freeIPA 1.2.1

Administration Reference

IPA Solutions from the IPA Experts

freeIPA 1.2.1 Administration Reference
IPA Solutions from the IPA Experts
Edition 1.0

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

1801 Varsity Drive
Raleigh, NC 27606-2072
USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park, NC 27709
USA

This reference provides detailed information about IPA, the technologies with which it works, and some of the terminology used to describe it.

Preface

Welcome to the IPA Administration Reference. This reference provides detailed information about IPA servers and clients, their supporting technologies, and the tools and services required to use and manage them. It also includes conceptual information about the technologies that comprise IPA, and how they work together.

1. Audience

This reference is intended for system administrators and those responsible for ensuring that IPA is installed and configured correctly for a particular deployment. It is also intended for those responsible for customizing or extending any aspects of IPA to suit the needs of a particular installation.

The IPA Administration Reference assumes a good understanding of various operating systems, including `Linux`, `Solaris` and other UNIX systems, `Macintosh` and Microsoft `Windows`. It also assumes a working knowledge of LDAP and either Red Hat or Fedora Directory Server.

2. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

2.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file **`my_novel`** in your current working directory, enter the **`cat my_novel`** command at the shell prompt and then press **`Enter`**.

The above example includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**-**`Alt`**-**`F1`** to switch to the first virtual terminal. Press **`Ctrl`**-**`Alt`**-**`F7`** to return to your X-Windows session.

[1] https://fedorahosted.org/liberation-fonts/

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`Mono-spaced Bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This avoids the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh username@domain.name`** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> To see the version of a currently installed package, use the **`rpm -q package`** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new or important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

2.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

2.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A Note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning
A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

3. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *https://bugzilla.redhat.com/enter_bug.cgi?product=freeIPA* against the Documentation component.

When submitting a bug report, be sure to mention the manual's identifier: *Administrators_Reference*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Introduction to IPA

IPA is an integrated solution which combines the following technologies:

- Fedora (server-side)
- Fedora Directory Server
- MIT Kerberos™
- NTP
- DNS
- Web and command-line provisioning and administration tools

The architecture of an IPA server can be represented as follows:

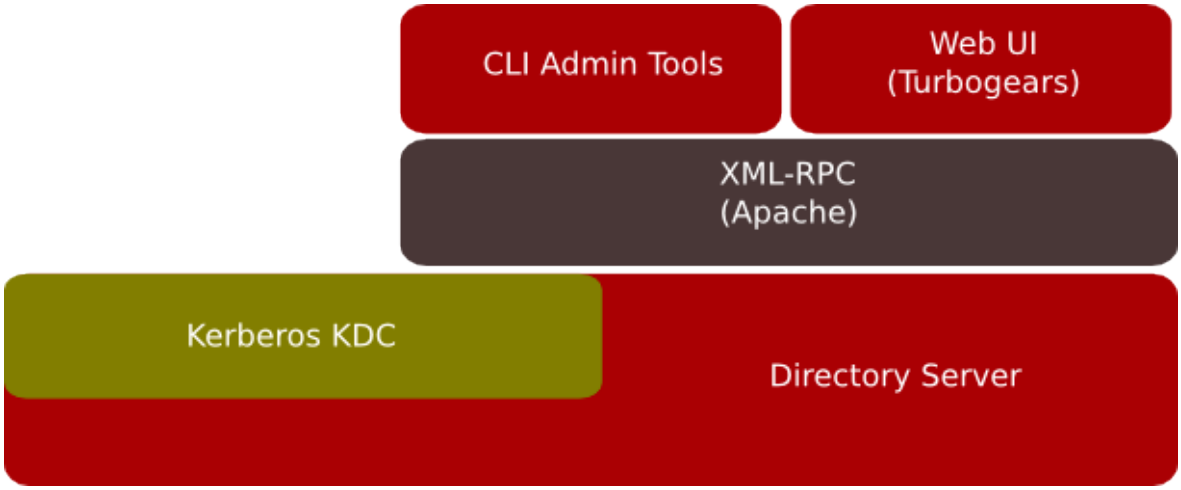

Figure 1.1. Architecture of an IPA server.

