly *bi-directional*. Information is sent back and forth between the IdM domain and the Windows domain in a process that is very similar to how IdM servers and replicas share information among themselves. It is possible to configure synchronization to only sync one way. That is *uni-directional* synchronization.

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

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

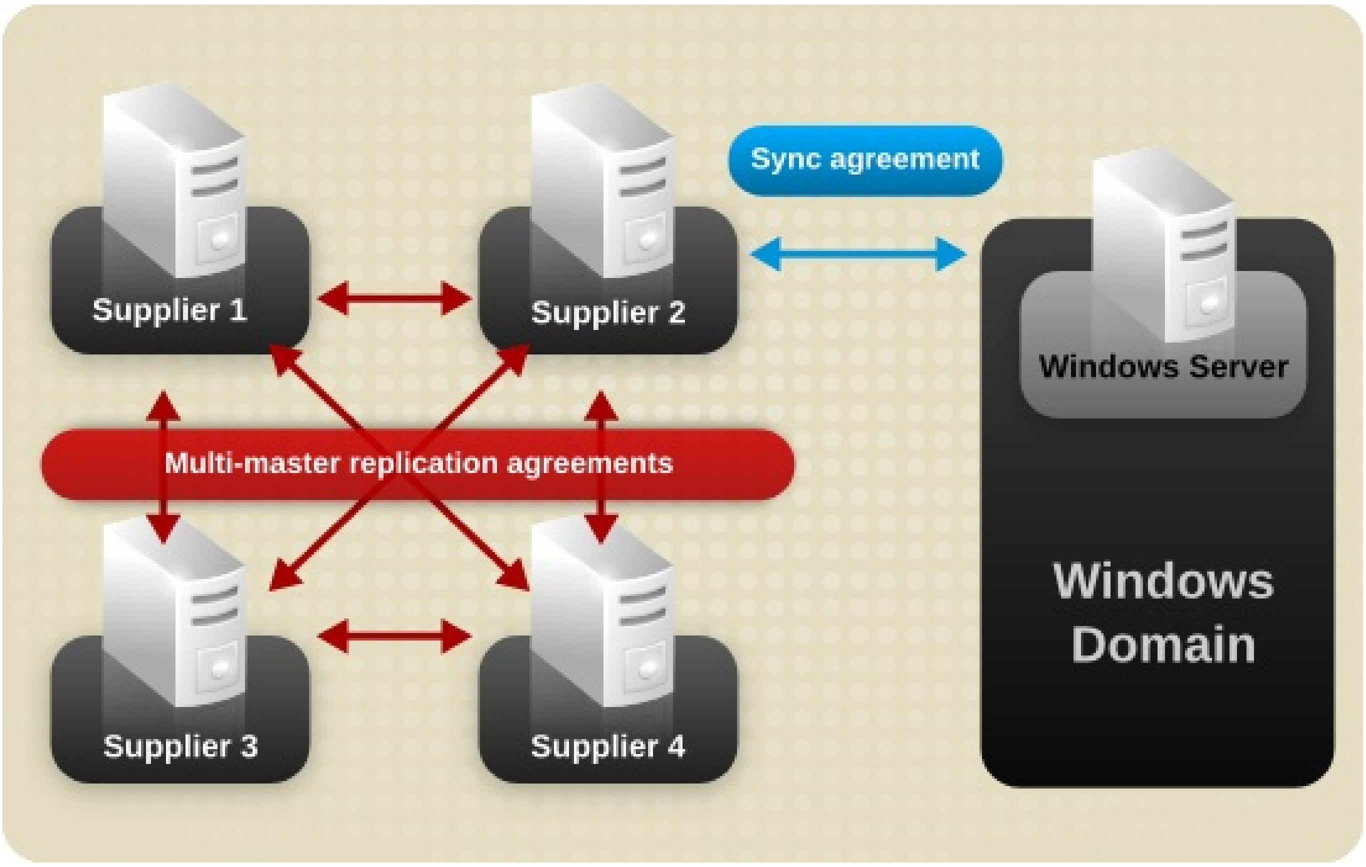


Figure 7.2. Synchronization Topology

There are some key features to IdM synchronization:

- A synchronization operation runs every five minutes.
- Synchronization can only be configured with one Active Directory domain.
- Synchronization can only be configured with *one* Active Directory domain controller.
- Only user information is synchronized.
- Both user attributes and passwords can be synchronized.
- While modifications are bi-directional (going both from Active Directory to IdM and from IdM to Active Directory), creating accounts is only uni-directional, from Active Directory to Identity Management. New accounts created in Active Directory are synchronized over to IdM automatically. However, user accounts created in IdM must also be created in Active Directory before they will be synchronized.
- Account lock information is synchronized by default, so a user account which is disabled in one domain is disabled in the other.
- Password synchronization changes take effect immediately. If a user password is added or changed on one peer, that change is immediately propagated to the other peer server.

  The Password Sync client synchronizes new passwords or password updates.

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

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

Some of the data in synchronization can be modified as part of the synchronization process. For examples, certain attributes can be automatically added to Active Directory user accounts when they are synced over to the IdM domain. These attribute changes are defined as part of the synchronization agreement and are described in Section 7.5.3, "Changing the Behavior for Syncing User Account Attributes".

7.3. About Synchronized Attributes

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

NOTE

Most POSIX attributes are not synchronized.

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

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

- cn [6]
- physicalDeliveryOfficeName
- description
- postOfficeBox
- destinationIndicator
- postalAddress
- facsimileTelephoneNumber
- postalCode
- givenname
- registeredAddress
- homePhone
- sn
- homePostalAddress

- st
- initials
- street
- l
- telephoneNumber
- mail
- teletexTerminalIdentifier
- mobile
- telexNumber
- o
- title
- ou
- userCertificate
- pager
- x121Address

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

Table 7.2. User Schema Mapped between Identity Management and Active Directory

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

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

7.3.1. User Schema Differences between Identity Management and Active Directory

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

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

7.3.1.1. Values for cn Attributes

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

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

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

7.3.1.2. Values for street and streetAddress

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

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

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

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

7.3.1.3. Constraints on the initials Attribute

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

7.3.1.4. Requiring the surname (sn) Attribute

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

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

7.3.2. Active Directory Entries and POSIX Attributes

Windows uses unique, random *security IDs (SIDs)* to identify users. These SIDs are assigned in blocks or ranges, identifying different system user types within the Windows domain. When users are synchronized between Identity Management and Active Directory, Windows SIDs for users are mapped to the Unix UIDs used by the Identity Management entry. Another way of saying this is that the Windows SID is the only ID within the Windows entry which is used as an identifier in the corresponding Unix entry, and then it is used in a mapping.

When Active Directory domains interact with Unix-style applications or domains, then the Active Directory domain may use Services for Unix or IdM for Unix to enable Unix-style ***uidNumber*** and ***gidNumber*** attributes. This allows Windows user entries to follow the specifications for those attributes in RFC 2307.

However, the ***uidNumber*** and ***gidNumber*** attributes are not actually used as the ***uidNumber*** and ***gidNumber*** attributes for the Identity Management entry. The Identity Management ***uidNumber*** and ***gidNumber*** attributes are generated when the Windows user is synced over.

NOTE

The ***uidNumber*** and ***gidNumber*** attributes defined and used in Identity Management are not the same ***uidNumber*** and ***gidNumber*** attributes defined and used in the Active Directory entry, and the numbers are not related.

7.4. Setting up Active Directory for Synchronization

Synchronizing user accounts alone is enabled within IdM, so all that is necessary is to set up a sync agreement (Section 7.5.2, “Creating Synchronization Agreements”). However, the Active Directory does need to be configured in a way that allows the Identity Management server to connect to it.

7.4.1. Creating an Active Directory User for Sync

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

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

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

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

- Add the sync user as a member of the **Account Operator** and **Enterprise Read-Only Domain controller** groups. It is not necessary for the user to belong to the full **Domain Admin** group.

7.4.2. Setting up an Active Directory Certificate Authority

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

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

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

7.5. Managing Synchronization Agreements

7.5.1. Trusting the Active Directory and IdM CA Certificates

Both Active Directory and Identity Management use certificates for server authentication. For the Active Directory and IdM SSL server certificates to be trusted by each other, both servers need to trust the CA certificate for the CA which issued those certificates. This means that the Active Directory CA certificate needs to be imported into the IdM database, and the IdM CA certificate needs to be imported into the Active Directory database.

1. On the Active Directory server, download the IdM server's CA certificate from **`http://ipa.example.com/ipa/config/ca.crt`**.

2. Install the IdM CA certificate in the Active Directory certificate database. This can be done using the Microsoft Management Console or the certutil utility.

   Right-click on the executable, and select **`Run as administrator`**, then run **`certutil`** with the **`-installcert`** option. For example:

   ```
   C:\Windows\system32\certutil -installcert -v -config
   "ipaserver.example.com\Example Domain CA" c:\path\to\ca.crt
   ```

   This command must be run as an administrative account, or it will fail because it cannot access the certificate database.

