



Red Hat Enterprise Linux 8

Working with DNS in Identity Management

Managing the IdM-integrated DNS service

Red Hat Enterprise Linux 8 Working with DNS in Identity Management

Managing the IdM-integrated DNS service

Legal Notice

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

DNS is an important component in a Red Hat Identity Management (IdM) domain. For example, clients use DNS to locate services and identify servers in the same site. You can manage records, zones, locations, and forwarding in the DNS server that is integrated in IdM by using the command line, the IdM Web UI, and Ansible Playbooks.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	5
CHAPTER 1. MANAGING GLOBAL DNS CONFIGURATION IN IDM USING ANSIBLE PLAYBOOKS	6
1.1. HOW IDM ENSURES THAT GLOBAL FORWARDERS FROM /ETC/RESOLV.CONF ARE NOT REMOVED BY NETWORKMANAGER	6
1.2. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE	7
1.3. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE	9
1.4. THE ACTION: MEMBER OPTION IN IPADNSCONFIG ANSIBLE-FREEIPA MODULES	11
1.5. DNS FORWARD POLICIES IN IDM	12
1.6. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT THE FORWARD FIRST POLICY IS SET IN IDM DNS GLOBAL CONFIGURATION	13
1.7. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT GLOBAL FORWARDERS ARE DISABLED IN IDM DNS	14
1.8. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT SYNCHRONIZATION OF FORWARD AND REVERSE LOOKUP ZONES IS DISABLED IN IDM DNS	16
CHAPTER 2. MANAGING DNS ZONES IN IDM	18
2.1. SUPPORTED DNS ZONE TYPES	18
2.2. ADDING A PRIMARY DNS ZONE IN IDM WEB UI	19
2.3. ADDING A PRIMARY DNS ZONE IN IDM CLI	20
2.4. REMOVING A PRIMARY DNS ZONE IN IDM WEB UI	21
2.5. REMOVING A PRIMARY DNS ZONE IN IDM CLI	21
2.6. DNS CONFIGURATION PRIORITIES	21
2.7. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES	22
2.8. EDITING THE CONFIGURATION OF A PRIMARY DNS ZONE IN IDM WEB UI	24
2.9. EDITING THE CONFIGURATION OF A PRIMARY DNS ZONE IN IDM CLI	25
2.10. ZONE TRANSFERS IN IDM	26
2.11. ENABLING ZONE TRANSFERS IN IDM WEB UI	26
2.12. ENABLING ZONE TRANSFERS IN IDM CLI	27
2.13. ADDITIONAL RESOURCES	28
CHAPTER 3. USING ANSIBLE PLAYBOOKS TO MANAGE IDM DNS ZONES	29
3.1. SUPPORTED DNS ZONE TYPES	29
3.2. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES	30
3.3. USING ANSIBLE TO CREATE A PRIMARY ZONE IN IDM DNS	32
3.4. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A PRIMARY DNS ZONE IN IDM WITH MULTIPLE VARIABLES	34
3.5. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A ZONE FOR REVERSE DNS LOOKUP WHEN AN IP ADDRESS IS GIVEN	36
CHAPTER 4. MANAGING DNS LOCATIONS IN IDM	39
4.1. DNS-BASED SERVICE DISCOVERY	39
4.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS	40
4.3. DNS TIME TO LIVE (TTL)	40
4.4. CREATING DNS LOCATIONS USING THE IDM WEB UI	40
4.5. CREATING DNS LOCATIONS USING THE IDM CLI	41
4.6. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM WEB UI	42
4.7. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM CLI	43
4.8. CONFIGURING AN IDM CLIENT TO USE IDM SERVERS IN THE SAME LOCATION	44
4.9. ADDITIONAL RESOURCES	45
CHAPTER 5. USING ANSIBLE TO MANAGE DNS LOCATIONS IN IDM	46

5.1. DNS-BASED SERVICE DISCOVERY	46
5.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS	47
5.3. DNS TIME TO LIVE (TTL)	47
5.4. USING ANSIBLE TO ENSURE AN IDM LOCATION IS PRESENT	47
5.5. USING ANSIBLE TO ENSURE AN IDM LOCATION IS ABSENT	49
5.6. ADDITIONAL RESOURCES	50
CHAPTER 6. MANAGING DNS FORWARDING IN IDM	51
6.1. THE TWO ROLES OF AN IDM DNS SERVER	51
6.2. DNS FORWARD POLICIES IN IDM	52
6.3. ADDING A GLOBAL FORWARDER IN THE IDM WEB UI	52
6.4. ADDING A GLOBAL FORWARDER IN THE CLI	55
6.5. ADDING A DNS FORWARD ZONE IN THE IDM WEB UI	56
6.6. ADDING A DNS FORWARD ZONE IN THE CLI	59
6.7. ESTABLISHING A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE	60
6.8. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE	62
6.9. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE	63
6.10. ENSURING DNS GLOBAL FORWARDERS ARE DISABLED IN IDM USING ANSIBLE	65
6.11. ENSURING THE PRESENCE OF A DNS FORWARD ZONE IN IDM USING ANSIBLE	66
6.12. ENSURING A DNS FORWARD ZONE HAS MULTIPLE FORWARDERS IN IDM USING ANSIBLE	68
6.13. ENSURING A DNS FORWARD ZONE IS DISABLED IN IDM USING ANSIBLE	70
6.14. ENSURING THE ABSENCE OF A DNS FORWARD ZONE IN IDM USING ANSIBLE	71
CHAPTER 7. MANAGING DNS RECORDS IN IDM	74
7.1. DNS RECORDS IN IDM	74
7.2. ADDING DNS RESOURCE RECORDS IN THE IDM WEB UI	75
7.3. ADDING DNS RESOURCE RECORDS FROM THE IDM CLI	76
7.4. COMMON IPA DNSRECORD-* OPTIONS	77
7.5. DELETING DNS RECORDS IN THE IDM WEB UI	80
7.6. DELETING AN ENTIRE DNS RECORD IN THE IDM WEB UI	81
7.7. DELETING DNS RECORDS IN THE IDM CLI	82
7.8. ADDITIONAL RESOURCES	82
CHAPTER 8. USING ANSIBLE TO MANAGE DNS RECORDS IN IDM	83
8.1. DNS RECORDS IN IDM	83
8.2. COMMON IPA DNSRECORD-* OPTIONS	84
8.3. ENSURING THE PRESENCE OF A AND AAAA DNS RECORDS IN IDM USING ANSIBLE	86
8.4. ENSURING THE PRESENCE OF A AND PTR DNS RECORDS IN IDM USING ANSIBLE	88
8.5. ENSURING THE PRESENCE OF MULTIPLE DNS RECORDS IN IDM USING ANSIBLE	90
8.6. ENSURING THE PRESENCE OF MULTIPLE CNAME RECORDS IN IDM USING ANSIBLE	92
8.7. ENSURING THE PRESENCE OF AN SRV RECORD IN IDM USING ANSIBLE	93
CHAPTER 9. USING CANONICALIZED DNS HOST NAMES IN IDM	96
9.1. ADDING AN ALIAS TO A HOST PRINCIPAL	96
9.2. ENABLING CANONICALIZATION OF HOST NAMES IN SERVICE PRINCIPALS ON CLIENTS	96
9.3. OPTIONS FOR USING HOST NAMES WITH DNS HOST NAME CANONICALIZATION ENABLED	97

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see [our CTO Chris Wright's message](#).

In Identity Management, planned terminology replacements include:

- ***block list*** replaces *blacklist*
- ***allow list*** replaces *whitelist*
- ***secondary*** replaces *slave*
- The word *master* is being replaced with more precise language, depending on the context:
 - ***IdM server*** replaces *IdM master*
 - ***CA renewal server*** replaces *CA renewal master*
 - ***CRL publisher server*** replaces *CRL master*
 - ***multi-supplier*** replaces *multi-master*

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Jira (account required)

1. Log in to the [Jira](#) website.
2. Click **Create** in the top navigation bar
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Create** at the bottom of the dialogue.

CHAPTER 1. MANAGING GLOBAL DNS CONFIGURATION IN IDM USING ANSIBLE PLAYBOOKS

Using the Red Hat Ansible Engine **dnsconfig** module, you can configure global configuration for Identity Management (IdM) DNS. Settings defined in global DNS configuration are applied to all IdM DNS servers. However, the global configuration has lower priority than the configuration for a specific IdM DNS zone.

The **dnsconfig** module supports the following variables:

- The global forwarders, specifically their IP addresses and the port used for communication.
- The global forwarding policy: only, first, or none. For more details on these types of DNS forward policies, see [DNS forward policies in IdM](#).
- The synchronization of forward lookup and reverse lookup zones.

Prerequisites

- DNS service is installed on the IdM server. For more information about how to install an IdM server with integrated DNS, see one of the following links:
 - [Installing an IdM server: With integrated DNS, with an integrated CA as the root CA](#)
 - [Installing an IdM server: With integrated DNS, with an external CA as the root CA](#)
 - [Installing an IdM server: With integrated DNS, without a CA](#)

This chapter includes the following sections:

- [How IdM ensures that global forwarders from /etc/resolv.conf are not removed by NetworkManager](#)
- [Ensuring the presence of a DNS global forwarder in IdM using Ansible](#)
- [Ensuring the absence of a DNS global forwarder in IdM using Ansible](#)
- [The **action: member** option in ipadnsconfig ansible-freeipa modules](#)
- [An introduction to \[DNS forward policies in IdM\]\(#\)](#)
- [Using an Ansible playbook to ensure that the forward first policy is set in IdM DNS global configuration](#)
- [Using an Ansible playbook to ensure that global forwarders are disabled in IdM DNS](#)
- [Using an Ansible playbook to ensure that synchronization of forward and reverse lookup zones is disabled in IdM DNS](#)

1.1. HOW IDM ENSURES THAT GLOBAL FORWARDERS FROM /ETC/RESOLV.CONF ARE NOT REMOVED BY NETWORKMANAGER

Installing Identity Management (IdM) with integrated DNS configures the `/etc/resolv.conf` file to point to the **127.0.0.1** localhost address:

```
# Generated by NetworkManager
search idm.example.com
nameserver 127.0.0.1
```

In certain environments, such as networks that use **Dynamic Host Configuration Protocol** (DHCP), the **NetworkManager** service may revert changes to the `/etc/resolv.conf` file. To make the DNS configuration persistent, the IdM DNS installation process also configures the **NetworkManager** service in the following way:

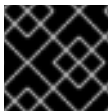
1. The DNS installation script creates an `/etc/NetworkManager/conf.d/zzz-ipa.conf` **NetworkManager** configuration file to control the search order and DNS server list:

```
# auto-generated by IPA installer
[main]
dns=default

[global-dns]
searches=$DOMAIN

[global-dns-domain-*]
servers=127.0.0.1
```

2. The **NetworkManager** service is reloaded, which always creates the `/etc/resolv.conf` file with the settings from the last file in the `/etc/NetworkManager/conf.d/` directory. This is in this case the **zzz-ipa.conf** file.