1.1. IPA and Directory Server

Fedora Directory Server is an open source, LDAP-based directory service, which provides an LDAP server, a web management interface, and command-line and graphical management tools. It is highly scalable, and supports a number of features including:

- Multi-master replication
- TLS/SSL and SASL security
- Support for custom plug-in extensions
- Online schema and configuration updates over LDAP
- Internationalized entries
- Optional on-disk encryption of selected attributes
- Virtual DIT views

Directory Server consists of several different components. The core directory server, *ns-slapd*, consists of a front end which handles network communications, extensible plug-ins which handle basic server functions, and a database back-end which implements an indexed, transactional store on top of a Berkeley DB.

1.1.1. How IPA and Directory Server Work Together

Directory Server is an integral part of IPA. In IPA, the Directory Server functions as the data store, maintaining all of an organization's information. The Directory Server's internal controls restrict the level of access that IPA users have to Directory Server information. These internal controls cannot be overridden by any IPA permissions, delegations, or other controls.

1.2. IPA and Kerberos

Kerberos is a network authentication protocol created by MIT, and uses symmetric-key cryptography to authenticate users to network services. This means that passwords are never actually sent over the network.

Consequently, when users authenticate to network services using Kerberos, unauthorized users attempting to gather passwords by monitoring network traffic are effectively thwarted.

Kerberos' primary design goal is to eliminate the transmission of unencrypted passwords over the network. If used properly, Kerberos effectively eliminates the threat that packet sniffers would otherwise pose on a network.

1.2.1. How IPA and Kerberos Work Together

The IPA implementation of Kerberos differs from a typical Kerberos implementation, mainly in that it uses Directory Server to store data instead of a flat file. The Directory Server also provides access controls to protect data. The IPA Kerberos implementation does not use the native Kerberos tools because by default the KDC is not aware of the Directory Server.

IPA provides its own set of tools for working with Kerberos. Kerberos' native tools, for example, those provided by **kadmin.local**, should not be used. For many reasons, **kadmin.local** does not currently have permission to operate outside of *cn=kerberos*. One reason is that it cannot communicate using LDAP, which means that you cannot create principals anywhere that you might want. Another reason is that it cannot proxy a password change to the IPA plug-in. Consequently, if a password change is performed via **kadmin.local**, it would corrupt the user entry.

Warning

The **kadmin** and **kadmin.local** tools are not supported in IPA, and it is highly recommended that you avoid their use.

1.2.2. IPA, Kerberos, and Service Principals

Server programs require *Service Principals* to perform Kerberos authentication. Kerberos authentication works by obtaining an encrypted ticket for a service, a ticket that only the service can decrypt. This in turn verifies that the user obtained it from the KDC, indirectly proving that the client was able to authenticate to the KDC and is therefore trustworthy.

The client needs the Service Principal to indicate to the KDC which service it needs a ticket for. The KDC uses the Service Principal to store and provide a secret to the service at the moment the Service Principal is created.

Service Principals are typically released per service, although it is possible for one Service Principal to be used for more services. For example, host/<fqdn>@REALM is used for both the SSH service and also as the generic "host" principal.

1.2.3. IPA, Kerberos, and DNS

As discussed in *Section 1.4.1, "How IPA and DNS Work Together"*, IPA relies heavily on a fully-functional DNS for correct operation. Because of its tight integration with IPA, Kerberos also requires that the DNS be configured correctly.

1.2.3.1. Using CNAME and A Records

When Kerberos requests a ticket to begin authentication, it always resolves a CNAME to its corresponding A record; Kerberos libraries never use a CNAME to request a ticket. This means that when you create service or host principals you need to use the host A record. Consider the following zone file entry:

```
CNAME www.example.com -> A name web-01.example.com
```

If you use the following command to connect to the host via SSH, and want GSSAPI authentication:

```
$ ssh www.example.com
```

it will actually request a ticket for host/web-01.example.com@EXAMPLE.COM

This is the service principal that you must use to obtain and save tickets in **/etc/krb5.keytab** for this host.

1.3. IPA and NTP

Many computer services (for example, Kerberos) require that the time differential between hosts on a network be kept to a minimum for correct or accurate operation. The Network Time Protocol (NTP) is a protocol used to synchronize computer clocks over the network. Most operating systems can be configured to synchronize their clocks with any of a number of time servers.

1.3.1. How IPA and NTP Work Together

IPA combines a number of different technologies, many of which constantly communicate with each other over the network. In order for these technologies to work together correctly, the time differential between the clients and servers on the network must be kept to a minimum.

Kerberos, for example, only tolerates a five minute time difference between the KDC and a client requesting authentication. A time difference greater than five minutes will result in a failed authentication.