   For more details on installing certificates, see the Active Directory documentation.

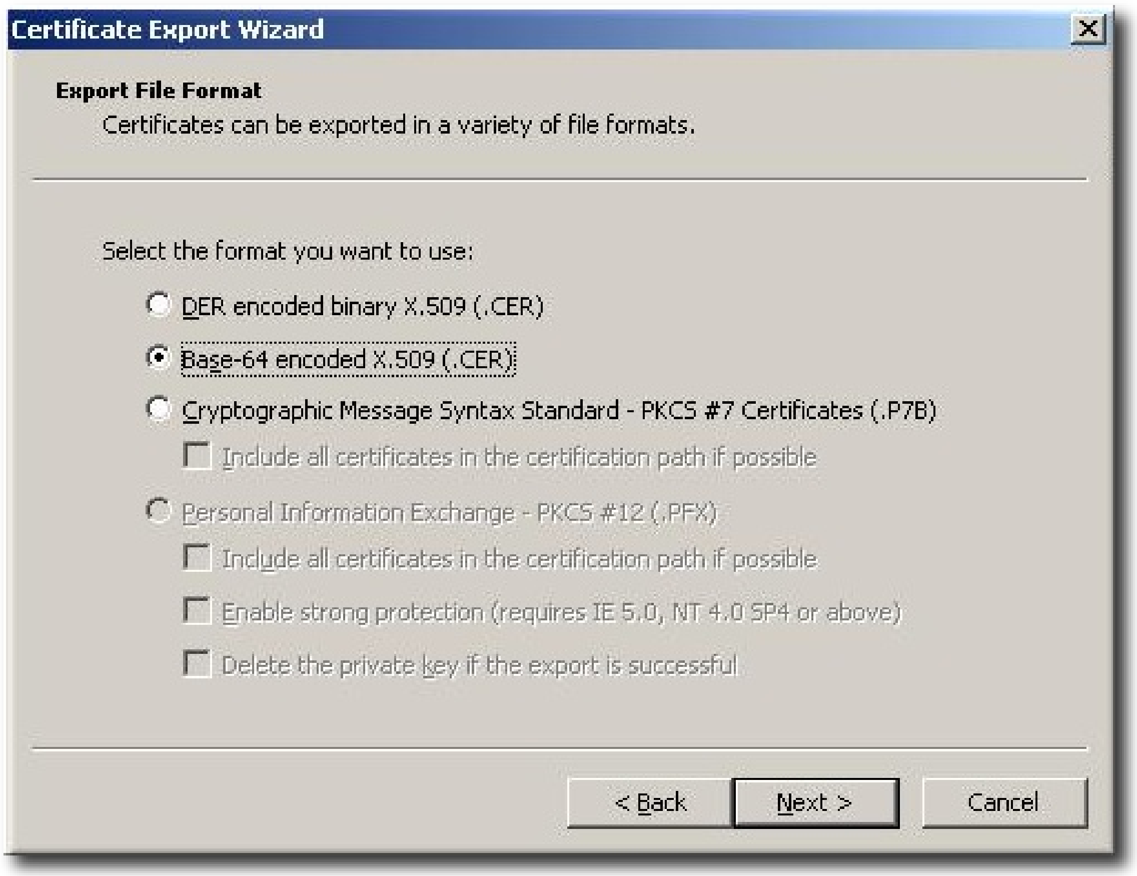
3. Export the Active Directory CA certificate.

   a. In **`My Network Places`**, open the CA distribution point.

   b. Double-click the security certificate file (**`.crt`** file) to display the **`Certificate`** dialog box.

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

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


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
  - [ ] Enable strong protection (requires IE 5.0, NT 4.0 SP4 or above)
  - [ ] Delete the private key if the export is successful

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

4. Copy the Active Directory certificate over to the IdM server machine.

5. Download the IdM server's CA certificate from **http://ipa.example.com/ipa/config/ca.crt**.

6. Copy both the Active Directory CA certificate and the IdM CA certificate into the **/etc/openldap/cacerts/** directory.

7. Update the hash symlinks for the certificates.

```
cacertdir_rehash /etc/openldap/cacerts/
```

8. Edit the **/etc/openldap/ldap.conf** file, and add the information to point to and use the certificates in the **/etc/openldap/cacerts/** directory.

```
TLS_CACERTDIR /etc/openldap/cacerts/
TLS_REQCERT allow
```

7.5.2. Creating Synchronization Agreements

Synchronization agreements are created on the IdM server using the **ipa-replica-manage connect** command because it creates a *connection* to the Active Directory domain. The options to create the synchronization agreement are listed in Table 7.3, "Synchronization Agreement Options".

1. Make sure that the Active Directory and IdM servers trust each other's CA certificates, as in Section 7.5.1, “Trusting the Active Directory and IdM CA Certificates”.

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

   ```
   $ kdestroy
   ```

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

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

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

4. When prompted, enter the Directory Manager password.

5. *Optional.* Configure Password Synchronization, as in Section 7.6.2, “Setting up Password Synchronization”. Without the Password Sync client, user attributes are synchronized between the peer servers, but passwords are not.

NOTE

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

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

Table 7.3. Synchronization Agreement Options

Option	Description
--winsync	Identifies this as a synchronization agreement.
--binddn	Gives the full user DN of the synchronization identity. This is the user DN that the IdM LDAP server uses to bind to Active Directory. This user must exist in the Active Directory domain and must have replicator, read, search, and write permissions on the Active Directory subtree.
--bindpw	Gives the password for the sync user.

Option	Description
--passsync	Gives the password for the Windows user account which is involved in synchronization.
--cacert	Gives the full path and file name of the Active Directory CA certificate. This certificate is exported in Section 7.5.1, “Trusting the Active Directory and IdM CA Certificates”.
--win-subtree	Gives the DN of the Windows subtree containing the users to synchronize. The default value is **`cn=Users,$SUFFIX`**.
AD_server_name	Gives the hostname of the Active Directory domain controller.

7.5.3. Changing the Behavior for Syncing User Account Attributes

When the sync agreement is created, it has certain default behaviors defined for how the synchronization process handles the user account attributes during synchronization. The types of behaviors are things like how to handle lockout attributes or how to handle different DN formats. This behavior can be changed by editing the synchronization agreement. The list of attribute-related parameters are in Table 7.4, “Synced Attribute Settings”.

The sync agreement exists as a special plug-in entry in the LDAP server and each attribute behavior is set through an LDAP attribute. To change the sync behavior, use the **`ldapmodify`** command to modify the LDAP server entry directly.

For example, account lockout attributes are synchronized between IdM and Active Directory by default, but this can be disabled by editing the ***`ipaWinSyncAcctDisable`*** attribute. (Changing this means that if an account is disabled in Active Directory, it is still active in IdM and vice versa.)

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password

dn: cn=ipa-winsync,cn=plugins,cn=config
changetype: modify
replace: ipaWinSyncAcctDisable
ipaWinSyncAcctDisable: none

modifying entry "cn=ipa-winsync,cn=plugins,cn=config"
```

Table 7.4. Synced Attribute Settings

Parameter	Description	Possible Values
General User Account Parameters		
ipaWinSyncNewEntryFilter	Sets the search filter to use to find the entry which contains the list of object classes to add to new user entries.	The default is **`(cn=ipaConfig)`**.
ipaWinSyncNewUserOCAttr	Sets the attribute in the configuration entry which actually contains the list of object classes to add to new user entries.	The default is **`ipauserobjectclasses`**.

Parameter	Description	Possible Values
ipaWinSyncHomeDirAttr	Identifies which attribute in the entry contains the default location of the POSIX home directory.	The default is **`ipaHomesRootDir`**.
ipaWinSyncUserAttr	Sets an additional attribute with a specific value to add to Active Directory users when they are synced over from the Active Directory domain. If the attribute is multi-valued, then it can be set multiple times, and the sync process adds all of the values to the entry. **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 synced over.	ipaWinSyncUserAttr: *attributeName attributeValue*
ipaWinSyncForceSync	Sets whether to check existing IdM users which match an existing Active Directory user should be automatically edited so they can be synchronized. If an IdM user account has a ***`uid`*** parameter which is identical to the ***`sAMAccountName`*** in an existing Active Directory user, then that account is *not* synced by default. This attribute tells the sync service to add the ***`ntUser`*** and ***`ntUserDomainId`*** to the IdM user entries automatically, which allows them to be synchronized.	true | false
User Account Lock Parameters		

Parameter	Description	Possible Values
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 synced over to Active Directory and overrides the local Active Directory value. By default, account lockout attributes are synced from both domains.	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.	The default is **(&(cn=inactivated)(objectclass=groupOfNames))**.
ipaWinSyncActivatedFilter	Sets the search filter to use to find the DN of the group used to hold active users. This does not need to be changed in most deployments.	The default is **(&(cn=activated)(objectclass=groupOfNames))**.
Group Parameters		
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.	The default is **ipaDefaultPrimaryGroup**.
ipaWinSyncDefaultGroupFilter	Sets the search filter to map the group name to the POSIX ***gidNumber***.	The default is **(&(gidNumber=*)(objectclass=posixGroup)(cn=***groupAttr_value***))**.
Realm Parameters		
ipaWinSyncRealmAttr	Sets the attribute which contains the realm name in the realm entry.	The default is ***cn***.
ipaWinSyncRealmFilter	Sets the search filter to use to find the entry which contains the IdM realm name.	The default is **(objectclass=krbRealmContainer)**.

7.5.4. Changing the Synchronized Windows Subtree

Creating a synchronization agreement automatically sets the two subtrees to use as the synchronized user database. In IdM, the default is **cn=users,cn=accounts,$SUFFIX**, and for Active Directory, the default is **CN=Users,$SUFFIX**.

The value for the Active Directory subtree can be set to a non-default value when the sync agreement is created by using the **--win-subtree** option. After the agreement is created, the Active Directory subtree can be changed by using the **ldapmodify** command to edit the ***nsds7WindowsReplicaSubtree*** value in the sync agreement entry.

1. Get the name of the sync agreement, using **ldapsearch**. This search returns only the values for the ***dn*** and ***nsds7WindowsReplicaSubtree*** attributes instead of the entire entry.