IMPORTANT

Do not modify the `/etc/resolv.conf` file manually.

1.2. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the presence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of **7.7.9.9** and IP v6 address of **2001:db8::1:0** on port **53**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipadmin_password`.

- You know the IdM administrator password.

Procedure

1. Navigate to the **/usr/share/doc/ansible-freeipa/playbooks/dnsconfig** directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-of-a-global-forwarder.yml
```

4. Open the **ensure-presence-of-a-global-forwarder.yml** file for editing.

5. Adapt the file by setting the following variables:

- a. Change the **name** variable for the playbook to **Playbook to ensure the presence of a global forwarder in IdM DNS**.
- b. In the **tasks** section, change the **name** of the task to **Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53**.
- c. In the **forwarders** section of the **ipadnsconfig** portion:
 - i. Change the first **ip_address** value to the IPv4 address of the global forwarder: **7.7.9.9**.
 - ii. Change the second **ip_address** value to the IPv6 address of the global forwarder: **2001:db8::1:0**.
 - iii. Verify the **port** value is set to **53**.
- d. Change the **state** to **present**.
This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the presence of a global forwarder in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port
    53
    ipadnsconfig:
      forwarders:
      - ip_address: 7.7.9.9
```

```
- ip_address: 2001:db8::1:0
  port: 53
  state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-presence-
of-a-global-forwarder.yml
```

Additional resources

- See the **README-dnsconfig.md** file in the `/usr/share/doc/ansible-freeipa/` directory.

1.3. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the absence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS global forwarder with an Internet Protocol (IP) v4 address of **8.8.6.6** and IP v6 address of **2001:4860:4860::8800** on port **53**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

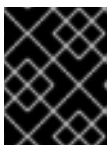
```
$ cp forwarders-absent.yml ensure-absence-of-a-global-forwarder.yml
```

4. Open the **ensure-absence-of-a-global-forwarder.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the **name** variable for the playbook to **Playbook to ensure the absence of a global forwarder in IdM DNS**.
 - b. In the **tasks** section, change the **name** of the task to **Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53**.
 - c. In the **forwarders** section of the **ipadnsconfig** portion:
 - i. Change the first **ip_address** value to the IPv4 address of the global forwarder: **8.8.6.6**.
 - ii. Change the second **ip_address** value to the IPv6 address of the global forwarder: **2001:4860:4860::8800**.
 - iii. Verify the **port** value is set to **53**.
 - d. Set the **action** variable to **member**.
 - e. Verify the **state** is set to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the absence of a global forwarder in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure the absence of a DNS global forwarder to 8.8.6.6 and
    2001:4860:4860::8800 on port 53
    ipadnsconfig:
      forwarders:
      - ip_address: 8.8.6.6
      - ip_address: 2001:4860:4860::8800
      port: 53
      action: member
      state: absent
```



IMPORTANT

If you only use the **state: absent** option in your playbook without also using **action: member**, the playbook fails.

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-absence-of-a-global-forwarder.yml
```

Additional resources

- The **README-dnsconfig.md** file in the `/usr/share/doc/ansible-freeipa/` directory
- The **action: member** option in `ipadnsconfig` `ansible-freeipa` modules

1.4. THE ACTION: MEMBER OPTION IN IPADNSCONFIG ANSIBLE-FREEIPA MODULES

Excluding global forwarders in Identity Management (IdM) by using the **ansible-freeipa** `ipadnsconfig` module requires using the **action: member** option in addition to the **state: absent** option. If you only use **state: absent** in your playbook without also using **action: member**, the playbook fails. Consequently, to remove all global forwarders, you must specify all of them individually in the playbook. In contrast, the **state: present** option does not require **action: member**.

The following table provides configuration examples for both adding and removing DNS global forwarders that demonstrate the correct use of the `action: member` option. The table shows, in each line:

- The global forwarders configured before executing a playbook
- An excerpt from the playbook
- The global forwarders configured after executing the playbook

Table 1.1. `ipadnsconfig` management of global forwarders

Forwarders before	Playbook excerpt	Forwarders after
8.8.6.6	<pre>[...] tasks: - name: Ensure the presence of DNS global forwarder 8.8.6.7 ipadnsconfig: forwarders: - ip_address: 8.8.6.7 state: present</pre>	8.8.6.7
8.8.6.6	<pre>[...] tasks: - name: Ensure the presence of DNS global forwarder 8.8.6.7 ipadnsconfig: forwarders: - ip_address: 8.8.6.7 action: member state: present</pre>	8.8.6.6, 8.8.6.7

Forwarders before	Playbook excerpt	Forwarders after
8.8.6.6, 8.8.6.7	<pre>[...] tasks: - name: Ensure the absence of DNS global forwarder 8.8.6.7 ipadnsconfig: forwarders: - ip_address: 8.8.6.7 state: absent</pre>	Trying to execute the playbook results in an error. The original configuration - 8.8.6.6, 8.8.6.7 - is left unchanged.
8.8.6.6, 8.8.6.7	<pre>[...] tasks: - name: Ensure the absence of DNS global forwarder 8.8.6.7 ipadnsconfig: forwarders: - ip_address: 8.8.6.7 action: member state: absent</pre>	8.8.6.6

1.5. DNS FORWARD POLICIES IN IDM

IdM supports the **first** and **only** standard BIND forward policies, as well as the **none** IdM-specific forward policy.

Forward first(*default*)

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The **forward first** policy is the default policy, and it is suitable for optimizing DNS traffic.

Forward only

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND returns an error to the client. The **forward only** policy is recommended for environments with split DNS configuration.

None (*forwarding disabled*)

DNS queries are not forwarded with the **none** forwarding policy. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.



NOTE

You cannot use forwarding to combine data in IdM with data from other DNS servers. You can only forward queries for specific subzones of the primary zone in IdM DNS.

By default, the BIND service does not forward queries to another server if the queried DNS name belongs to a zone for which the IdM server is authoritative. In such a situation, if the queried DNS name cannot be found in the IdM database, the **NXDOMAIN** answer is returned. Forwarding is not used.

Example 1.1. Example Scenario

The IdM server is authoritative for the **test.example.** DNS zone. BIND is configured to forward queries to the DNS server with the **192.0.2.254** IP address.

When a client sends a query for the **nonexistent.test.example.** DNS name, BIND detects that the IdM server is authoritative for the **test.example.** zone and does not forward the query to the **192.0.2.254.** server. As a result, the DNS client receives the **NXDomain** error message, informing the user that the queried domain does not exist.

1.6. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT THE FORWARD FIRST POLICY IS SET IN IDM DNS GLOBAL CONFIGURATION

Follow this procedure to use an Ansible playbook to ensure that global forwarding policy in IdM DNS is set to **forward first**.

If you use the **forward first** DNS forwarding policy, DNS queries are forwarded to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The forward first policy is the default policy. It is suitable for traffic optimization.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipaadmin_password**.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **set-configuration.yml** Ansible playbook file. For example:

```
$ cp set-configuration.yml set-forward-policy-to-first.yml
```

4. Open the **set-forward-policy-to-first.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnsconfig** task section:
 - Set the **ipaadmin_password** variable to your IdM administrator password.
 - Set the **forward_policy** variable to **first**.
Delete all the other lines of the original playbook that are irrelevant. This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to set global forwarding policy to first
  hosts: ipaserver
  become: true

  tasks:
  - name: Set global forwarding policy to first.
    ipadnsconfig:
      ipaadmin_password: "{{ ipaadmin_password }}"
      forward_policy: first
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file set-forward-policy-to-first.yml
```

Additional resources

- See [DNS forward policies in IdM](#).
- See the **README-dnsconfig.md** file in the **/usr/share/doc/ansible-freeipa/** directory.
- For more sample playbooks, see the **/usr/share/doc/ansible-freeipa/playbooks/dnsconfig** directory.

1.7. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT GLOBAL FORWARDERS ARE DISABLED IN IDM DNS

Follow this procedure to use an Ansible playbook to ensure that global forwarders are disabled in IdM DNS. The disabling is done by setting the **forward_policy** variable to **none**.

Disabling global forwarders causes DNS queries not to be forwarded. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `disable-global-forwarders.yml` Ansible playbook file. For example:

```
$ cp disable-global-forwarders.yml disable-global-forwarders-copy.yml
```

4. Open the `disable-global-forwarders-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipadnsconfig` task section:

- Set the `ipaadmin_password` variable to your IdM administrator password.
- Set the `forward_policy` variable to `none`.
This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to disable global DNS forwarders
  hosts: ipaserver
  become: true

  tasks:
  - name: Disable global forwarders.
```

```
ipadnsconfig:
  ipadmin_password: "{{ ipadmin_password }}"
  forward_policy: none
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file disable-
global-forwarders-copy.yml
```

Additional resources

- See [DNS forward policies in IdM](#).
- See the **README-dnsconfig.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See more sample playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory.