System Administrators also rely on accurate time keeping for correlation of system logs across machines on the network. In the event of problems on the network or other aspects of a deployment, it

may be necessary to inspect the log files of various machines to determine if specific problems occur at the same time. Without NTP or another time synchronization system, such troubleshooting would be all but impossible.

The IPA startup process ensures that the NTP service starts and that the time and date are synchronized before any other IPA-related processes start. This avoids problems with certificates, LDAP entry creation dates, password and account expiration dates, and other date-related issues.

1.4. IPA and DNS

A Domain Name Service (DNS) associates host names with their respective IP addresses. This enables users to refer to networked machines by name, rather than having to remember IP addresses.

DNS is normally implemented using centralized servers that are authoritative for some domains and refer to other DNS servers for other domains.

1.4.1. How IPA and DNS Work Together

IPA clients find, or *discover*, IPA servers using a process known as *Service Discovery*. This can occur automatically, using DNS, or manually, by entering the IPA server details during the client configuration phase.

1.4.1.1. Service Discovery using DNS

The recommended method for ensuring that IPA clients discover IPA servers is to use DNS. This requires adding special records to the DNS configuration. The IPA server installation generates a sample zone file which contains sample records for this purpose. In particular, it includes records for the LDAP servers, Kerberos realm, and Kerberos servers required in an IPA deployment. These records can be added to an existing DNS infrastructure - even one hosted on a different OS - or they can be added to a new DNS server if one does not already exist.

The following is an extract from a zone file where the IPA server, KDC, and DNS server all exist on the same machine (ipaserver) in the realm EXAMPLE.COM:

```
; ldap servers
_ldap._tcp              IN SRV 0 100 389        ipaserver

;kerberos realm
_kerberos               IN TXT EXAMPLE.COM

; kerberos servers
_kerberos._tcp          IN SRV 0 100 88         ipaserver
_kerberos._udp          IN SRV 0 100 88         ipaserver
_kerberos-master._tcp   IN SRV 0 100 88         ipaserver
_kerberos-master._udp   IN SRV 0 100 88         ipaserver
_kpasswd._tcp           IN SRV 0 100 464        ipaserver
_kpasswd._udp           IN SRV 0 100 464        ipaserver
```

If you already have DNS configured on your network, you can add this to the existing zone file (in this example, **/var/named/example.com.zone.db**) so that IPA clients can discover the LDAP and Kerberos servers.

Clients try to discover the IPA server first using parameters passed via the command line, then using the configuration file (**/etc/ipa/ipa.conf**), and then via DNS.

1.4.1.2. Service Discovery without using DNS

It is possible to implement IPA without a DNS, but this is not recommended. If you do not use DNS for service discovery, then your clients will not automatically find other IPA services in a high-availability setup.

During an IPA client installation, if the DNS is not configured so that the client can discover the IPA server, you must enter the appropriate information manually.

1.4.2. Using IPA with Multi-Homed Machines

Some of the machines in your IPA deployment may support multiple Network Interface Cards (NICs). This is often the case for servers or other machines where high availability or failover is required. These multi-homed machines typically have multiple IPs assigned to the same host name. Normally this will not be a problem for IPA, because it listens to all available interfaces except `localhost`. Similarly for the KDC; it only listens to the machine's public IP addresses. It does not listen to the `localhost` interface.

For an IPA server that has multiple NICs and IP addresses, you need to configure the DNS with the appropriate A records if the server is to be available via any interface.

For example, the zone file on the DNS server might have the following A records for an IPA server (`ipaserver.example.com`) with three NICs:

```
ipaserver  IN A  192.168.1.100
ipaserver  IN A  192.168.1.101
ipaserver  IN A  192.168.1.102
```

This allows IPA clients to discover the IPA server using any of the available network interfaces.

1.5. Password Management in IPA

IPA manages user passwords as part of its function as an identity management store. The first time a password is set, or whenever a password is reset, its status is immediately set to "expired". New and reset passwords are referred to as *initial passwords.* Account owners are required to change these initial passwords before they can use their account. This behavior is built in to IPA to help address common security issues, and cannot be changed.

Basic Password Requirements

The fundamental requirement of a password is that it be known only by the user authorized to use it. If password authentication is the only authentication method in use, the user's password essentially comprises their entire identity. Consequently, only the authorized user should know the password.

Whenever an administrator creates an initial password for a user account, this requirement is not satisfied. At least two people now know the password, which renders it ineffective as a means of exclusively identifying a single user. It is important to remedy this situation as soon as possible.