```
[jsmith@ipaserver ~]$ ldapsearch -xLLL -D "cn=directory manager" -w
password -p 389 -h ipaserver.example.com -b cn=config
objectclass=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 sync 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 sync operation is currently running, then it takes effect as soon as the current operation completes.

7.5.5. Configuring Uni-Directional Sync

By default, all modifications and deletions are bi-directional. A change in Active Directory is synced over to Identity Management, and a change to an entry in Identity Management is synced over to Active Directory. This is essentially an equitable, multi-master relationship, where both Active Directory and Identity Management are equal peers in synchronization and are both data masters.

However, there can be some data structure or IT designs where only one domain should be a data master and the other domain should accept updates. This changes the sync relationship from a multi-master relationship (where the peer servers are equal) to a master-consumer relationship.

This is done by setting the ***oneWaySync*** parameter on the sync agreement. The possible values are ***fromWindows*** (for Active Directory to Identity Management sync) and ***toWindows*** (for Identity Management to Active Directory sync).

For example, to sync changes from Active Directory to Identity Management:

```
[jsmith@ipaserver ~]$ ldapmodify -x -D "cn=directory manager" -w password
-p 389 -h ipaserver.example.com
```

```
dn:
cn=windows.example.com,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
tree,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```

IMPORTANT

Enabling uni-directional sync does *not* automatically prevent changes on the un-synchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync is configured to go from Active Directory to Identity Management, so Active Directory is (in essence) the data master. If an entry is modified or even deleted on the Identity Management, then the Identity Management information is different then the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

7.5.6. Deleting Synchronization Agreements

Synchronization can be stopped by deleting the sync agreement which *disconnects* the IdM and Active Directory servers. In the inverse of creating a sync agreement, deleting a sync agreement uses the **`ipa-replica-manage disconnect`** command and then the hostname of the Active Directory server.

1. Delete the sync agreement.

   ```
   # ipa-replica-manage disconnect adserver.example.com
   ```

2. Remove the Active Directory CA certificate from the IdM server database:

   ```
   # certutil -D -d /etc/dirsrv/slapd-EXAMPLE.COM/ -n "Imported CA"
   ```

7.5.7. Winsync Agreement Failures

Creating the sync agreement fails because it cannot connect to the Active Directory server.

One of the most common sync agreement failures is that the IdM server cannot connect to the Active Directory server:

```
"Update failed! Status: [81  - LDAP error: Can't contact LDAP server]
```

This can occur if the wrong Active Directory CA certificate was specified when the agreement was created. This creates duplicate certificates in the IdM LDAP database (in the **`/etc/dirsrv/slapd-DOMAIN/`** directory) with the name *Imported CA*. This can be checked using **`certutil`**:

```
$ certutil -L -d /etc/dirsrv/slapd-DOMAIN/

Certificate Nickname                                         Trust
Attributes
```

```
SSL,S/MIME,JAR/XPI

CA certificate                                              CTu,u,Cu
Imported CA                                                 CT,,C
Server-Cert                                                 u,u,u
Imported CA                                                 CT,,C
```

To resolve this issue, clear the certificate database:

```
# certutil -d /etc/dirsrv/slapd-DOMAIN-NAME -D -n "Imported CA"
```

This deletes the CA certificate from the LDAP database.

There are errors saying passwords are not being synced because it says the entry exists

For some entries in the user database, there may be an informational error message that the password is not being reset because the entry already exists:

```
"Windows PassSync entry exists, not resetting password"
```

This is not an error. This message occurs when an exempt user, the Password Sync user, is not being changed. The Password Sync user is the operational user which is used by the service to change the passwords in IdM.

7.6. Managing Password Synchronization

Synchronizing user entries is configured with the sync agreement. However, passwords in both Active Directory and Identity Management are not part of the normal user synchronization process. A separate client must be installed on the Active Directory servers to capture passwords as user accounts are created or passwords are changed, and then to forward that password information with the sync updates.

NOTE

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

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

7.6.1. Setting up the Windows Server for Password Synchronization

Synchronizing passwords requires two things:

- Active Directory must be running in SSL.
- The Password Sync Service must be installed on *each* Active Directory domain controller.

The Password Sync Service records password changes and synchronizes them, over a secure connection, to the IdM entry.

TIP

Install the Microsoft Certificate System in Enterprise Root Mode. Active Directory will then automatically enroll to retrieve its SSL server certificate.

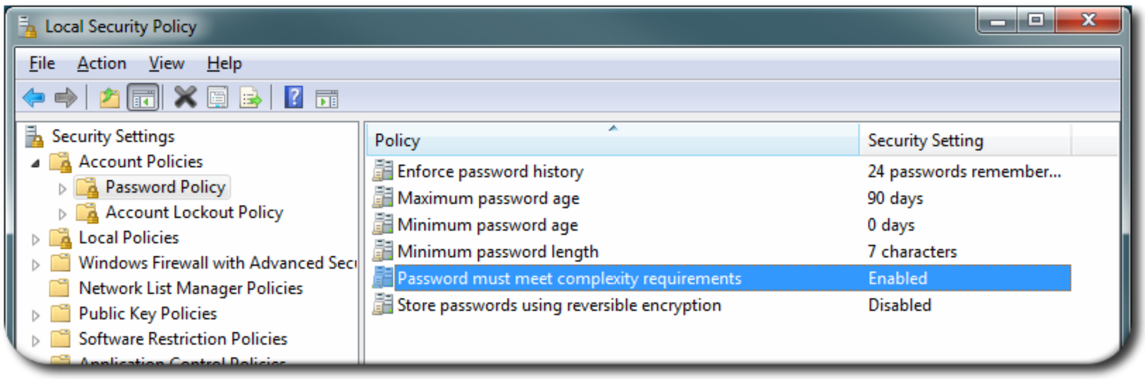
1. Make sure that the Active Directory password complexity policies are enabled so that the Password Sync service will run.

   a. Run **secpol.msc** from the command line.

   b. Select **Security Settings**.

   c. Open **Account Policies**, and then open **Password Policy**.

   d. Enable the **Password must meet complexity requirements** option and save.



Policy	Security Setting
Enforce password history	24 passwords remember...
Maximum password age	90 days
Minimum password age	0 days
Minimum password length	7 characters
Password must meet complexity requirements	Enabled
Store passwords using reversible encryption	Disabled

2. If SSL is not already enabled, set up SSL on the Active Directory server. Setting up LDAPS is explained in more detail in the Microsoft knowledgebase at http://support.microsoft.com/kb/321051.

   a. Install a certificate authority in the **Windows Components** section in **Add/Remove Programs**.

   b. Select the **Enterprise Root CA** option.

   c. Reboot the Active Directory server. If IIS web services are running, the CA certificate can be accessed by opening **http://***servername***/certsrv**.

   d. Set up the Active Directory server to use the SSL server certificate.

      a. Create a certificate request **.inf**, using the fully-qualified domain name of the Active Directory as the certificate subject. For example:

```
;----------------- request.inf -----------------

[Version]

Signature="$Windows NT$

[NewRequest]

Subject = "CN=ad.server.example.com, O=Engineering,
L=Raleigh, S=North Carolina, C=US"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = False
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic
Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0

[EnhancedKeyUsageExtension]

OID=1.3.6.1.5.5.7.3.1

;-----------------------------------------------
```

For more information on the **.inf** request file, see the Microsoft documentation, such as http://technet.microsoft.com/en-us/library/cc783835.aspx.

b. Generate the certificate request.