1.8. USING AN ANSIBLE PLAYBOOK TO ENSURE THAT SYNCHRONIZATION OF FORWARD AND REVERSE LOOKUP ZONES IS DISABLED IN IDM DNS

Follow this procedure to use an Ansible playbook to ensure that forward and reverse lookup zones are not synchronized in IdM DNS.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.
- Your IdM environment contains an integrated DNS server.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **disallow-reverse-sync.yml** Ansible playbook file. For example:

```
$ cp disallow-reverse-sync.yml disallow-reverse-sync-copy.yml
```

4. Open the **disallow-reverse-sync-copy.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnsconfig** task section:

- Set the **ipaadmin_password** variable to your IdM administrator password.
- Set the **allow_sync_ptr** variable to **no**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to disallow reverse record synchronization
  hosts: ipaserver
  become: true

  tasks:
  - name: Disallow reverse record synchronization.
    ipadnsconfig:
      ipaadmin_password: "{{ ipaadmin_password }}"
      allow_sync_ptr: no
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file disallow-
reverse-sync-copy.yml
```

Additional resources

- See the **README-dnsconfig.md** file in the **/usr/share/doc/ansible-freeipa/** directory.
- For more sample playbooks, see the **/usr/share/doc/ansible-freeipa/playbooks/dnsconfig** directory.

CHAPTER 2. MANAGING DNS ZONES IN IDM

As Identity Management (IdM) administrator, you can manage how IdM DNS zones work. The chapter describes the following topics and procedures:

- [What DNS zone types are supported in IdM](#)
 - [How to add primary IdM DNS zones using the IdM Web UI](#)
 - [How to add primary IdM DNS zones using the IdM CLI](#)
 - [How to remove primary IdM DNS zones using the IdM Web UI](#)
 - [How to remove primary IdM DNS zones using the IdM CLI](#)
- [What DNS attributes you can configure in IdM](#)
 - [How you can configure these attributes in the IdM Web UI](#)
 - [How you can configure these attributes in the IdM CLI](#)
- [How zone transfers work in IdM](#)
 - [How you can allow zone transfers in the IdM Web UI](#)
 - [How you can allow zone transfers in the IdM CLI](#)

Prerequisites

- DNS service is installed on the IdM server. For more information about how to install an IdM server with integrated DNS, see one of the following links:
 - [Installing an IdM server: With integrated DNS, with an integrated CA as the root CA](#)
 - [Installing an IdM server: With integrated DNS, with an external CA as the root CA](#)
 - [Installing an IdM server: With integrated DNS, without a CA](#)

2.1. SUPPORTED DNS ZONE TYPES

Identity Management (IdM) supports two types of DNS zones: *primary* and *forward* zones. These two types of zones are described here, including an example scenario for DNS forwarding.



NOTE

This guide uses the BIND terminology for zone types which is different from the terminology used for Microsoft Windows DNS. Primary zones in BIND serve the same purpose as *forward lookup zones* and *reverse lookup zones* in Microsoft Windows DNS. Forward zones in BIND serve the same purpose as *conditional forwarders* in Microsoft Windows DNS.

Primary DNS zones

Primary DNS zones contain authoritative DNS data and can accept dynamic DNS updates. This behavior is equivalent to the **type master** setting in standard BIND configuration. You can manage primary zones using the **ipa dnszone-*** commands.

In compliance with standard DNS rules, every primary zone must contain **start of authority** (SOA) and **nameserver** (NS) records. IdM generates these records automatically when the DNS zone is created, but you must copy the NS records manually to the parent zone to create proper delegation.

In accordance with standard BIND behavior, queries for names for which the server is not authoritative are forwarded to other DNS servers. These DNS servers, so called forwarders, may or may not be authoritative for the query.

Example 2.1. Example scenario for DNS forwarding

The IdM server contains the **test.example.** primary zone. This zone contains an NS delegation record for the **sub.test.example.** name. In addition, the **test.example.** zone is configured with the **192.0.2.254** forwarder IP address for the **sub.test.example** subzone.

A client querying the name **nonexistent.test.example.** receives the **NXDomain** answer, and no forwarding occurs because the IdM server is authoritative for this name.

On the other hand, querying for the **host1.sub.test.example.** name is forwarded to the configured forwarder **192.0.2.254** because the IdM server is not authoritative for this name.

Forward DNS zones

From the perspective of IdM, forward DNS zones do not contain any authoritative data. In fact, a forward "zone" usually only contains two pieces of information:

- A domain name
- The IP address of a DNS server associated with the domain

All queries for names belonging to the domain defined are forwarded to the specified IP address. This behavior is equivalent to the **type forward** setting in standard BIND configuration. You can manage forward zones using the **ipa dnsforwardzone-*** commands.

Forward DNS zones are especially useful in the context of IdM-Active Directory (AD) trusts. If the IdM DNS server is authoritative for the **idm.example.com** zone and the AD DNS server is authoritative for the **ad.example.com** zone, then **ad.example.com** is a DNS forward zone for the **idm.example.com** primary zone. That means that when a query comes from an IdM client for the IP address of **somehost.ad.example.com**, the query is forwarded to an AD domain controller specified in the **ad.example.com** IdM DNS forward zone.

2.2. ADDING A PRIMARY DNS ZONE IN IDM WEB UI

Follow this procedure to add a primary DNS zone using the Identity Management (IdM) Web UI.

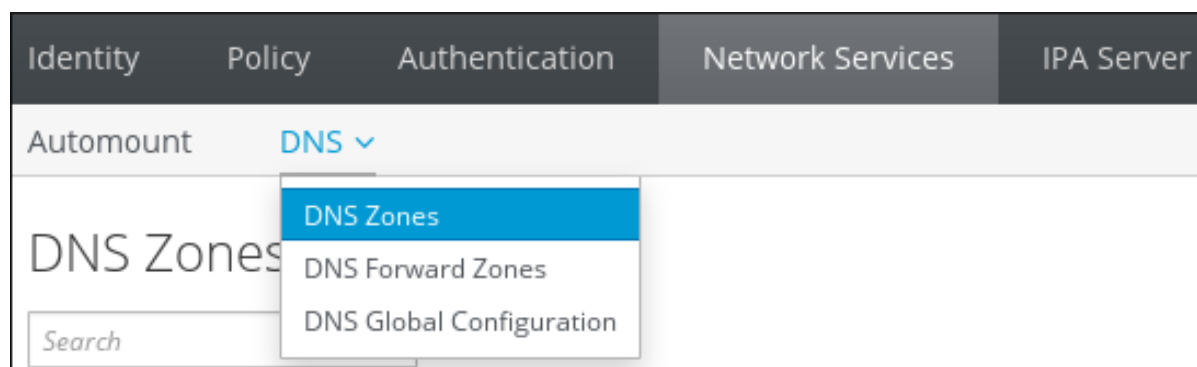
Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services → DNS → DNS Zones**.

Figure 2.1. Managing IdM DNS primary zones



2. Click **Add** at the top of the list of all zones.
3. Provide the zone name.

Figure 2.2. Entering an new IdM primary zone

 The screenshot shows a dialog box titled 'Add DNS Zone' with a close button (X) in the top right corner. Inside the dialog, there are two radio button options: 'Zone name' (selected) and 'Reverse zone'. The 'Zone name' option has a text input field containing 'zone.example.com.'. The 'Reverse zone' option has a text input field that is currently empty. Below these options, there is a label '* Required field'. At the bottom of the dialog, there are four buttons: 'Add', 'Add and Add Another', 'Add and Edit', and 'Cancel'.

4. Click **Add**.

2.3. ADDING A PRIMARY DNS ZONE IN IDM CLI

Follow this procedure to add a primary DNS zone using the Identity Management (IdM) command-line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.

Procedure

- The **ipa dnszone-add** command adds a new zone to the DNS domain. Adding a new zone requires you to specify the name of the new subdomain. You can pass the subdomain name directly with the command:

```
$ ipa dnszone-add newzone.idm.example.com
```

If you do not pass the name to **ipa dnszone-add**, the script prompts for it automatically.

Additional resources

- See `ipa dnszone-add --help`.

2.4. REMOVING A PRIMARY DNS ZONE IN IDM WEB UI

Follow this procedure to remove a primary DNS zone from Identity Management (IdM) using the IdM Web UI.

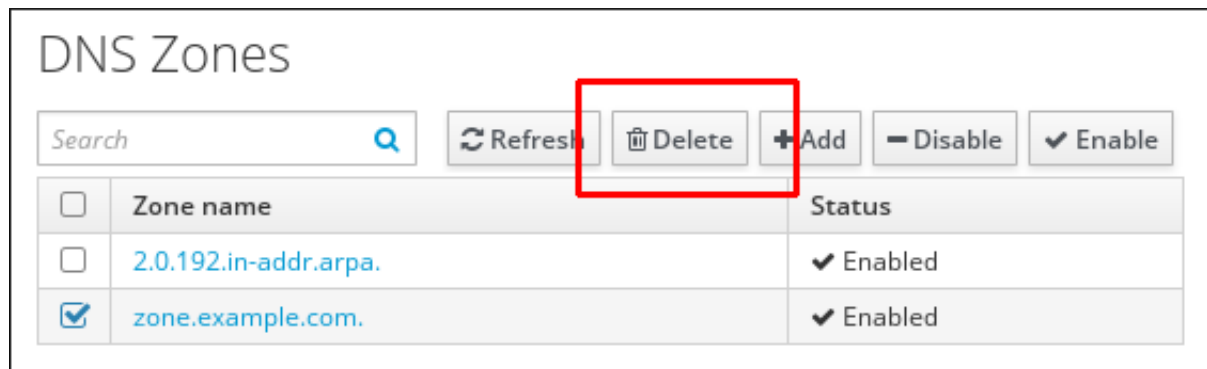
Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.
2. Select the check box by the zone name and click **Delete**.

Figure 2.3. Removing a primary DNS Zone



3. In the **Remove DNS zones** dialog window, confirm that you want to delete the selected zone.

2.5. REMOVING A PRIMARY DNS ZONE IN IDM CLI

Follow this procedure to remove a primary DNS zone from Identity Management (IdM) using the IdM command-line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.

Procedure

- To remove a primary DNS zone, enter the `ipa dnszone-del` command, followed by the name of the zone you want to remove. For example:

```
$ ipa dnszone-del idm.example.com
```

2.6. DNS CONFIGURATION PRIORITIES

You can configure many DNS configuration options on the following levels. Each level has a different priority.

Zone-specific configuration

The level of configuration specific for a particular zone defined in IdM has the highest priority. You can manage zone-specific configuration by using the **ipa dnszone-*** and **ipa dnsforwardzone-*** commands.

Per-server configuration

You are asked to define per-server forwarders during the installation of an IdM server. You can manage per-server forwarders by using the **ipa dnsserver-*** commands. If you do not want to set a per-server forwarder when installing a replica, you can use the **--no-forwarder** option.

Global DNS configuration

If no zone-specific configuration is defined, IdM uses global DNS configuration stored in LDAP. You can manage global DNS configuration using the **ipa dnsconfig-*** commands. Settings defined in global DNS configuration are applied to all IdM DNS servers.

Configuration in `/etc/named.conf`

Configuration defined in the `/etc/named.conf` file on each IdM DNS server has the lowest priority. It is specific for each server and must be edited manually.

The `/etc/named.conf` file is usually only used to specify DNS forwarding to a local DNS cache. Other options are managed using the commands for zone-specific and global DNS configuration mentioned above.

You can configure DNS options on multiple levels at the same time. In such cases, configuration with the highest priority takes precedence over configuration defined at lower levels.

Additional resources

- The **Priority order of configuration** section in [Per Server Config in LDAP](#)

2.7. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES

Identity Management (IdM) creates a new zone with certain default configuration, such as the refresh periods, transfer settings, or cache settings. In [IdM DNS zone attributes](#), you can find the attributes of the default zone configuration that you can modify using one of the following options:

- The **dnszone-mod** command in the command-line interface (CLI). For more information, see [Editing the configuration of a primary DNS zone in IdM CLI](#).
- The IdM Web UI. For more information, see [Editing the configuration of a primary DNS zone in IdM Web UI](#).
- An Ansible playbook that uses the **ipadnszone** module. For more information, see [Managing DNS zones in IdM](#).

Along with setting the actual information for the zone, the settings define how the DNS server handles the *start of authority* (SOA) record entries and how it updates its records from the DNS name server.

Table 2.1. IdM DNS zone attributes

Attribute	Command-Line Option	Description
-----------	---------------------	-------------

Attribute	Command-Line Option	Description
Authoritative name server	--name-server	<p>Sets the domain name of the primary DNS name server, also known as SOA MNAME.</p> <p>By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using --name-server is ignored.</p>
Administrator e-mail address	--admin-email	Sets the email address to use for the zone administrator. This defaults to the root account on the host.
SOA serial	--serial	Sets a serial number in the SOA record. Note that IdM sets the version number automatically and users are not expected to modify it.
SOA refresh	--refresh	Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.
SOA retry	--retry	Sets the time, in seconds, to wait before retrying a failed refresh operation.
SOA expire	--expire	Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.
SOA minimum	--minimum	Sets the time to live (TTL) value in seconds for negative caching according to RFC 2308 .
SOA time to live	--ttl	Sets TTL in seconds for records at zone apex. In zone example.com , for instance, all records (A, NS, or SOA) under name example.com are configured, but no other domain names, like test.example.com , are affected.
Default time to live	--default-ttl	Sets the default time to live (TTL) value in seconds for negative caching for all values in a zone that never had an individual TTL value set before. Requires a restart of the named-pkcs11 service on all IdM DNS servers after changes to take effect.
BIND update policy	--update-policy	Sets the permissions allowed to clients in the DNS zone.
Dynamic update	--dynamic-update=TRUE FALSE	<p>Enables dynamic updates to DNS records for clients.</p> <p>Note that if this is set to false, IdM client machines will not be able to add or update their IP address.</p>

Attribute	Command-Line Option	Description
Allow transfer	--allow-transfer = <i>string</i>	Gives a list of IP addresses or network names which are allowed to transfer the given zone, separated by semicolons (;). Zone transfers are disabled by default. The default --allow-transfer value is none .
Allow query	--allow-query	Gives a list of IP addresses or network names which are allowed to issue DNS queries, separated by semicolons (;).
Allow PTR sync	--allow-sync-ptr =1 0	Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.
Zone forwarders	--forwarder = <i>IP_address</i>	Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the IdM domain. To specify multiple forwarders, use the option multiple times.
Forward policy	--forward-policy =none only first	Specifies the forward policy. For information about the supported policies, see DNS forward policies in IdM .

2.8. EDITING THE CONFIGURATION OF A PRIMARY DNS ZONE IN IDM WEB UI

Follow this procedure to edit the configuration attributes of a primary Identity Management (IdM) DNS using the IdM Web UI.

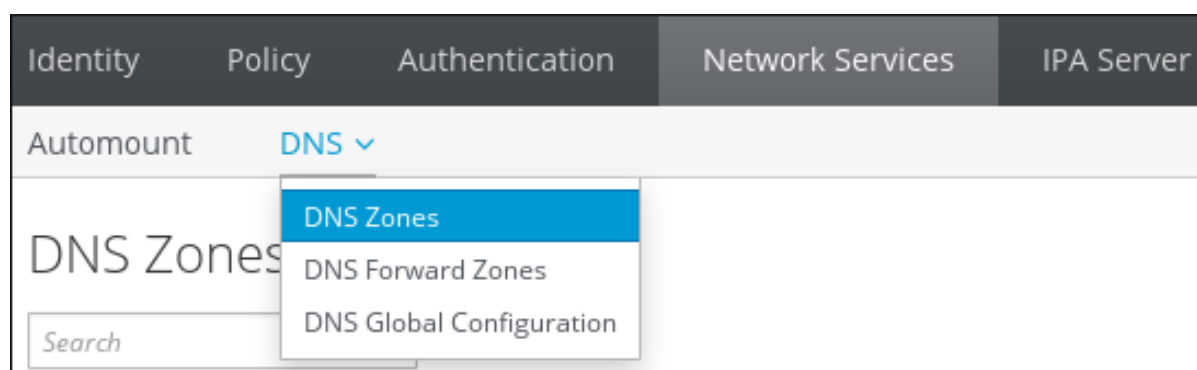
Prerequisites

- You are logged in as IdM administrator.

Procedure

- In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.

Figure 2.4. DNS primary zones management



2. In the **DNS Zones** section, click on the zone name in the list of all zones to open the DNS zone page.

Figure 2.5. Editing a primary zone

DNS Zones		
<input type="text" value="Search"/> <input type="button" value="Refresh"/> <input type="button" value="Delete"/> <input type="button" value="+ Add"/> <input type="button" value="- Disable"/> <input type="button" value="✓ Enable"/>		
<input type="checkbox"/>	Zone name	Status
<input type="checkbox"/>	2.0.192.in-addr.arpa.	✓ Enabled
<input type="checkbox"/>	zone.example.com.	✓ Enabled

3. Click **Settings**.

Figure 2.6. The Settings tab in the primary zone edit page

✓ DNS Zone: zone.example.com.

DNS Zone Settings

Zone name zone.example.com.

4. Change the zone configuration as required.
For information about the available settings, see [IdM DNS zone attributes](#).
5. Click **Save** to confirm the new configuration.

**NOTE**

If you are changing the default time to live (TTL) of a zone, restart the **named-pkcs11** service on all IdM DNS servers to make the changes take effect. All other settings are automatically activated immediately.

2.9. EDITING THE CONFIGURATION OF A PRIMARY DNS ZONE IN IDM CLI

Follow this procedure to edit the configuration of a primary DNS zone using the Identity Management (IdM) command-line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.

Procedure

- To modify an existing primary DNS zone, use the **ipa dnszone-mod** command. For example, to set the time to wait before retrying a failed refresh operation to 1800 seconds:

```
$ ipa dnszone-mod --retry 1800
```

For more information about the available settings and their corresponding CLI options, see [IdM DNS zone attributes](#).

If a specific setting does not have a value in the DNS zone entry you are modifying, the **ipa dnszone-mod** command adds the value. If the setting does not have a value, the command overwrites the current value with the specified value.



NOTE

If you are changing the default time to live (TTL) of a zone, restart the **named-pkcs11** service on all IdM DNS servers to make the changes take effect. All other settings are automatically activated immediately.

Additional resources

- See **ipa dnszone-mod --help**.

2.10. ZONE TRANSFERS IN IDM

In an Identity Management (IdM) deployment that has integrated DNS, you can use *zone transfers* to copy all resource records from one name server to another. Name servers maintain authoritative data for their zones. If you make changes to the zone on a DNS server that is authoritative for *zone A* DNS zone, you must distribute the changes among the other name servers in the IdM DNS domain that are outside *zone A*.



IMPORTANT

The IdM-integrated DNS can be written to by different servers simultaneously. The Start of Authority (SOA) serial numbers in IdM zones are not synchronized among the individual IdM DNS servers. For this reason, configure your DNS servers outside the to-be-transferred zone to only use one specific DNS server inside the to-be-transferred zone. This prevents zone transfer failures caused by non-synchronized SOA serial numbers.

IdM supports zone transfers according to the [RFC 5936](#) (AXFR) and [RFC 1995](#) (IXFR) standards.

Additional resources

- See [Enabling zone transfers in IdM Web UI](#).
- See [Enabling zone transfers in IdM CLI](#).

2.11. ENABLING ZONE TRANSFERS IN IDM WEB UI

Follow this procedure to enable zone transfers in Identity Management (IdM) using the IdM Web UI.

Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.
2. Click **Settings**.
3. Under **Allow transfer**, specify the name servers to which you want to transfer the zone records.

Figure 2.7. Enabling zone transfers



The screenshot shows a web form titled 'Allow transfer'. It contains three input fields, each with an IP address and an 'Undo' button to its right. The IP addresses are 192.0.2.1, 198.51.100.1, and 203.0.113.1. Below these fields are two buttons: 'Add' and 'Undo All'.

4. Click **Save** at the top of the DNS zone page to confirm the new configuration.

2.12. ENABLING ZONE TRANSFERS IN IDM CLI

Follow this procedure to enable zone transfers in Identity Management (IdM) using the IdM command-line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.
- You have root access to the secondary DNS servers.

Procedure

- To enable zone transfers in the **BIND** service, enter the **ipa dnszone-mod** command, and specify the list of name servers that are outside the to-be-transferred zone to which the zone records will be transferred using the **--allow-transfer** option. For example:

```
$ ipa dnszone-mod --allow-transfer=192.0.2.1;198.51.100.1;203.0.113.1
idm.example.com
```

Verification steps

1. SSH to one of the DNS servers to which zone transfer has been enabled:

```
$ ssh 192.0.2.1
```

2. Transfer the IdM DNS zone using a tool such as the **dig** utility:

```
# dig @ipa-server zone_name AXFR
```

If the command returns no error, you have successfully enabled zone transfer for *zone_name*.

2.13. ADDITIONAL RESOURCES

- See [Using Ansible playbooks to manage IdM DNS zones](#) .

CHAPTER 3. USING ANSIBLE PLAYBOOKS TO MANAGE IDM DNS ZONES

As Identity Management (IdM) administrator, you can manage how IdM DNS zones work using the **dnszone** module available in the **ansible-freeipa** package.

- [What DNS zone types are supported in IdM](#)
- [What DNS attributes you can configure in IdM](#)
- [How to use an Ansible playbook to create a primary zone in IdM DNS](#)
- [How to use an Ansible playbook to ensure the presence of a primary IdM DNS zone with multiple variables](#)
- [How to use an Ansible playbook to ensure the presence of a zone for reverse DNS lookup when an IP address is given](#)

Prerequisites

- DNS service is installed on the IdM server. For more information about how to use Red Hat Ansible Engine to install an IdM server with integrated DNS, see [Installing an Identity Management server using an Ansible playbook](#).

3.1. SUPPORTED DNS ZONE TYPES

Identity Management (IdM) supports two types of DNS zones: *primary* and *forward* zones. These two types of zones are described here, including an example scenario for DNS forwarding.



NOTE

This guide uses the BIND terminology for zone types which is different from the terminology used for Microsoft Windows DNS. Primary zones in BIND serve the same purpose as *forward lookup zones* and *reverse lookup zones* in Microsoft Windows DNS. Forward zones in BIND serve the same purpose as *conditional forwarders* in Microsoft Windows DNS.

Primary DNS zones

Primary DNS zones contain authoritative DNS data and can accept dynamic DNS updates. This behavior is equivalent to the **type master** setting in standard BIND configuration. You can manage primary zones using the **ipa dnszone-*** commands.

In compliance with standard DNS rules, every primary zone must contain **start of authority** (SOA) and **nameserver** (NS) records. IdM generates these records automatically when the DNS zone is created, but you must copy the NS records manually to the parent zone to create proper delegation.

In accordance with standard BIND behavior, queries for names for which the server is not authoritative are forwarded to other DNS servers. These DNS servers, so called forwarders, may or may not be authoritative for the query.

Example 3.1. Example scenario for DNS forwarding

The IdM server contains the **test.example.** primary zone. This zone contains an NS delegation record for the **sub.test.example.** name. In addition, the **test.example.** zone is configured with the **192.0.2.254** forwarder IP address for the **sub.text.example** subzone.

A client querying the name **nonexistent.test.example.** receives the **NXDomain** answer, and no forwarding occurs because the IdM server is authoritative for this name.

On the other hand, querying for the **host1.sub.test.example.** name is forwarded to the configured forwarder **192.0.2.254** because the IdM server is not authoritative for this name.

Forward DNS zones

From the perspective of IdM, forward DNS zones do not contain any authoritative data. In fact, a forward "zone" usually only contains two pieces of information:

- A domain name
- The IP address of a DNS server associated with the domain

All queries for names belonging to the domain defined are forwarded to the specified IP address. This behavior is equivalent to the **type forward** setting in standard BIND configuration. You can manage forward zones using the **ipa dnsforwardzone-*** commands.

Forward DNS zones are especially useful in the context of IdM-Active Directory (AD) trusts. If the IdM DNS server is authoritative for the **idm.example.com** zone and the AD DNS server is authoritative for the **ad.example.com** zone, then **ad.example.com** is a DNS forward zone for the **idm.example.com** primary zone. That means that when a query comes from an IdM client for the IP address of **somehost.ad.example.com**, the query is forwarded to an AD domain controller specified in the **ad.example.com** IdM DNS forward zone.

3.2. CONFIGURATION ATTRIBUTES OF PRIMARY IDM DNS ZONES

Identity Management (IdM) creates a new zone with certain default configuration, such as the refresh periods, transfer settings, or cache settings. In [IdM DNS zone attributes](#), you can find the attributes of the default zone configuration that you can modify using one of the following options:

- The **dnszone-mod** command in the command-line interface (CLI). For more information, see [Editing the configuration of a primary DNS zone in IdM CLI](#).
- The IdM Web UI. For more information, see [Editing the configuration of a primary DNS zone in IdM Web UI](#).
- An Ansible playbook that uses the **ipadnszone** module. For more information, see [Managing DNS zones in IdM](#).

Along with setting the actual information for the zone, the settings define how the DNS server handles the *start of authority* (SOA) record entries and how it updates its records from the DNS name server.

Table 3.1. IdM DNS zone attributes

Attribute	ansible-freeipa variable	Description
-----------	--------------------------	-------------

Attribute	ansible-freeipa variable	Description
Authoritative name server	name_server	<p>Sets the domain name of the primary DNS name server, also known as SOA MNAME.</p> <p>By default, each IdM server advertises itself in the SOA MNAME field. Consequently, the value stored in LDAP using --name-server is ignored.</p>
Administrator e-mail address	admin_email	Sets the email address to use for the zone administrator. This defaults to the root account on the host.
SOA serial	serial	Sets a serial number in the SOA record. Note that IdM sets the version number automatically and users are not expected to modify it.
SOA refresh	refresh	Sets the interval, in seconds, for a secondary DNS server to wait before requesting updates from the primary DNS server.
SOA retry	retry	Sets the time, in seconds, to wait before retrying a failed refresh operation.
SOA expire	expire	Sets the time, in seconds, that a secondary DNS server will try to perform a refresh update before ending the operation attempt.
SOA minimum	minimum	Sets the time to live (TTL) value in seconds for negative caching according to RFC 2308 .
SOA time to live	ttl	Sets TTL in seconds for records at zone apex. In zone example.com , for instance, all records (A, NS, or SOA) under name example.com are configured, but no other domain names, like test.example.com , are affected.
Default time to live	default_ttl	Sets the default time to live (TTL) value in seconds for negative caching for all values in a zone that never had an individual TTL value set before. Requires a restart of the named-pkcs11 service on all IdM DNS servers after changes to take effect.
BIND update policy	update_policy	Sets the permissions allowed to clients in the DNS zone.
Dynamic update	dynamic_update=TRUE FALSE	<p>Enables dynamic updates to DNS records for clients.</p> <p>Note that if this is set to false, IdM client machines will not be able to add or update their IP address.</p>

Attribute	ansible-freeipa variable	Description
Allow transfer	allow_transfer =string	Gives a list of IP addresses or network names which are allowed to transfer the given zone, separated by semicolons (;). Zone transfers are disabled by default. The default allow_transfer value is none .
Allow query	allow_query	Gives a list of IP addresses or network names which are allowed to issue DNS queries, separated by semicolons (;).
Allow PTR sync	allow_sync_ptr =1 0	Sets whether A or AAAA records (forward records) for the zone will be automatically synchronized with the PTR (reverse) records.
Zone forwarders	forwarder =IP_address	Specifies a forwarder specifically configured for the DNS zone. This is separate from any global forwarders used in the IdM domain. To specify multiple forwarders, use the option multiple times.
Forward policy	forward_policy =none only first	Specifies the forward policy. For information about the supported policies, see DNS forward policies in IdM .

Additional resources

- See the **README-dnszone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.

3.3. USING ANSIBLE TO CREATE A PRIMARY ZONE IN IDM DNS

Follow this procedure to use an Ansible playbook to ensure that a primary DNS zone exists. In the example used in the procedure below, you ensure the presence of the **zone.idm.example.com** DNS zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `dnszone-present.yml` Ansible playbook file. For example:

```
$ cp dnszone-present.yml dnszone-present-copy.yml
```

4. Open the `dnszone-present-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipadnszone` task section:

- Set the `ipaadmin_password` variable to your IdM administrator password.
- Set the `zone_name` variable to `zone.idm.example.com`.
This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure zone is present.
      ipadnszone:
        ipaadmin_password: "{{ ipaadmin_password }}"
        zone_name: zone.idm.example.com
        state: present
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file dnszone-present-copy.yml
```

Additional resources

- See [Supported DNS zone types](#).
- See the `README-dnszone.md` file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory.

3.4. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A PRIMARY DNS ZONE IN IDM WITH MULTIPLE VARIABLES

Follow this procedure to use an Ansible playbook to ensure that a primary DNS zone exists. In the example used in the procedure below, an IdM administrator ensures the presence of the **zone.idm.example.com** DNS zone. The Ansible playbook configures multiple parameters of the zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipaadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **dnszone-all-params.yml** Ansible playbook file. For example:

```
$ cp dnszone-all-params.yml dnszone-all-params-copy.yml
```

4. Open the **dnszone-all-params-copy.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnszone** task section:
 - Set the **ipaadmin_password** variable to your IdM administrator password.
 - Set the **zone_name** variable to **zone.idm.example.com**.
 - Set the **allow_sync_ptr** variable to true if you want to allow the synchronization of forward and reverse records, that is the synchronization of A and AAAA records with PTR records.
 - Set the **dynamic_update** variable to true to enable IdM client machines to add or update their IP addresses.
 - Set the **dnssec** variable to true to allow inline DNSSEC signing of records in the zone.

- Set the **allow_transfer** variable to the IP addresses of secondary name servers in the zone.
- Set the **allow_query** variable to the IP addresses or networks that are allowed to issue queries.
- Set the **forwarders** variable to the IP addresses of global forwarders.
- Set the **serial** variable to the SOA record serial number.
- Define the **refresh**, **retry**, **expire**, **minimum**, **ttl**, and **default_ttl** values for DNS records in the zone.
- Define the NSEC3PARAM record for the zone using the **nsec3param_rec** variable.
- Set the **skip_overlap_check** variable to true to force DNS creation even if it overlaps with an existing zone.
- Set the **skip_nameserver_check** to true to force DNS zone creation even if the nameserver is not resolvable.

This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure zone is present.
      ipadszone:
        ipaadmin_password: "{{ ipaadmin_password }}"
        zone_name: zone.idm.example.com
        allow_sync_ptr: true
        dynamic_update: true
        dnssec: true
        allow_transfer:
          - 1.1.1.1
          - 2.2.2.2
        allow_query:
          - 1.1.1.1
          - 2.2.2.2
        forwarders:
          - ip_address: 8.8.8.8
          - ip_address: 8.8.4.4
            port: 52
        serial: 1234
        refresh: 3600
        retry: 900
        expire: 1209600
        minimum: 3600
        ttl: 60
        default_ttl: 90
        name_server: server.idm.example.com.
        admin_email: admin.admin@idm.example.com
        nsec3param_rec: "1 7 100 0123456789abcdef"
```

```
skip_overlap_check: true
skip_nameserver_check: true
state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file dnszone-all-params-copy.yml
```

Additional resources

- See [Supported DNS zone types](#).
- See [Configuration attributes of primary IdM DNS zones](#).
- See the **README-dnszone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory.

3.5. USING AN ANSIBLE PLAYBOOK TO ENSURE THE PRESENCE OF A ZONE FOR REVERSE DNS LOOKUP WHEN AN IP ADDRESS IS GIVEN

Follow this procedure to use an Ansible playbook to ensure that a reverse DNS zone exists. In the example used in the procedure below, an IdM administrator ensures the presence of a reverse DNS lookup zone using the IP address and prefix length of an IdM host.

Providing the prefix length of the IP address of your DNS server using the **name_from_ip** variable allows you to control the zone name. If you do not state the prefix length, the system queries DNS servers for zones and, based on the **name_from_ip** value of `192.168.1.2`, the query can return any of the following DNS zones:

- `1.168.192.in-addr.arpa.`
- `168.192.in-addr.arpa.`
- `192.in-addr.arpa.`

Because the zone returned by the query might not be what you expect, **name_from_ip** can only be used with the **state** option set to **present** to prevent accidental removals of zones.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.

- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnszone
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `dnszone-reverse-from-ip.yml` Ansible playbook file. For example:

```
$ cp dnszone-reverse-from-ip.yml dnszone-reverse-from-ip-copy.yml
```

4. Open the `dnszone-reverse-from-ip-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the **ipadnszone** task section:

- Set the **ipaadmin_password** variable to your IdM administrator password.
- Set the **name_from_ip** variable to the IP of your IdM nameserver, and provide its prefix length.

This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure dnszone present
  hosts: ipaserver
  become: true

  tasks:
    - name: Ensure zone for reverse DNS lookup is present.
      ipadnszone:
        ipaadmin_password: "{{ ipaadmin_password }}"
        name_from_ip: 192.168.1.2/24
        state: present
        register: result
    - name: Display inferred zone name.
      debug:
        msg: "Zone name: {{ result.dnszone.name }}"
```

The playbook creates a zone for reverse DNS lookup from the **192.168.1.2** IP address and its prefix length of 24. Next, the playbook displays the resulting zone name.