Setting the status of initial passwords to "expired" forces users to remedy this situation before they begin using their account, and reduces the threat of impersonation.

Password Distribution

Password distribution refers to the transmission of initial passwords from the administrator to the account holder. Whenever an administrator creates an initial password for a user account, not only does the administrator now know the password for the user account, but must also transmit this password to the account holder. Common means of transmission include telephone, email, or writing the password down and physically passing it on.

All of these methods pose a significant threat to the security of the password, and leave ample margins for an attacker to gain access to system credentials. Forcing a password reset can reduce the threat that a stolen password is abused and for the abuse to go unnoticed.

If an attacker gains access to the initial password during transmission, only a very small window of opportunity exists to take advantage of this access. Initial passwords are one-time passwords. That is, they can only be used once to connect to IPA, at which point the user must change it to something that only they know, and according to the IPA password policy. If the authorized user does this first, the attacker is left with a useless password. If the attacker does this first, when the authorized user tries to authenticate, the password will fail and access will be denied. At this point the user can notify the administrator who can then investigate and take corrective action.

Again, immediately setting the status of initial passwords to "expired" reduces the opportunity for attackers to steal and take advantage of users' credentials.

This protection scheme is just one way that IPA attempts to protect user credentials. It is not an infallible protection scheme, and you should make every effort to protect user credentials during transmission. Many other attack vectors exist that this scheme does not address at all. This just provides an additional measure to make life more difficult for an attacker.

IPA and Windows Synchronization

2.1. Introduction

To synchronize user identity information between Directory Server and Windows Active Directory, IPA employs a plug-in that extends the functionality of the Directory Server Windows Sync utility. This plug-in allows IPA to perform the data manipulation necessary to achieve synchronization between Directory Server and Windows Active Directory. The IPA Windows Sync plug-in uses the `ipaWinSyncUserAttr` parameter to specify what attributes and values to add to new users that are synchronized from Active Directory.

Refer to the *Directory Server Administration Guide*[1] for more information on the Windows Sync utility.

Refer to the *Directory Server Plug-in Programmer's Guide*[2] for more information on working with plug-ins.

2.2. Directory Server and Active Directory Synchronization Features

Some differences exist in the way that IPA synchronizes data between Directory Server and Active Directory compared to an environment that does not include IPA. These are described below.

2.2.1. What does IPA Synchronize?

- IPA only synchronizes user data.

  The `Windows` subtree containing the users you want to synchronize can be specified in the synchronization agreement. By default, IPA synchronizes the `"CN=users,$SUFFIX"` subtree. You can specify a different subtree using the `--win-subtree` argument when you create the synchronization agreement. Refer to the ipa-replica-manage man page for more information.

Note

If full POSIX attributes exist in Active Directory they are not synchronized.

- IPA synchronizes new users added to Active Directory.

  Any necessary IPA attributes (POSIX, `Kerberos`) are created as part of the synchronization process. These IPA attributes are not synchronized back to Active Directory.

  Where necessary, IPA changes the DN and schema before the entry is stored in Directory Server. This involves *flattening* the DN, and discarding any OU RDNs.

[1] http://www.redhat.com/docs/manuals/dir-server/ag/8.0/Windows_Sync.html
[2] http://www.redhat.com/docs/manuals/dir-server/plugin/7.1/contents.htm

Consider the following Active Directory DN:

cn=Joe User, ou=Engineering, cn=Users, dc=example, dc=com

IPA removes the *ou=Engineering* RDN and modifies the entry to:

uid=juser,cn=users,cn=accounts,dc=example,dc=com

The user entry might appear as follows:

```
dn: uid=juser,cn=users,cn=accounts,dc=example,dc=com
objectclass: ipaUser
...
```

Example 2.1. Example of flattening an Active Directory DN before synchronizing with RHDS.

- IPA synchronizes account lock information.

  Accounts that are locked in AD are also locked in IPA, and vice versa. You can configure this feature to perform bidirectional or unidirectional synchronization, or disable it completely. The default configuration synchronizes account lock information (*ipaWinSyncAcctDisable: both*).

Note

Microsoft Windows uses the terms *Enabled* and *Disabled* when referring to the status of accounts.

2.2.2. What does IPA Not Synchronize?

- IPA does not synchronize groups.
- Users added to IPA are not synchronized to Active Directory.

  You need to manually add these users to Active Directory, after which they are synchronized between Directory Server and Active Directory. The synchronization key is the UID (username), which must be the same as the samAccountName in Active Directory.