```
certreq -new request.inf request.req
```

c. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```

NOTE

If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is **http://***servername***/certsrv**.

d. Accept the certificate request. For example:

```
certreq -accept certnew.cer
```

e. Make sure that the server certificate is present on the Active Directory server.

In the **File** menu, click **Add/Remove**, then click **Certificates** and **Personal>Certificates**.

f. Import the CA certificate from Directory Server into Active Directory. Click **Trusted Root CA**, then **Import**, and browse for the Directory Server CA certificate.

e. Reboot the domain controller.

7.6.2. Setting up Password Synchronization

Install the Password Sync Service on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

1. Download the **`PassSync.msi`** file to the Active Directory machine.

    a. Log into the Customer Portal.

    b. Click the **`Downloads`** tab.

    c. Click the **`Red Hat Enterprise Linux`** downloads button in the middle of the page.

    d. Filter the downloads by using a search term such as *Directory Server*, and then expand one of the Red Hat Enterprise Linux versions.

    e. Click the Directory Server link.

    f. On the Directory Server page, download the appropriate version of the WinSync Installer. This is the Password Sync MSI file (**`RedHat-PassSync-1.1.5-`*`arch`*`.msi`**).

> **NOTE**
>
> Regardless of the Red Hat Enterprise Linux architecture, there are two PassSync packages available, one for 32-bit Windows servers and one for 64-bit. Make sure to select the appropriate packages for your Windows platform.

2. Double-click the Password Sync MSI file to install it.

3. The **`Password Sync Setup`** window appears. Hit **`Next`** to begin installing.

4. Fill in the information to establish the connection to the IdM server.

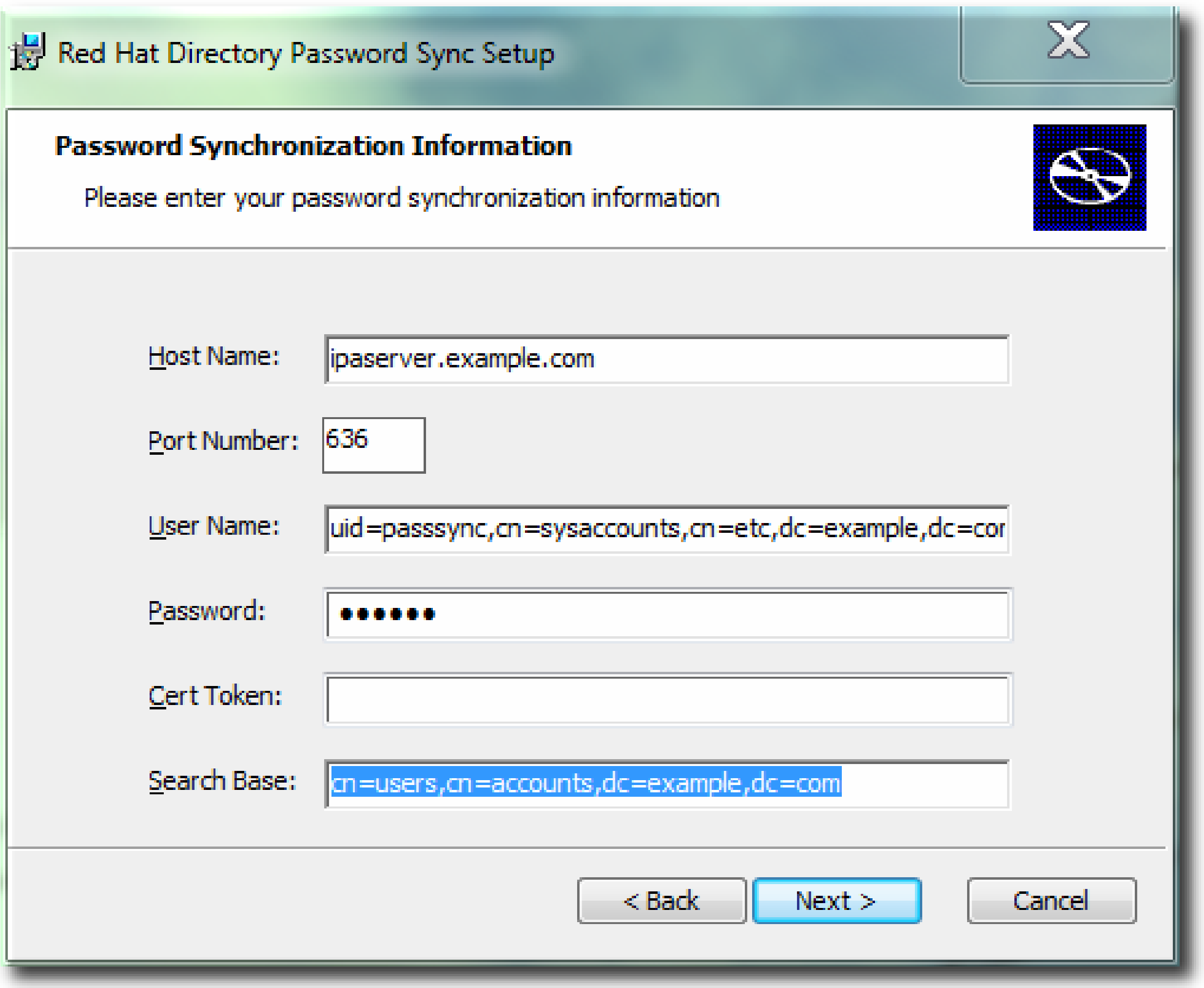
    - The IdM server connection information, including the hostname and secure port number.

    - The username of the system user which Active Directory uses to connect to the IdM machine. This account is configured automatically when sync is configured on the IdM server. The default account is **`uid=passsync,cn=sysaccounts,cn=etc,dc=example,dc=com`**.

    - The password set in the **`--passsync`** option when the sync agreement was created.

    - The search base for the people subtree on the IdM server. The Active Directory server connects to the IdM server similar to an **`ldapsearch`** or replication operation, so it has to know where in the IdM subtree to look for user accounts. The user subtree is **`cn=users,cn=accounts,dc=example,dc=com`**.

    - The certificate token is not used at this time, so that field should be left blank.



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 Sync.

5. Import the IdM server's CA certificate into the Active Directory certificate store.

   a. Download the IdM server's CA certificate from **http://ipa.example.com/ipa/config/ca.crt**.

   b. Copy the IdM CA certificate to the Active Directory server.

   c. Install the IdM CA certificate in the Password Sync database. For example:

   ```
   cd "C:\Program Files\Red Hat Directory Password
   Synchronization"

   certutil.exe -d . -A -n "IPASERVER.EXAMPLE.COM IPA CA" -t
   CT,, -a -i ipaca.crt
   ```

6. Reboot the Windows machine to start Password Sync.

NOTE

The Windows machine must be rebooted. Without the rebooting, **PasswordHook.dll** is not enabled, and password synchronization will not function.

7. If passwords for existing accounts should be synchronized, reset the user passwords.

NOTE

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

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

The first attempt to synchronize passwords, which happened when the Password Sync application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory sync peers. The tools to create the certificate and key databases is installed with the **`.msi`**.

7.6.3. Allowing Users to Change Other Users' Passwords Cleanly

By default, every time an administrator changes a user password, that user is required to reset the password at the next login. However, this behavior can be changed to allow administrators to reset a password *without* requiring an immediate password reset.

The ***`passSyncManagersDNs`*** attribute lists administrator accounts which are allowed to perform password change operations *and* which will not then require a password reset.

IMPORTANT

This is required for password synchronization because, otherwise, whenever a password is synchronized, the IdM server would interpret that as a password change operation and then require a password change at the next login.

Edit the password synchronization entry, **`cn=ipa_pwd_extop,cn=plugins,cn=config`**, and add the ***`passSyncManagersDNs`*** attribute with the name of the user. This attribute is multi-valued. For example:

```
$ ldapmodify -x -D "cn=Directory Manager" -w secret -h ldap.example.com
-p 389