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file dnszone-
reverse-from-ip-copy.yml
```

Additional resources

- See [Supported DNS zone types](#).
- See the **README-dnszone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnszone` directory.

CHAPTER 4. MANAGING DNS LOCATIONS IN IDM

To learn more about managing Identity Management (IdM) DNS locations by using the IdM Web UI and IdM command-line interface (CLI), see the following topics and procedures:

- [DNS-based service discovery](#)
- [Deployment considerations for DNS locations](#)
- [DNS time to live \(TTL\)](#)
- [Creating DNS locations using the IdM Web UI](#)
- [Creating DNS locations using the IdM CLI](#)
- [Assigning an IdM server to a DNS location using the IdM Web UI](#)
- [Assigning an IdM server to a DNS location using the IdM Web UI](#)
- [Configuring an IdM client to use IdM servers in the same location](#)

4.1. DNS-BASED SERVICE DISCOVERY

DNS-based service discovery is a process in which a client uses the DNS protocol to locate servers in a network that offer a specific service, such as **LDAP** or **Kerberos**. One typical type of operation is to allow clients to locate authentication servers within the closest network infrastructure, because they provide a higher throughput and lower network latency, lowering overall costs.

The major advantages of service discovery are:

- No need for clients to be explicitly configured with names of nearby servers.
- DNS servers are used as central providers of policy. Clients using the same DNS server have access to the same policy about service providers and their preferred order.

In an Identity Management (IdM) domain, DNS service records (SRV records) exist for **LDAP**, **Kerberos**, and other services. For example, the following command queries the DNS server for hosts providing a TCP-based **Kerberos** service in an IdM DNS domain:

Example 4.1. DNS location independent results

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
0 100 88 idmserver-01.idm.example.com.
0 100 88 idmserver-02.idm.example.com.
```

The output contains the following information:

- **0** (priority): Priority of the target host. A lower value is preferred.
- **100** (weight). Specifies a relative weight for entries with the same priority. For further information, see [RFC 2782, section 3](#).
- **88** (port number): Port number of the service.
- Canonical name of the host providing the service.

In the example, the two host names returned have the same priority and weight. In this case, the client uses a random entry from the result list.

When the client is, instead, configured to query a DNS server that is configured in a DNS location, the output differs. For IdM servers that are assigned to a location, tailored values are returned. In the example below, the client is configured to query a DNS server in the location **germany**:

Example 4.2. DNS location-based results

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
_kerberos._tcp.germany._locations.idm.example.com.
0 100 88 idmserver-01.idm.example.com.
50 100 88 idmserver-02.idm.example.com.
```

The IdM DNS server automatically returns a DNS alias (CNAME) pointing to a DNS location specific SRV record which prefers local servers. This CNAME record is shown in the first line of the output. In the example, the host **idmserver-01.idm.example.com** has the lowest priority value and is therefore preferred. The **idmserver-02.idm.example.com** has a higher priority and thus is used only as backup for cases when the preferred host is unavailable.

4.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS

Identity Management (IdM) can generate location-specific service (SRV) records when using the integrated DNS. Because each IdM DNS server generates location-specific SRV records, you have to install at least one IdM DNS server in each DNS location.

The client's affinity to a DNS location is only defined by the DNS records received by the client. For this reason, you can combine IdM DNS servers with non-IdM DNS consumer servers and recursors if the clients doing DNS service discovery resolve location-specific records from IdM DNS servers.

In the majority of deployments with mixed IdM and non-IdM DNS services, DNS recursors select the closest IdM DNS server automatically by using round-trip time metrics. Typically, this ensures that clients using non-IdM DNS servers are getting records for the nearest DNS location and thus use the optimal set of IdM servers.

4.3. DNS TIME TO LIVE (TTL)

Clients can cache DNS resource records for an amount of time that is set in the zone's configuration. Because of this caching, a client might not be able to receive the changes until the time to live (TTL) value expires. The default TTL value in Identity Management (IdM) is **1 day**.

If your client computers roam between sites, you should adapt the TTL value for your IdM DNS zone. Set the value to a lower value than the time clients need to roam between sites. This ensures that cached DNS entries on the client expire before they reconnect to another site and thus query the DNS server to refresh location-specific SRV records.

Additional resources

- See [Configuration attributes of primary IdM DNS zones](#) .

4.4. CREATING DNS LOCATIONS USING THE IDM WEB UI

You can use DNS locations to increase the speed of communication between Identity Management (IdM) clients and servers. Follow this procedure to create a DNS location using the IdM Web UI.

Prerequisites

- Your IdM deployment has integrated DNS.
- You have a permission to create DNS locations in IdM. For example, you are logged in as IdM admin.

Procedure

1. Open the **IPA Server** tab.
2. Select **Topology** subtab.
3. Click **IPA Locations** in the navigation bar.
4. Click **Add** at the top of the locations list.
5. Fill in the location name.
6. Click the **Add** button to save the location.
7. Optional: Repeat the steps to add further locations.

Additional resources

- See [Assigning an IdM server to a DNS location using the IdM Web UI](#) .
- See [Using Ansible to ensure an IdM location is present](#) .

4.5. CREATING DNS LOCATIONS USING THE IDM CLI

You can use DNS locations to increase the speed of communication between Identity Management (IdM) clients and servers. Follow this procedure to create DNS locations using the **ipa location-add** command in the IdM command-line interface (CLI).

Prerequisites

- Your IdM deployment has integrated DNS.
- You have a permission to create DNS locations in IdM. For example, you are logged in as IdM admin.

Procedure

1. For example, to create a new location **germany**, enter:

```
$ ipa location-add germany
-----
Added IPA location "germany"
-----
Location name: germany
```

2. Optional: Repeat the step to add further locations.

Additional resources

- See [Assigning an IdM Server to a DNS Location using the IdM CLI](#) .
- See [Using Ansible to ensure an IdM location is present](#) .

4.6. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM WEB UI

You can use Identity Management (IdM) DNS locations to increase the speed of communication between IdM clients and servers. Follow this procedure to assign IdM servers to DNS locations using the IdM Web UI.

Prerequisites

- Your IdM deployment has integrated DNS.
- You are logged in as a user with a permission to assign a server to a DNS location, for example the IdM admin user.
- You have **root** access to the host that you want to assign a DNS location to.
- You have [created the IdM DNS locations](#) to which you want to assign servers.

Procedure

1. Open the **IPA Server** tab.
2. Select the **Topology** subtab.
3. Click **IPA Servers** in the navigation.
4. Click on the IdM server name.
5. Select a DNS location, and optionally set a service weight:

Figure 4.1. Assigning a server to a DNS location

IPA Server: idmserver-01.idm.example.com

Refresh Revert Save

Server name	idmserver-01.idm.example.com.
Min domain level	0
Max domain level	1
Managed suffixes	domain ca
Location	germany
Service weight	100

- Click **Save**.
- In the command-line interface (CLI) of the host you assigned in the previous steps the DNS location to, restart the **named-pkcs11** service:

```
[root@idmserver-01 ~]# systemctl restart named-pkcs11
```

- Optional: Repeat the steps to assign DNS locations to further IdM servers.

Additional resources

- See [Configuring an IdM client to use IdM servers in the same location](#) .

4.7. ASSIGNING AN IDM SERVER TO A DNS LOCATION USING THE IDM CLI

You can use Identity Management (IdM) DNS locations to increase the speed of communication between IdM clients and servers. Follow this procedure to assign IdM servers to DNS locations using the IdM command-line interface (CLI).

Prerequisites

- Your IdM deployment has integrated DNS.
- You are logged in as a user with a permission to assign a server to a DNS location, for example the IdM admin user.
- You have **root** access to the host that you want to assign a DNS location to.
- You have [created the IdM DNS locations](#) to which you want to assign servers.

Procedure

- Optional: List all configured DNS locations:

```
[root@server ~]# ipa location-find
-----
2 IPA locations matched
-----
Location name: australia
Location name: germany
-----
Number of entries returned: 2
-----
```

- Assign the server to the DNS location. For example, to assign the location **germany** to the server **idmserver-01.idm.example.com**, run:

```
# ipa server-mod idmserver-01.idm.example.com --location=germany
ipa: WARNING: Service named-pkcs11.service requires restart on IPA server
idmserver-01.idm.example.com to apply configuration changes.
-----
Modified IPA server "idmserver-01.idm.example.com"
-----
Servername: idmserver-01.idm.example.com
Min domain level: 0
Max domain level: 1
Location: germany
Enabled server roles: DNS server, NTP server
```

- Restart the **named-pkcs11** service on the host you assigned in the previous steps the DNS location to:

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

- Optional: Repeat the steps to assign DNS locations to further IdM servers.

Additional resources

- See [Configuring an IdM client to use IdM servers in the same location](#) .

4.8. CONFIGURING AN IDM CLIENT TO USE IDM SERVERS IN THE SAME LOCATION

Identity Management (IdM) servers are assigned to DNS locations as described in [Assigning an IdM server to a DNS location using the IdM Web UI](#). Now you can configure the clients to use a DNS server that is in the same location as the IdM servers:

- If a **DHCP** server assigns the DNS server IP addresses to the clients, configure the **DHCP** service. For further details about assigning a DNS server in your **DHCP** service, see the **DHCP** service documentation.
- If your clients do not receive the DNS server IP addresses from a **DHCP** server, manually set the IPs in the client's network configuration. For further details about configuring the network on Red Hat Enterprise Linux, see the [Configuring Network Connection Settings](#) section in the *Red Hat Enterprise Linux Networking Guide*.

**NOTE**

If you configure the client to use a DNS server that is assigned to a different location, the client contacts IdM servers in both locations.

Example 4.3. Different name server entries depending on the location of the client

The following example shows different name server entries in the `/etc/resolv.conf` file for clients in different locations:

Clients in Prague:

```
nameserver 10.10.0.1
nameserver 10.10.0.2
```

Clients in Paris:

```
nameserver 10.50.0.1
nameserver 10.50.0.3
```

Clients in Oslo:

```
nameserver 10.30.0.1
```

Clients in Berlin:

```
nameserver 10.30.0.1
```

If each of the DNS servers is assigned to a location in IdM, the clients use the IdM servers in their location.

4.9. ADDITIONAL RESOURCES

- See [Using Ansible to manage DNS locations in IdM](#) .

CHAPTER 5. USING ANSIBLE TO MANAGE DNS LOCATIONS IN IDM

As Identity Management (IdM) administrator, you can manage IdM DNS locations using the **location** module available in the **ansible-freeipa** package.

- [DNS-based service discovery](#)
- [Deployment considerations for DNS locations](#)
- [DNS time to live \(TTL\)](#)
- [Using Ansible to ensure an IdM location is present](#)
- [Using Ansible to ensure an IdM location is absent](#)

5.1. DNS-BASED SERVICE DISCOVERY

DNS-based service discovery is a process in which a client uses the DNS protocol to locate servers in a network that offer a specific service, such as **LDAP** or **Kerberos**. One typical type of operation is to allow clients to locate authentication servers within the closest network infrastructure, because they provide a higher throughput and lower network latency, lowering overall costs.

The major advantages of service discovery are:

- No need for clients to be explicitly configured with names of nearby servers.
- DNS servers are used as central providers of policy. Clients using the same DNS server have access to the same policy about service providers and their preferred order.

In an Identity Management (IdM) domain, DNS service records (SRV records) exist for **LDAP**, **Kerberos**, and other services. For example, the following command queries the DNS server for hosts providing a TCP-based **Kerberos** service in an IdM DNS domain:

Example 5.1. DNS location independent results

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
0 100 88 idmsvr-01.idm.example.com.
0 100 88 idmsvr-02.idm.example.com.
```

The output contains the following information:

- **0** (priority): Priority of the target host. A lower value is preferred.
- **100** (weight). Specifies a relative weight for entries with the same priority. For further information, see [RFC 2782, section 3](#).
- **88** (port number): Port number of the service.
- Canonical name of the host providing the service.

In the example, the two host names returned have the same priority and weight. In this case, the client uses a random entry from the result list.

When the client is, instead, configured to query a DNS server that is configured in a DNS location, the output differs. For IdM servers that are assigned to a location, tailored values are returned. In the example below, the client is configured to query a DNS server in the location **germany**:

Example 5.2. DNS location-based results

```
$ dig -t SRV +short _kerberos._tcp.idm.example.com
_kerberos._tcp.germany._locations.idm.example.com.
0 100 88 idmsvr-01.idm.example.com.
50 100 88 idmsvr-02.idm.example.com.
```

The IdM DNS server automatically returns a DNS alias (CNAME) pointing to a DNS location specific SRV record which prefers local servers. This CNAME record is shown in the first line of the output. In the example, the host **idmsvr-01.idm.example.com** has the lowest priority value and is therefore preferred. The **idmsvr-02.idm.example.com** has a higher priority and thus is used only as backup for cases when the preferred host is unavailable.

5.2. DEPLOYMENT CONSIDERATIONS FOR DNS LOCATIONS

Identity Management (IdM) can generate location-specific service (SRV) records when using the integrated DNS. Because each IdM DNS server generates location-specific SRV records, you have to install at least one IdM DNS server in each DNS location.

The client's affinity to a DNS location is only defined by the DNS records received by the client. For this reason, you can combine IdM DNS servers with non-IdM DNS consumer servers and recursors if the clients doing DNS service discovery resolve location-specific records from IdM DNS servers.

In the majority of deployments with mixed IdM and non-IdM DNS services, DNS recursors select the closest IdM DNS server automatically by using round-trip time metrics. Typically, this ensures that clients using non-IdM DNS servers are getting records for the nearest DNS location and thus use the optimal set of IdM servers.

5.3. DNS TIME TO LIVE (TTL)

Clients can cache DNS resource records for an amount of time that is set in the zone's configuration. Because of this caching, a client might not be able to receive the changes until the time to live (TTL) value expires. The default TTL value in Identity Management (IdM) is **1 day**.

If your client computers roam between sites, you should adapt the TTL value for your IdM DNS zone. Set the value to a lower value than the time clients need to roam between sites. This ensures that cached DNS entries on the client expire before they reconnect to another site and thus query the DNS server to refresh location-specific SRV records.

Additional resources

- See [Configuration attributes of primary IdM DNS zones](#).

5.4. USING ANSIBLE TO ENSURE AN IDM LOCATION IS PRESENT

As a system administrator of Identity Management (IdM), you can configure IdM DNS locations to allow clients to locate authentication servers within the closest network infrastructure.

The following procedure describes how to use an Ansible playbook to ensure a DNS location is present in IdM. The example describes how to ensure that the **germany** DNS location is present in IdM. As a result, you can assign particular IdM servers to this location so that local IdM clients can use them to reduce server response time.

Prerequisites

- You know the IdM administrator password.
- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You understand the [deployment considerations for DNS locations](#).

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```
$ cd ~/MyPlaybooks/
```

2. Make a copy of the `location-present.yml` file located in the `/usr/share/doc/ansible-freeipa/playbooks/location/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/location/location-present.yml location-present-copy.yml
```

3. Open the `location-present-copy.yml` Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the `ipalocation` task section:
 - Adapt the **name** of the task to correspond to your use case.
 - Set the `ipaadmin_password` variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the location.

This is the modified Ansible playbook file for the current example:

```
---
- name: location present example
  hosts: ipaserver

  vars_files:
    - /home/user_name/MyPlaybooks/secret.yml
  tasks:
    - name: Ensure that the "germany" location is present
```

```

ipalocation:
  ipadmin_password: "{{ ipadmin_password }}"
  name: germany

```

5. Save the file.
6. Run the Ansible playbook. Specify the playbook file, the file storing the password protecting the **secret.yml** file, and the inventory file:

```

$ ansible-playbook --vault-password-file=password_file -v -i inventory location-
present-copy.yml

```

Additional resources

- See [Assigning an IdM server to a DNS location using the IdM Web UI](#) or [Assigning an IdM server to a DNS location using the IdM CLI](#).

5.5. USING ANSIBLE TO ENSURE AN IDM LOCATION IS ABSENT

As a system administrator of Identity Management (IdM), you can configure IdM DNS locations to allow clients to locate authentication servers within the closest network infrastructure.

The following procedure describes how to use an Ansible playbook to ensure that a DNS location is absent in IdM. The example describes how to ensure that the **germany** DNS location is absent in IdM. As a result, you cannot assign particular IdM servers to this location and local IdM clients cannot use them.

Prerequisites

- You know the IdM administrator password.
- No IdM server is assigned to the **germany** DNS location.
- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- The example assumes that you have [created and configured](#) the `~/MyPlaybooks/` directory as a central location to store copies of sample playbooks.

Procedure

1. Navigate to the `~/MyPlaybooks/` directory:

```

$ cd ~/MyPlaybooks/

```

2. Make a copy of the **location-absent.yml** file located in the `/usr/share/doc/ansible-freeipa/playbooks/location/` directory:

```
$ cp /usr/share/doc/ansible-freeipa/playbooks/location/location-absent.yml location-absent-copy.yml
```

3. Open the **location-absent-copy.yml** Ansible playbook file for editing.
4. Adapt the file by setting the following variables in the **ipalocation** task section:
 - Adapt the **name** of the task to correspond to your use case.
 - Set the **ipaadmin_password** variable to the password of the IdM administrator.
 - Set the **name** variable to the name of the DNS location.
 - Make sure that the **state** variable is set to **absent**.

This is the modified Ansible playbook file for the current example:

```
---
- name: location absent example
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure that the "germany" location is absent
    ipalocation:
      ipaadmin_password: "{{ ipaadmin_password }}"
      name: germany
      state: absent
```