2.2.3. Other Synchronization Features

To ensure the correct synchronization of user entries between Directory Server and Active Directory, the IPA Windows Sync plug-in can also:

- Change the Active Directory dirsync search request arguments.
- Intercept and change a user entry, including the DN, before it is sent from Directory Server to Active Directory, both in *Initialization Mode* (also known as a Total Update) and *Update Mode* (also known as an Incremental Update).

- Intercept and change entries after they have been received from Active Directory, but before they are processed by Directory Server. This includes changing the DN, both in Initialization Mode and Update Mode.
- Force synchronization between all users who exist in both IPA and Active Directory. This is specified by the *--forceSync* parameter in the synchronization agreement.

  Set this parameter to **true** (default) to forcibly synchronize all users who have accounts in both IPA and Active Directory. For this to occur, the user's UID (username) in IPA must be the same as the samAccountName in Active Directory. The **ntUser** objectclass and ntUserDomainID attribute will always be added to the IPA user entry.

  

  Note

  Even if you remove them, the IPA Winsync plug-in always adds the **ntUser** and ntUserDomainID attributes to the IPA user entry the next time a total or incremental synchronization occurs, provided the entry has been modified in Active Directory.

  If *--forceSync* is set to **false**, the Winsync plug-in works the same way as in a standard Directory Server deployment. Consequently, you need to manually add **ntUser** and ntUserDomainID to existing IPA user entries in order to synchronize them with Active Directory.

2.2.4. Changing Synchronization Subtrees

After you have created a synchronization agreement, and an IPA account is in sync with an AD account, the two accounts are permanently linked. If you make a change to the account in IPA, **winsync** looks up the entry using the GUID in the AD entry, under the domain suffix. This occurs even if you delete the original synchronization agreement and create a new one that synchronizes with a different OU in AD.

Consider the following example, where an existing synchronization agreement exists using the default subtree "CN=Users,DC=example,DC=com":

If you create a new AD user "aduser01", this account is automatically synchronized with IPA. This account has the following characteristics:

In AD: "CN=aduser 01,CN=Users,DC=example,DC=com"
In IPA: " uid=aduser01,cn=users,cn=accounts,dc=example,dc=com"

You can delete this winsync agreement, and create a new agreement between the same IPA server and AD server, but using the *win-subtree* argument to specify a different OU to synchronize (for example, OU=nyoffice,DC=example,DC=com). New users that are created under the new OU are automatically synchronized with IPA.

If you now make changes to the AD user "aduser01" on the IPA server, these changes are still synchronized with AD, even though the user is not in the OU specified by the active synchronization agreement.

Example 2.2. Example of winsync behavior when changing synchronization agreements.

Note

The reverse is not true. If you create a new user under the OU in AD specified by the original agreement ("CN=Users,$SUFFIX"), this account is not synchronized with the IPA server.

IPA Command-Line Tools and Services

This chapter provides a list of all IPA commands and services, including a brief description.

3.1. IPA Command-Line Tools

The following is a list of the available IPA command-line tools.

Note

Some of these tools require `root` privileges. Refer to the man pages for full details of each command.

ipa-adddelegation
: Adds a new delegation. A delegation is used to grant write access to certain attributes from one group to another.

ipa-addgroup
: Adds a new group.

ipa-addservice
: Adds a new service principal.

ipa-adduser
: Adds a new user.

ipa-change-master-key
: Changes the IPA master key.

Warning

The **`ipa-change-master-key`** command is only for use in specific situations. It is not for use by end-users.

ipa-client-install
: Runs the IPA client installation script. This script is Red Hat Enterprise Linux 5-specific. Installation scripts are currently only available for a limited number of operating systems.

ipa-client-setup
: Runs the IPA client installation script. This script is Red Hat Enterprise Linux 4-specific. Installation scripts are currently only available for a limited number of operating systems.

ipa-defaultoptions
: Displays or modifies the IPA search and user policies.

ipa-deldelegation
: Deletes an existing delegation.

ipa-delgroup
: Deletes an existing group.

ipa-delservice
: Deletes an existing service principal.

ipa-deluser
: Deletes an existing user. Users are automatically removed from groups when they are deleted.

ipa-findgroup
: Searches for a group that contains a specified string. The search is a substring search the name and description attributes.

ipa-findservice
: Searches for a service principal that contains a specified string. The search is a substring search in the service principal.

ipa-finduser
: Searches for a user that contains a specified string. The search is a substring search in the username, given name, family name, telephone number, organization and title attributes.

ipa-getkeytab
: Retrieves and updates `Kerberos` keytabs.

ipa-ldap-updater
: Updates the IPA LDAP configuration.