dn: cn=ipa_pwd_extop,cn=plugins,cn=config
changetype: modify
add: passSyncManagersDNs
passSyncManagersDNs: uid=admin,cn=users,cn=accounts,dc=example,dc=com
```

WARNING

Be careful to limit the listed 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.

[6] The ***cn*** is treated differently than other synced attributes. It is mapped directly (***cn*** to ***cn***) when syncing from Identity Management to Active Directory. When syncing from Active Directory to Identity Management, however, ***cn*** is mapped from the ***name*** attribute on Windows to the ***cn*** attribute in Identity Management.

Chapter 8. ID Views and Migrating Existing Environments to Trust

The *ID Views* mechanism that is part of Red Hat Identity Management enables you to specify POSIX attributes for users or groups. When you create a new ID view, you can define what user or group attributes it should override; these newly defined attributes are then applied to the user or group. By allowing this, ID views provide a solution to preserve existing environments during migration from other identity management and system integration solutions.

After running the **`ipa-adtrust-install`** command, the Default Trust View is created. The Default Trust View is always applied to Active Directory users and groups, which allows you to define POSIX attributes for AD users and groups regardless of how AD itself defined them. If you add a host-specific ID view that overrides the AD users or groups, the attributes from the host-specific ID view are applied on top of the Default Trust View. While the new ID view overrides the Default Trust View, you cannot delete the default view itself. If no specific ID view is applied to a client, the Default Trust View always applies.

Note

If you do not run **`ipa-adtrust-install`**, you can still use the ID Views feature in a pure IdM environment to manage ID views and overrides for IdM users.

In a setup with a synchronization-based AD integration, all users are copied to the IdM server with generated POSIX attributes, such as login name, UID, GID, or shell. As explained in Section 1.3, “Indirect Integration”, the synchronization-based approach is generally discouraged, and it is recommended to use the trust-based approach instead. By enabling the administrator to modify the POSIX attributes that AD previously generated for AD users, the ID Views feature provides a solution to migrate existing environments to a trust-based AD integration.

Use cases covered by ID Views include:

Store POSIX attributes and SSH data for AD users

Define POSIX attributes or SSH keys and SSH login information for AD users, and let them be applied when an AD user authenticates to clients running SSSD with ID Views support or when the AD user authenticates using a *compat LDAP tree*, which offers a simplified LDAP tree with user and group data for legacy clients.

This capability is useful for migration from a synchronization-based solution or in a situation when a Linux administrator would like to manually define POSIX attributes for AD users but the AD policy does not allow it.

Migrate from a synchronization-based to a trust-based integration

Configure the POSIX attributes for users that are in a synchronization-based environment by creating an ID view override specifying previously used UID or other tools. Then move the users back to AD.

Perform per-host group override of the IdM user POSIX attributes

NIS-based infrastructure that is being migrated to an IdM integration with AD still often requires that the original POSIX data remain unchanged on some NIS domains or the company policies might prevent setting the original POSIX data in AD directly. In these situations, you can use ID Views to configure the POSIX data directly on the Identity Management server.

Set different POSIX attributes or SSH data for different environments

Set different POSIX attributes or different user SSH public keys for different production environments – such as development, testing, or production – depending on the corresponding host groups.

8.1. User Overrides and Group Overrides

Every *ID view* is a collection of *user overrides* and *group overrides* that applies to specified hosts. An override provides a new user or group attribute that overrides the previous one; this enables you to, for example, replace a previously generated attribute with a new one. Every override is related to an AD or IdM user or group.

Note

Non-IdM integration systems can generate the UID and GID attributes using an algorithm that is different from the algorithm used in IdM. By overriding the previously generated attributes so that they are in compliance with the IdM system, a client that used to be a member of another integration system can be fully integrated with IdM.

The following user attributes can be overridden in an ID view:

- **`uid`**: user login name
- **`uidNumber`**: user UID number
- **`gidNumber`**: user GID number
- **`loginShell`**: user login shell
- **`gecos`**: user GECOS entry
- **`homeDirectory`**: user home directory
- **`ipaSshPubkey`**: user SSH public key or keys

The following group attributes can be overridden in an ID view:

- **`cn`**: group name
- **`gidNumber`**: group GID number

Note

IdM uses *ID ranges* to avoid collisions of POSIX IDs from different domains. 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. If a collision occurs, it can be easily fixed by changing the conflicting IDs because POSIX IDs are managed manually in ID Views on the IdM side.

8.2. Managing ID Views

ID views can be added, modified, or deleted. You can define what ID attributes an ID view should override and to which client hosts it should apply.

For AD users, overrides from the Default Trust View are always applied. If an ID view assigned to the host overrides the values present in the Default Trust View or the original values from AD for some attributes, these overridden values are visible on the host. If an ID view does not override a value present in the Default Trust View, all clients assigned to this other ID view see the Default Trust View value.

Default Trust View accepts only AD user overrides; you cannot add overrides of IdM users or groups to the Default Trust View. For IdM users, the default view is represented by the values defined in their corresponding IdM user records.

IdM servers and replicas always apply the Default Trust View without any ID view overrides; you cannot assign a different ID view to them. The default view is also always applied to AD users or groups.

8.2.1. ID Views and SSSD

If the administrator applies another ID view on a client, the client and all the other clients applying this ID view must restart the SSSD service. Moreover, if the new ID view changes a UID or GID, the client and all the other clients applying the ID view must clear the SSSD cache.

Note

Applying an ID view can have a negative impact on SSSD performance because certain optimizations and ID views cannot run at the same time.

For example, ID views prevent SSSD from optimizing the process of looking up groups on the server. With ID views, SSSD must check every member on the returned list of group member names if the group name is overridden. Without ID views, SSSD can only collect the user names from the member attribute of the group object. This negative effect will most likely become apparent when the SSSD cache is empty or when all entries are invalid, that is, after clearing the cache.

ID views are applied on the client side, which means that clients running earlier versions of IdM only see the Default Trust View. If a client requires a different ID view, update SSSD on the client to a version with ID View support or have the client use the compat LDAP tree.

8.2.2. Managing ID Views from the Web UI

To manage ID views from the IdM Web UI, open the **`IPA Server`** main tab and then select the **`ID Views`** subtab.

To add a new ID view:

1. Click **`Add`** above the list of all ID views.

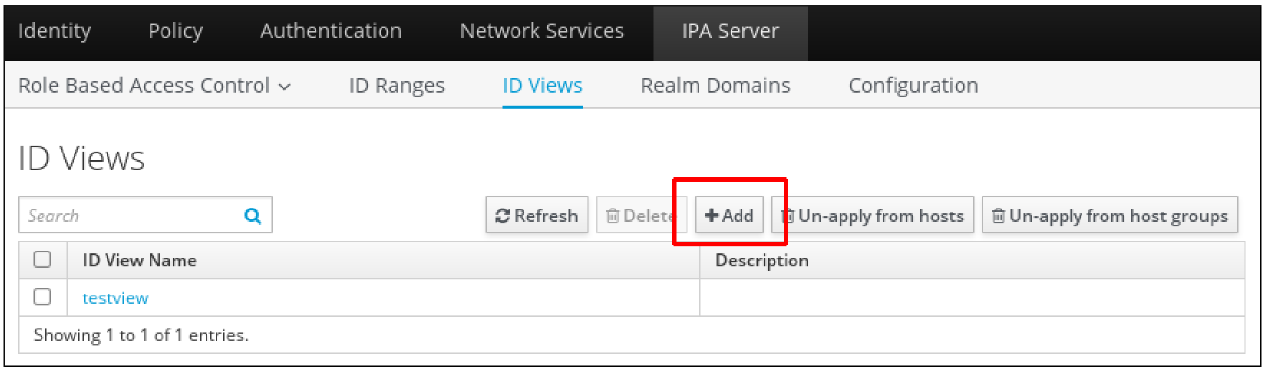


Figure 8.1. Adding a New ID View

2. Fill out the information about the new ID view in the form that shows up.

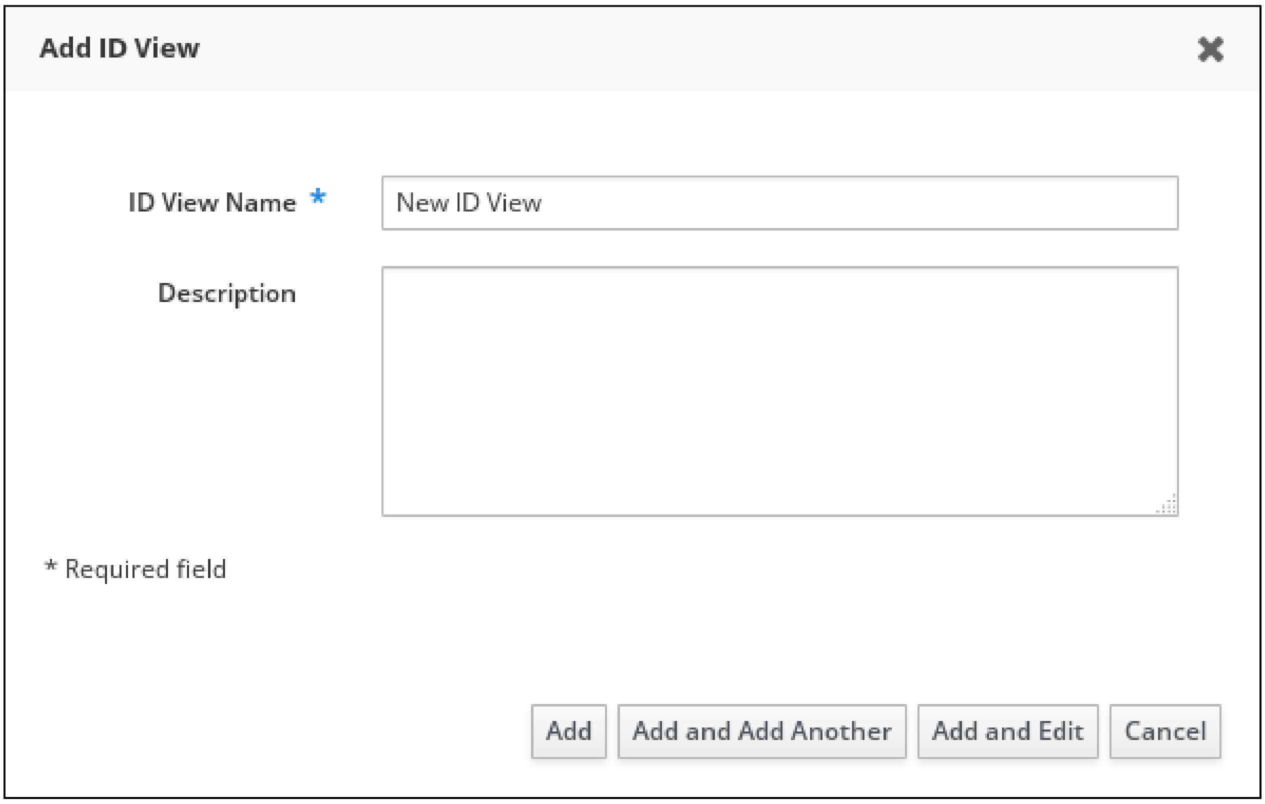
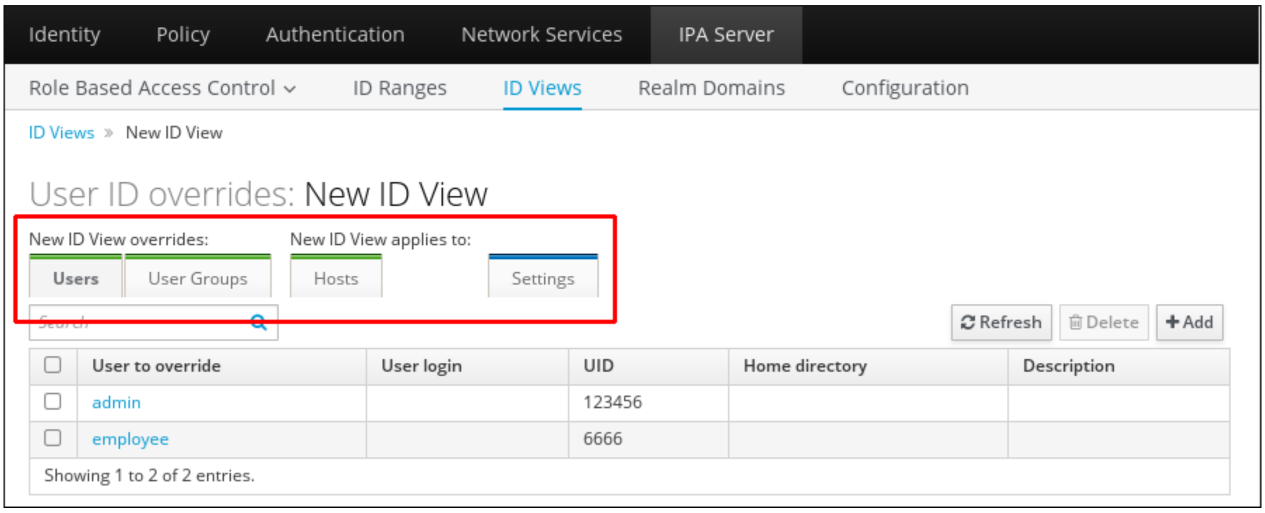


Figure 8.2. Form for Adding a New ID View

3. Click the **Add** button under the form.

To define the properties of an ID view:

1. Click on the name of the ID view in the list of ID views, and then choose the appropriate tab.

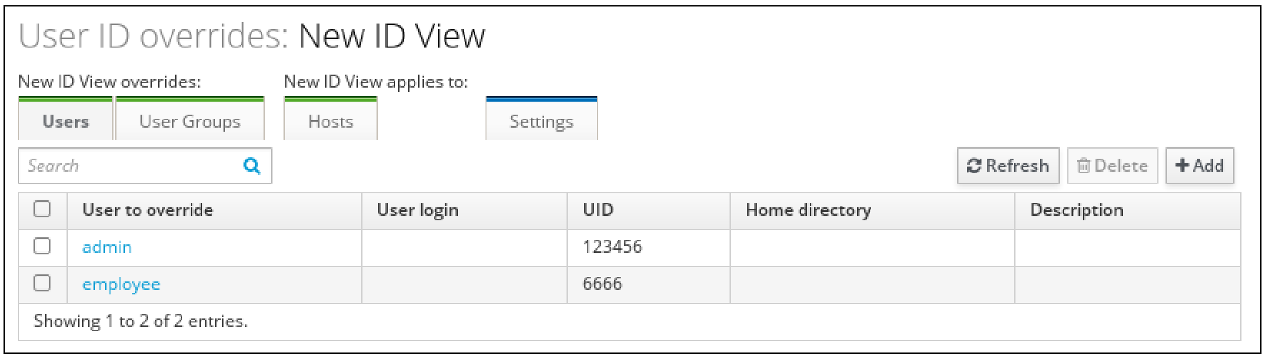


	User to override	User login	UID	Home directory	Description
☐	admin		123456		
☐	employee		6666		

Showing 1 to 2 of 2 entries.

Figure 8.3. ID View Tabs

2. **Users** shows the list of users whose user attributes the ID view overrides.

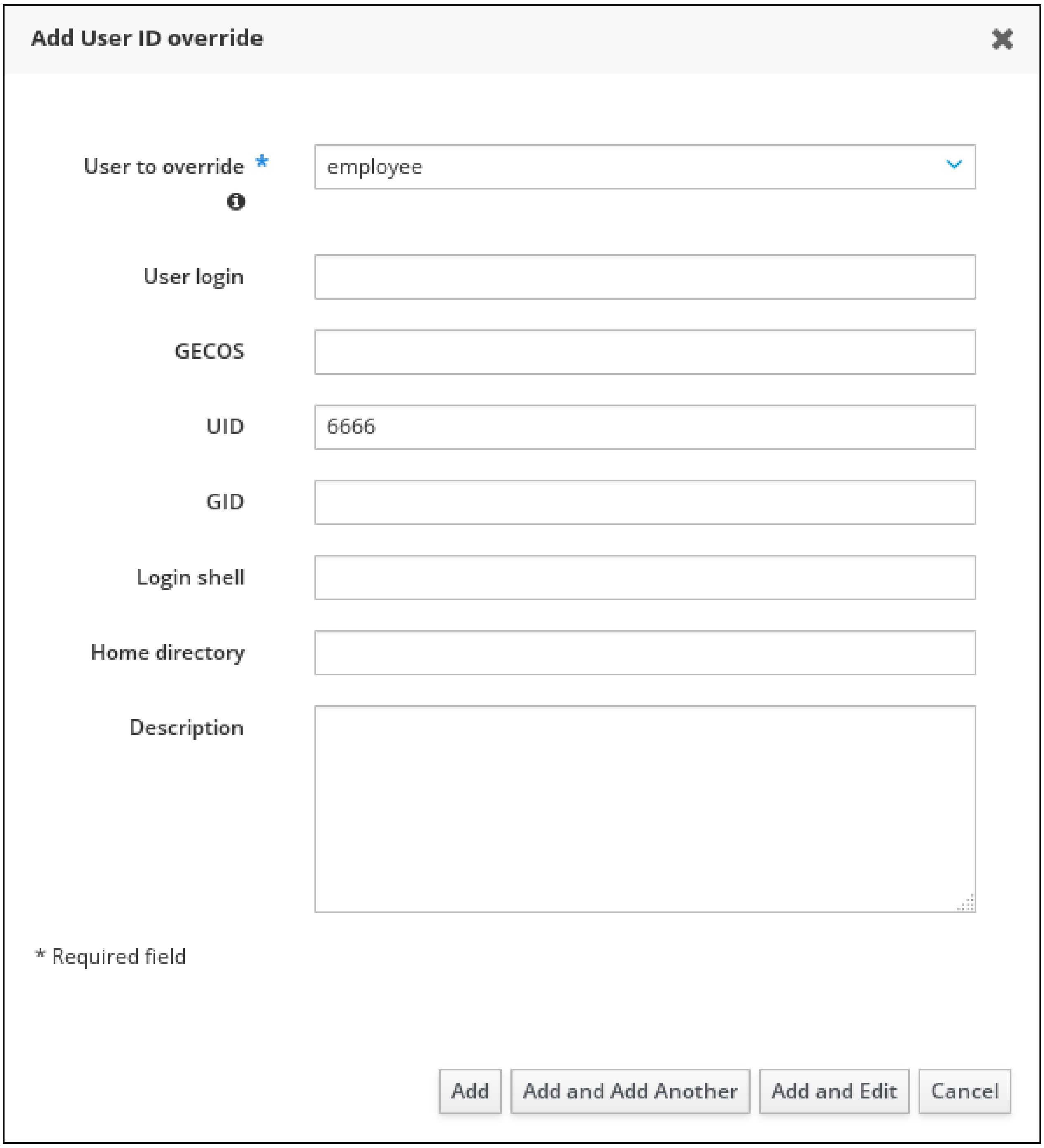


	User to override	User login	UID	Home directory	Description
☐	admin		123456		
☐	employee		6666		

Showing 1 to 2 of 2 entries.

Figure 8.4. Adding a User Override

Click **Add** to create a new user override; you will be asked to fill out the new values for the user attributes.


Add User ID override

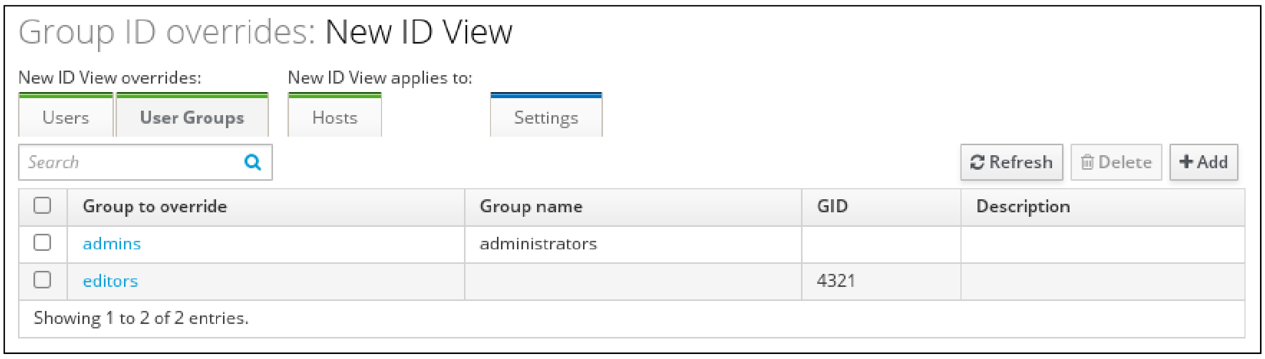
User to override *	employee
User login	
GECOS	
UID	6666
GID	
Login shell	
Home directory	
Description	

* Required field

Figure 8.5. Adding a User Override

Click **`Delete`** to remove selected user overrides.

3. **`User Groups`** shows the list of user groups whose group attributes the ID view overrides.

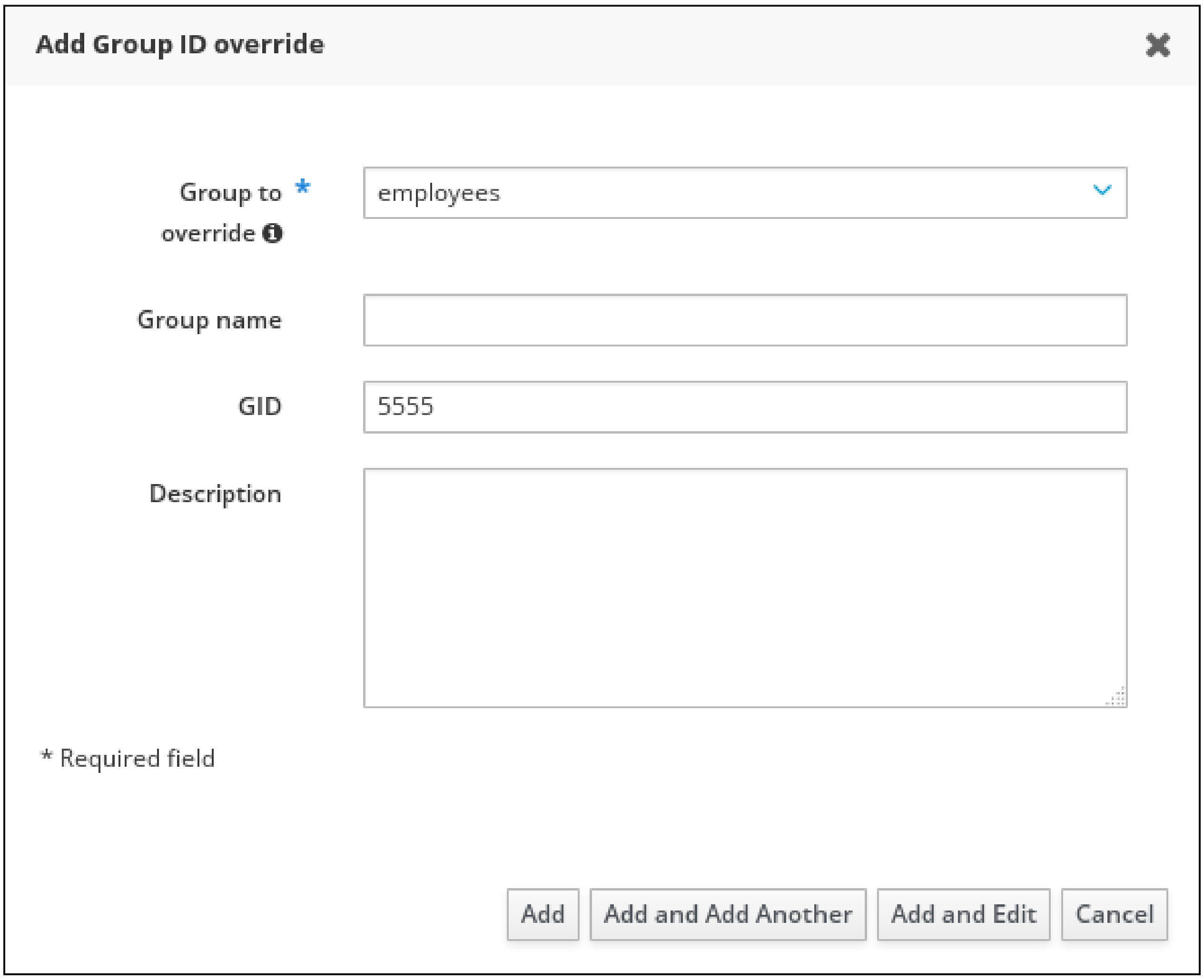

Group ID overrides: New ID View

	Group to override	Group name	GID	Description
☐	admins	administrators		
☐	editors		4321	

Showing 1 to 2 of 2 entries.

Figure 8.6. User Groups Tab

Click **Add** to create a new user group override; you will be asked to fill out the new values for the group attributes.



Add Group ID override

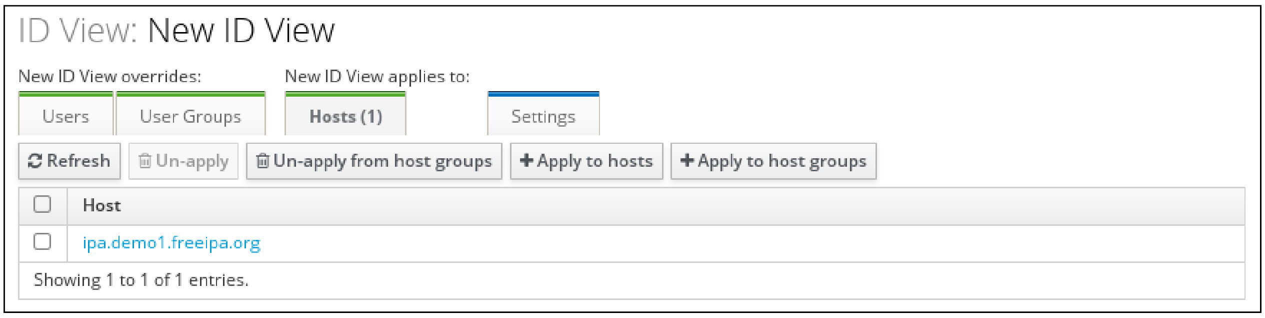
Group to override *	employees
Group name	
GID	5555
Description	

* Required field

Figure 8.7. Adding a Group Override

Click **Delete** to remove selected user group overrides.

4. **Hosts** shows the list of hosts or host groups to which the ID view applies.



ID View: New ID View

New ID View overrides: New ID View applies to:

	Host
☐	ipa.demo1.freeipa.org

Showing 1 to 1 of 1 entries.

Figure 8.8. Hosts Tab

Click **`Apply to hosts`** or **`Apply to host groups`** to add a new host or to add hosts belonging to a host group. In the form that shows up, move the required hosts or hosts group from the **`Available`** to **`Prospective`** column and click **`Apply`**.

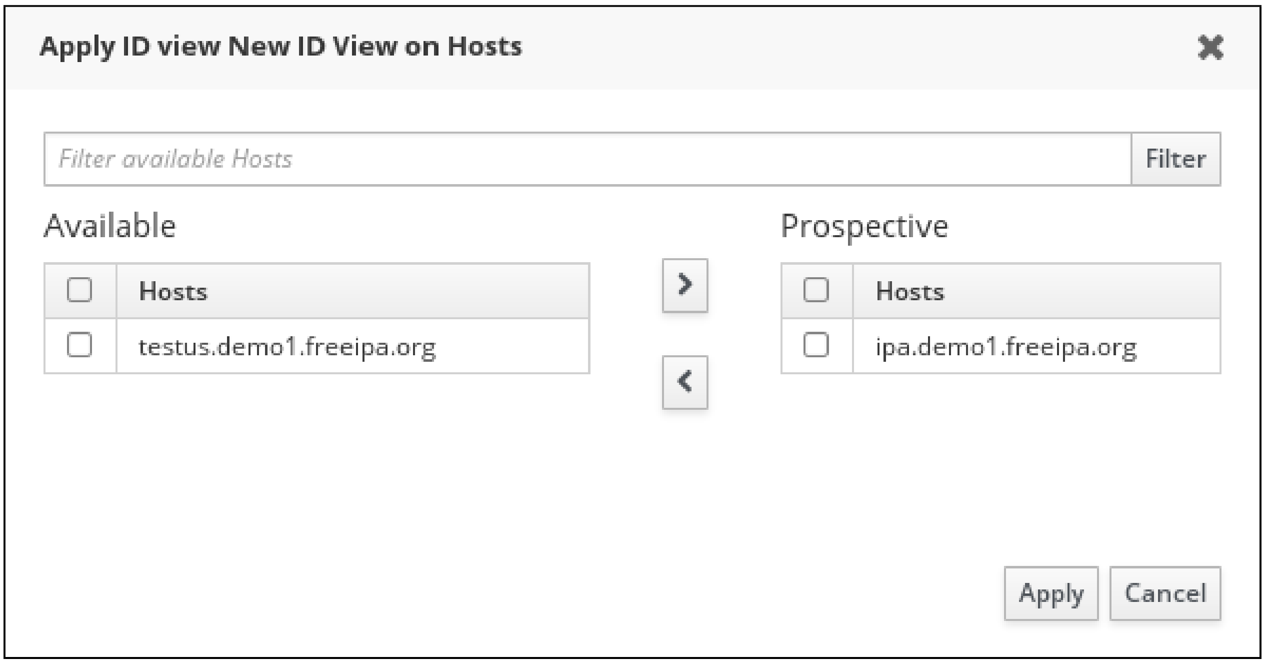


Figure 8.9. Applying an ID View to a Host

`Un-apply` removes the ID view from specified hosts. **`Un-apply from host groups`** enables you to remove the ID view from specified host groups.

5. **`Settings`** enables you to modify the ID view description.

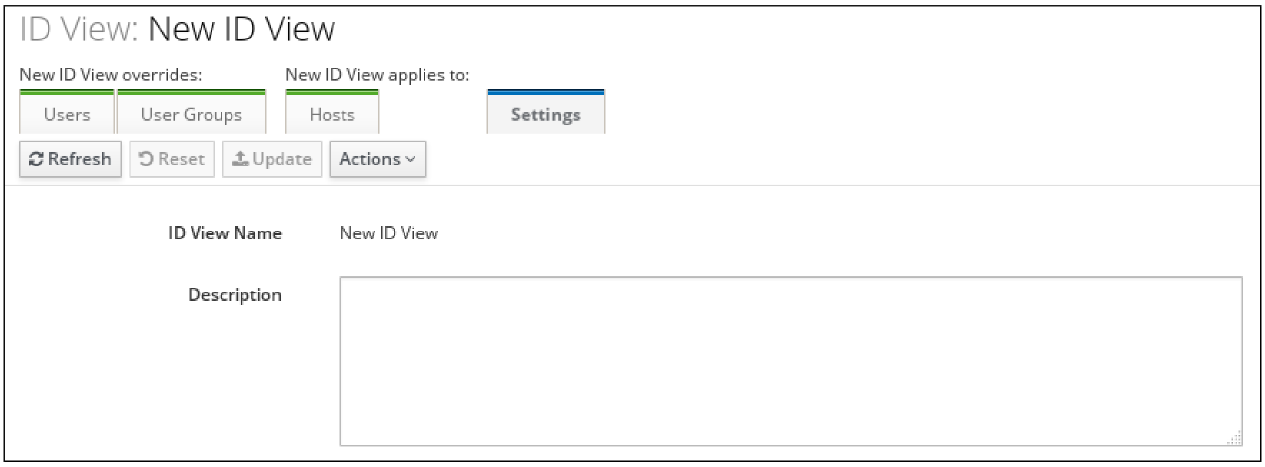


Figure 8.10. Settings Tab

8.2.3. Managing ID Views from the command line

To manage ID views on the command line, use the following commands:

- **`ipa idview-add`** adds a new ID view
- **`ipa idview-apply`** applies an ID view to specified hosts or host groups; any previously applied ID view is overridden

- **`ipa idview-del`** deletes an ID view
- **`ipa idview-find`** searches for a specified ID view
- **`ipa idview-mod`** modifies an ID view
- **`ipa idview-show`** displays information about an ID view
- **`ipa idview-unapply`** removes an ID view from specified hosts or host groups

To manage group and user ID overrides, use the following commands:

- **`ipa idoverridegroup-add`** adds a new group ID override

  `ipa idoverrideuser-add` adds a new user ID override
- **`ipa idoverridegroup-del`** deletes a group ID override

  `ipa idoverrideuser-del` deletes a user ID override
- **`ipa idoverridegroup-find`** searches for a specified group ID override

  `ipa idoverrideuser-find` searches for a specified user ID override
- **`ipa idoverridegroup-mod`** modifies a group ID override

  `ipa idoverrideuser-mod` modifies a user ID override
- **`ipa idoverridegroup-show`** displays information about a group ID override

  `ipa idoverrideuser-show` displays information about a user ID override

For detailed information on what options can be passed to these commands, see the corresponding man pages or run one of them with the **`--help`** option added.

Example 8.1. Storing POSIX Attributes and SSH Keys for AD Users Using a Host-Specific ID View

To change the UID of the **`testuser`** user to 6666:

1. Add a new host-specific ID view using **`ipa idview-add`** and supply the required values.