  If no file arguments are provided, **`ipa-ldap-updater`** processes all files in the **`/usr/share/ipa/updates`** directory with the **`.update`** extension.

ipa-listdelegation
: Lists all current delegations.

ipa-lockuser
: Locks or unlocks a user account.

ipa-moddelegation
: Modifies an existing delegation.

ipa-modgroup
: Modifies an existing group.

ipa-moduser
: Modifies an existing user.

ipa-passwd
: Changes a user's password.

ipa-pwpolicy
: Displays and updates the password policy.

ipa-replica-install
: Runs the IPA replica installation script.

ipa-replica-manage
: Manages (lists, adds, deletes) IPA server replicas.

ipa-replica-prepare
: Creates a replica information file for use by **`ipa-replica-install`**.

ipa-server-certinstall
: Installs a CA certificate for use by IPA.

ipa-server-install
: Runs the IPA server installation script.

ipa-upgradeconfig
: Updates the Apache configuration files of an installed server, if necessary, during an RPM update. This command should not be used by end-users.

3.2. IPA Services

The following is a list of the available IPA services.

Note

You need `root` privileges to work with these services.

ipactl
: A wrapper script to start and stop IPA-related services.

ipa_kpasswd
: Forwards password change operations to Directory Server.

ipa_webgui
: The IPA Web graphical user interface service.

XML-RPC Application Programming Interface (API) Documentation

This chapter provides a listing of the XML-RPC API functions and arguments, including any provided help.

4.1. IPA XML-RPC Application Programming Interface (API)

The XML-RPC can query the remote interface to retrieve the function and argument list, including any provided help.

add_group(group, group_container)
: Add a group in LDAP. Takes as input a dict where the key is the attribute name and the value is either a string or in the case of a multi-valued field a list of values. group_container sets where in the tree the group is placed.

add_groups_to_user(group_dns, user_dn)
: Given a list of group DNs, add them to the user.

  Returns a list of the group DNs that were not added.

add_group_to_group(group, tgroup)
: Add a group to an existing group.

  group is a DN of the group to add

  tgroup is the DN of the target group to be added to

add_members_to_group(member_dns, group_dn)
: Given a list of dn's, add them to the group cn denoted by group

  Returns a list of the member_dns that were not added to the group.

add_member_to_group(member_dn, group_dn)
: Add a member to an existing group.

add_service_principal(name, force)
: Given a name of the form: `service/FQDN`, create a service principal for it in the default realm.

  If *`force`* (Boolean) is true, create the service principal even if the host does not exist in the DNS or is not an A record.

add_user(user, user_container)
: Add a user in LDAP. Takes as input a dict where the key is the attribute name and the value is either a string or in the case of a multi-valued field a list of values. user_container sets where in the tree the user is placed.

add_users_to_group(user_uids, group_dn)
: Given a list of user uid's add them to the group cn denoted by group

  Returns a list of the users were not added to the group.

add_user_to_group(user_uid, group_dn)
Add a user to an existing group.

attrs_to_labels(attr_list)
Take a list of LDAP attributes and convert them to more friendly labels.

delete_group(group_dn)
Delete a group

group_dn is the DN of the group to delete

The memberOf plugin handles removing the group from any other groups.

delete_service_principal(principal)
Delete a service principal.

principal is the full DN of the entry to delete.

This should be called with much care.

delete_user(uid)
Delete a user. Not to be confused with `inactivate_user`. This function removes the user completely.

uid is the uid of the user to delete.

The `memberOf` plug-in handles removing the user from any other groups.

find_groups(criteria, sattrs, sizelimit=-1, timelimit=-1)
Return a list containing a User object for each existing group that matches the criteria.

find_service_principal(criteria, sattrs, sizelimit=-1, timelimit=-1)
Returns a list: counter followed by the results.

If the results are truncated, counter will be set to -1.

find_users(criteria, sattrs, sizelimit=-1, timelimit=-1)
Returns a list: counter followed by the results.

If the results are truncated, counter will be set to -1.

get_aci_entry(sattrs)
Returns the entry containing access control ACIs.

get_all_attrs()
We have a list of hard-coded attributes -> readable labels. Return that complete list if someone wants it.

get_all_users()
Return a list containing a User object for each existing user.

get_custom_fields()
Get the list of custom user fields.