```
[user@client ~]$ ipa idview-add testview --desc "Our new host-
specific view"
---------------------------------------------
Added ID View "testview"
---------------------------------------------
  ID View Name: testview
  Description: Our new host-specific view
```

2. Add an ID override to the ID view by running **`ipa idoverrideuser-add`** and supplying the required values.

```
[user@client ~]$ ipa idoverrideuser-add testview
testuser@example.com --uid 6666
---------------------------------------------
Added User ID override "testuser@example.com"
```

```
--------------------------------------------
  Anchor to override: testuser@example.com
  UID: 6666
```

3. Apply the ID view to a specific host by running **`ipa idview-apply`** and supplying the host using the **`--hosts`** option.

```
[user@client ~]$ ipa idview-apply testview --hosts
examplehost.com
--------------------------------------------
Applied ID View "testview"
--------------------------------------------
  hosts: examplehost.com
--------------------------------------------
Number of hosts the ID View was applied to: 1
```

You can override the GID and other attributes using a similar procedure. For more information, run the **`ipa idoverrideuser-add --help`** command.

Note

The **`--hostgroups`** option applies the ID view to hosts belonging in a specified host group and can be used in the same way as the **`--hosts`** option. The **`--hostgroups`** option does not associate the ID view with the host group itself; it expands the members of the specified host group and applies **`--hosts`** individually to every one of them.

8.3. Migrating from the Synchronization-Based to the Trust-Based Solution

In an environment that uses the synchronization-based integration, you can migrate to the trust-based integration by following these steps:

1. Create a trust with the synchronized domain. For information about creating trusts, see Chapter 5, *Creating Cross-Realm Trusts with Active Directory and Identity Management*.

2. For every synchronized user or group, individually create an ID override in a host-specific view or in the Default Trust View to preserve the UID and GID generated by IdM. For an example on how to do this, see Example 8.1, “Storing POSIX Attributes and SSH Keys for AD Users Using a Host-Specific ID View”.

3. Make a backup copy of the original synchronized user or group entry.

4. Delete all the original synchronized user or group entries.

Index

Revision History

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Enterprise Linux.

Revision	Date	Author
Revision 7.0-16	**Thu Apr 02 2015**	**Tomáš Čapek**
Async update to include ipa-advise, additional information on accessing a CIFS share with SSSD, and admonition for the Identity Management for UNIX extension.		
Revision 7.0-15	**Fri Mar 13 2015**	**Tomáš Čapek**
Async update with last-minute edits for 7.1.		
Revision 7.0-13	**Wed Feb 25 2015**	**Tomáš Čapek**
Version for 7.1 GA release.		
Revision 7.0-11	**Fri Dec 05 2014**	**Tomáš Čapek**
Rebuild to update the sort order on the splash page.		
Revision 7.0-7	**Mon Sep 15 2014**	**Tomáš Čapek**
Section 5.3 Creating Trusts temporarily removed for content updates.		
Revision 7.0-5	**June 27, 2014**	**Ella Deon Ballard**
Improving Samba+Kerberos+Winbind chapters.		
Revision 7.0-4	**June 13, 2014**	**Ella Deon Ballard**
Adding Kerberos realm chapter.		
Revision 7.0-3	**June 11, 2014**	**Ella Deon Ballard**
Initial release.		