A schema is a list of dict's of the form:

label: The label displayed to the user

field: the attribute name

required: true/false

It is displayed to the user in the order of the list.

get_entry_by_cn(cn, sattrs)
Get a specific entry by cn. Return as a dict of values.

Multi-valued fields are represented as lists.

get_entry_by_dn(dn, sattrs)
Get a specific entry. Return as a dict of values.

Multi-valued fields are represented as lists.

get_groups_by_member(member_dn, sattrs)
Get all of the groups an object is explicitly a member of.

This does not include groups an entry may be a member of as a result of recursion (being a group that is a member of another group). In other words, this searches on 'member' and not 'memberof'.

Return as a dict of values.

Multi-valued fields are represented as lists.

get_ipa_config()
Retrieve the IPA configuration

get_password_policy()
Retrieve the IPA password policy

get_users_by_manager(manager_dn, sattrs)
Gets the users that report to a particular manager.

get_user_by_email(email, sattrs)
Get a specific user's entry. Return as a dict of values.

Multi-valued fields are represented as lists.

get_user_by_principal(principal, sattrs)
Get a user entry searching by Kerberos Principal Name.

Return as a dict of values. Multi-valued fields are represented as lists.

get_user_by_uid(uid, sattrs)
Get a specific user's entry.

Return as a dict of values. Multi-valued fields are represented as lists.

group_members(groupdn, attr_list, membertype)
Do a memberOf search of groupdn and return the attributes in attr_list (an empty list returns all attributes).

membertype = 0 all members returned

membertype = 1 only direct members are returned

membertype = 2 only inherited members are returned

Members may be included in a group as a result of being a member of a group that is a member of the group being queried.

mark_group_active(cn)
Mark a group as active.

mark_group_inactive(cn)
Mark a group as inactive

mark_user_active(uid)
Mark a user as active

mark_user_inactive(uid)
Mark a user as inactive

modifyPassword(principal, oldpass, newpass)
Set/Reset a user's password

uid tells us who's password to change

oldpass is the old password (if available)

newpass is the new password

multiCall(calls)
Execute a multicall. Execute each method call in the calls list, collecting results and errors, and return those as a list.

ping()
Simple test to see if the XML-RPC is up and active.

remove_groups_from_user(group_dns, user_dn)
Given a list of group dn's remove them from the user.

Returns a list of the group dns that were not removed.

remove_members_from_group(member_dns, group_dn)
Given a list of member dn's remove them from the group.

Returns a list of the members not removed from the group.

remove_member_from_group(member_dn, group_dn)
Remove a member_dn from an existing group.

remove_users_from_group(user_uids, group_dn)
Given a list of user uid's remove them from the group

Returns a list of the user uids not removed from the group.

remove_user_from_group(user_uid, group_dn)
Remove a user from an existing group.

set_custom_fields(schema)
Set the list of custom user fields.

A schema is a list of dict's of the form:

label: The label displayed to the user

field: the attribute name

required: true/false

It is displayed to the user in the order of the list.

update_entry(oldentry, newentry)
Update an entry in LDAP

oldentry and newentry are XML-RPC structs.

If oldentry is not empty then it is used when determine what has changed.

If oldentry is empty then the value of newentry is compared to the current value of oldentry.

update_group(oldentry, newentry)
Wrapper around update_entry with group-specific handling.

oldentry and newentry are XML-RPC structs.

If oldentry is not empty then it is used when determine what has changed.

If oldentry is empty then the value of newentry is compared to the current value of oldentry.

If you want to change the RDN of a group you must use this function. update_entry will fail.

update_ipa_config(oldconfig, newconfig)
Update the IPA configuration.

oldconfig and newconfig are XML-RPC structs.

If oldconfig is not empty then it is used when determine what has changed.

If oldconfig is empty then the value of newconfig is compared to the current value of oldconfig.

update_password_policy(oldpolicy, newpolicy)
Update the IPA configuration

oldpolicy and newpolicy are XML-RPC structs.

If oldpolicy is not empty then it is used when determine what has changed.

If oldpolicy is empty then the value of newpolicy is compared to the current value of oldpolicy.

update_user(oldentry, newentry)
Wrapper around update_entry with user-specific handling.

oldentry and newentry are XML-RPC structs.

If oldentry is not empty then it is used when determine what has changed.

If oldentry is empty then the value of newentry is compared to the current value of oldentry.

If you want to change the RDN of a user you must use this function. update_entry will fail.

version()

The version of IPA

Glossary

A

authentication server (AS)	A server that issues tickets for a desired service which are in turn given to users for access to the service. The AS responds to requests from clients who do not have or do not send credentials with a request. It is usually used to gain access to the ticket-granting server (TGS) service by issuing a ticket-granting ticket (TGT). The AS usually runs on the same host as the key distribution center (KDC).

C

ciphertext	Encrypted data.
client	An entity on the network (a user, a host, or an application) that can receive a ticket from Kerberos.
credentials	A temporary set of electronic credentials that verify the identity of a client for a particular service. Also called a ticket.
credential cache or ticket file	A file which contains the keys for encrypting communications between a user and various network services. Kerberos 5 supports a framework for using other cache types, such as shared memory, but files are more thoroughly supported.
crypt hash	A one-way hash used to authenticate users. These are more secure than using unencrypted data, but they are still relatively easy to decrypt for an experienced cracker.

G

GSS-API	The Generic Security Service Application Program Interface (defined in RFC-2743 published by The Internet Engineering Task Force) is a set of functions which provide security services. This API is used by clients and services to authenticate to each other without either program having specific knowledge of the underlying mechanism. If a network service (such as cyrus-IMAP) uses GSS-API, it can authenticate using Kerberos.

H

hash	Also known as a hash value. A value generated by passing a string through a hash function. These values are typically used to ensure that transmitted data has not been tampered with.
hash function	A way of generating a digital "fingerprint" from input data. These functions rearrange, transpose or otherwise alter data to produce a hash value.

K

key

Data used when encrypting or decrypting other data. Encrypted data cannot be decrypted without the proper key or extremely good fortune on the part of the cracker.

key distribution center (KDC)

A service that issues Kerberos tickets, and which usually runs on the same host as the ticket-granting server (TGS).

keytab (or key table)

A file that includes an unencrypted list of principals and their keys. Servers retrieve the keys they need from keytab files instead of using kinit. The default keytab file is **`/etc/krb5.keytab`**. The KDC administration server, `/usr/kerberos/sbin/kadmind`, is the only service that uses any other file (it uses **`/var/kerberos/krb5kdc/kadm5.keytab`**).

kinit

The kinit command allows a principal who has already logged in to obtain and cache the initial ticket-granting ticket (TGT). Refer to the kinit man page for more information.

L

LDB

Local Database. LDB is a memory-mapped, LDAP-like database with persistence capabilities, developed by the Samba project. Because it is memory-mapped storage, it is fast and can act as a cache for dynamic identity data, for which IPA is the authoritative source. It can also be used to store local data. LDB also allows storing data retrieved from Directory Server in the same format as it is stored centrally.

Access to LDB is provided by an LDB library that the Info Pipe uses to access data.

P

principal (or principal name)

The principal is the unique name of a user or service allowed to authenticate using Kerberos. A principal follows the form root[/instance]@REALM. For a typical user, the root is the same as their login ID. The instance is optional. If the principal has an instance, it is separated from the root with a forward slash ("/"). An empty string ("") is considered a valid instance (which differs from the default NULL instance), but using it can be confusing. All principals in a realm have their own key, which for users is derived from a password or is randomly set for services.

R

realm

A network that uses Kerberos, composed of one or more servers called KDCs and a potentially large number of clients.

S

service

A program accessed over the network.

T

ticket	A temporary set of electronic credentials that verify the identity of a client for a particular service. Also called credentials.
ticket-granting server (TGS)	A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.
ticket-granting ticket (TGT)	A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

U

unencrypted password	A plain text, human-readable password.

Appendix A. Revision History

Revision 1.3 13 May, 2009 David O'Brien *davido@redhat.com*

Update links in Feedback page.
Revise tags used for API and Services for better presentation.

Revision 1.2 17 Apr, 2009 David O'Brien *davido@redhat.com*

Spell-check, add some new terms.

Revision 1.1 25 Nov, 2008 David O'Brien *davido@redhat.com*

BZ 470606. Document concept behind auto-expiry of user passwords.
BZ 471494. Document --win-subtree behavior.
BZ 469793. Updates from Tech Review.
BZ 471508. Add default account lock sync behavior.
Updates for AD synchronization.
Updates to IPA command list.

Revision 1.0 20 May, 2008 David O'Brien *davido@redhat.com*

Created.