5. Save the file.
6. Run the Ansible playbook. Specify the playbook file, the file storing the password protecting the **secret.yml** file, and the inventory file:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory location-absent-copy.yml
```

5.6. ADDITIONAL RESOURCES

- See the **README-location.md** file in the **/usr/share/doc/ansible-freeipa/** directory.
- See sample Ansible playbooks in the **/usr/share/doc/ansible-freeipa/playbooks/location** directory.

CHAPTER 6. MANAGING DNS FORWARDING IN IDM

Follow these procedures to configure DNS global forwarders and DNS forward zones in the Identity Management (IdM) Web UI, the IdM CLI, and using Ansible:

- [The two roles of an IdM DNS server](#)
- [DNS forward policies in IdM](#)
- [Adding a global forwarder in the IdM Web UI](#)
- [Adding a global forwarder in the CLI](#)
- [Adding a DNS Forward Zone in the IdM Web UI](#)
- [Adding a DNS Forward Zone in the CLI](#)
- [Establishing a DNS Global Forwarder in IdM using Ansible](#)
- [Ensuring the presence of a DNS global forwarder in IdM using Ansible](#)
- [Ensuring the absence of a DNS global forwarder in IdM using Ansible](#)
- [Ensuring DNS Global Forwarders are disabled in IdM using Ansible](#)
- [Ensuring the presence of a DNS Forward Zone in IdM using Ansible](#)
- [Ensuring a DNS Forward Zone has multiple forwarders in IdM using Ansible](#)
- [Ensuring a DNS Forward Zone is disabled in IdM using Ansible](#)
- [Ensuring the absence of a DNS Forward Zone in IdM using Ansible](#)

6.1. THE TWO ROLES OF AN IDM DNS SERVER

DNS forwarding affects how a DNS service answers DNS queries. By default, the Berkeley Internet Name Domain (BIND) service integrated with IdM acts as both an *authoritative* and a *recursive* DNS server:

Authoritative DNS server

When a DNS client queries a name belonging to a DNS zone for which the IdM server is authoritative, BIND replies with data contained in the configured zone. Authoritative data always takes precedence over any other data.

Recursive DNS server

When a DNS client queries a name for which the IdM server is not authoritative, BIND attempts to resolve the query using other DNS servers. If forwarders are not defined, BIND asks the root servers on the Internet and uses a recursive resolution algorithm to answer the DNS query.

In some cases, it is not desirable to let BIND contact other DNS servers directly and perform the recursion based on data available on the Internet. You can configure BIND to use another DNS server, a *forwarder*, to resolve the query.

When you configure BIND to use a forwarder, queries and answers are forwarded back and forth between the IdM server and the forwarder, and the IdM server acts as the DNS cache for non-authoritative data.

6.2. DNS FORWARD POLICIES IN IDM

IdM supports the **first** and **only** standard BIND forward policies, as well as the **none** IdM-specific forward policy.

Forward first (*default*)

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND falls back to the recursive resolution using servers on the Internet. The **forward first** policy is the default policy, and it is suitable for optimizing DNS traffic.

Forward only

The IdM BIND service forwards DNS queries to the configured forwarder. If a query fails because of a server error or timeout, BIND returns an error to the client. The **forward only** policy is recommended for environments with split DNS configuration.

None (*forwarding disabled*)

DNS queries are not forwarded with the **none** forwarding policy. Disabling forwarding is only useful as a zone-specific override for global forwarding configuration. This option is the IdM equivalent of specifying an empty list of forwarders in BIND configuration.



NOTE

You cannot use forwarding to combine data in IdM with data from other DNS servers. You can only forward queries for specific subzones of the primary zone in IdM DNS.

By default, the BIND service does not forward queries to another server if the queried DNS name belongs to a zone for which the IdM server is authoritative. In such a situation, if the queried DNS name cannot be found in the IdM database, the **NXDOMAIN** answer is returned. Forwarding is not used.

Example 6.1. Example Scenario

The IdM server is authoritative for the **test.example.** DNS zone. BIND is configured to forward queries to the DNS server with the **192.0.2.254** IP address.

When a client sends a query for the **nonexistent.test.example.** DNS name, BIND detects that the IdM server is authoritative for the **test.example.** zone and does not forward the query to the **192.0.2.254.** server. As a result, the DNS client receives the **NXDomain** error message, informing the user that the queried domain does not exist.

6.3. ADDING A GLOBAL FORWARDER IN THE IDM WEB UI

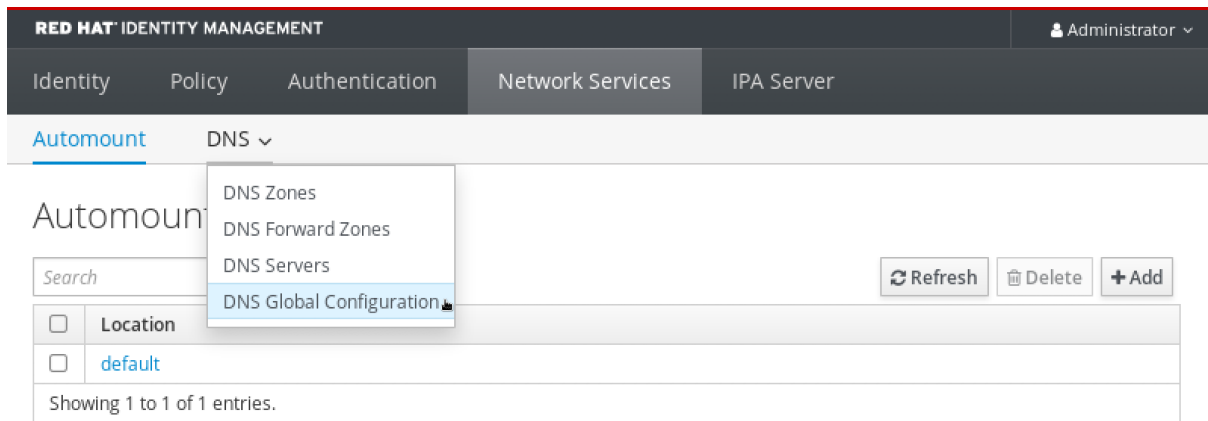
Follow this procedure to add a global DNS forwarder in the Identity Management (IdM) Web UI.

Prerequisites

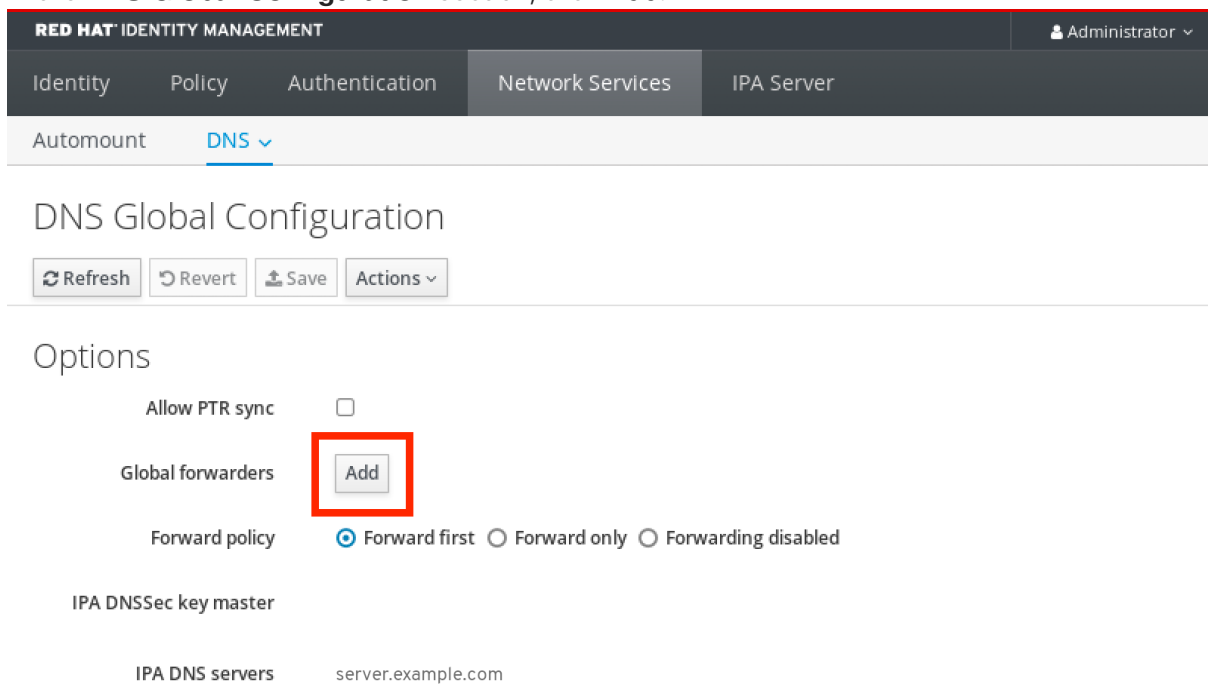
- You are logged in to the IdM WebUI as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

Procedure

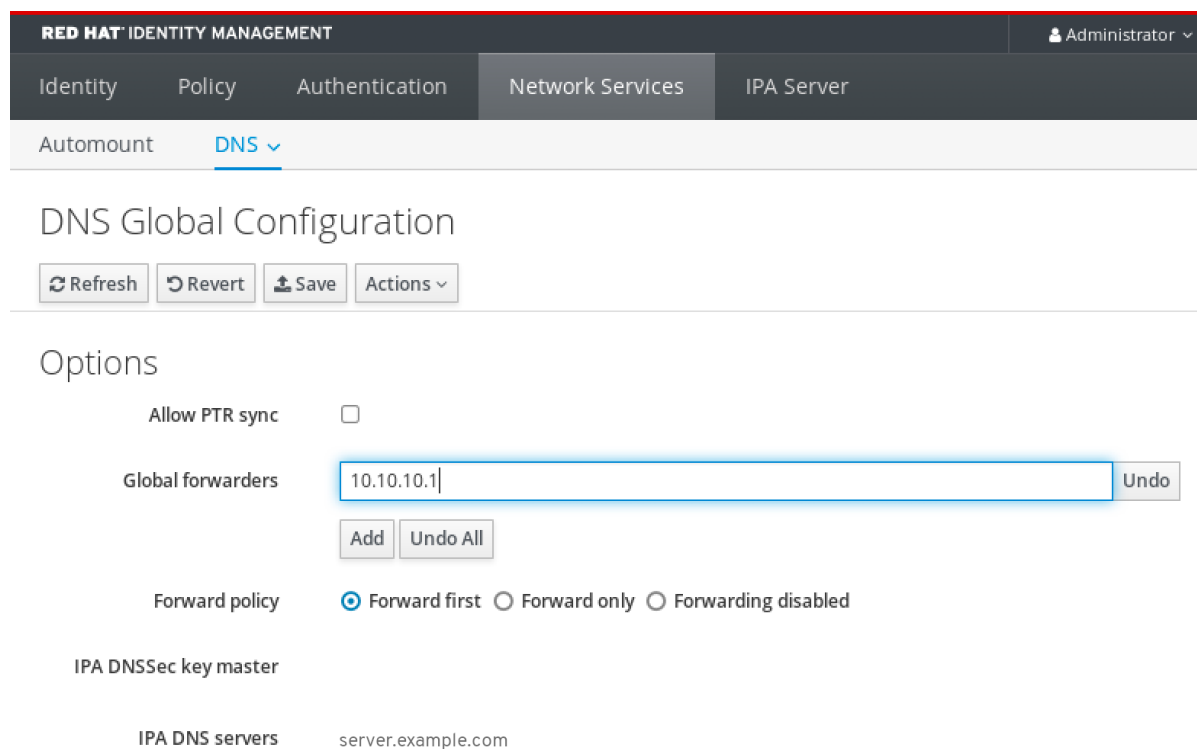
1. In the IdM Web UI, select **Network Services** → **DNS Global Configuration** → **DNS**.



2. In the **DNS Global Configuration** section, click **Add**.



3. Specify the IP address of the DNS server that will receive forwarded DNS queries.



RED HAT IDENTITY MANAGEMENT Administrator ▾

Identity Policy Authentication **Network Services** IPA Server

Automount **DNS ▾**

DNS Global Configuration

Options

Allow PTR sync ☐

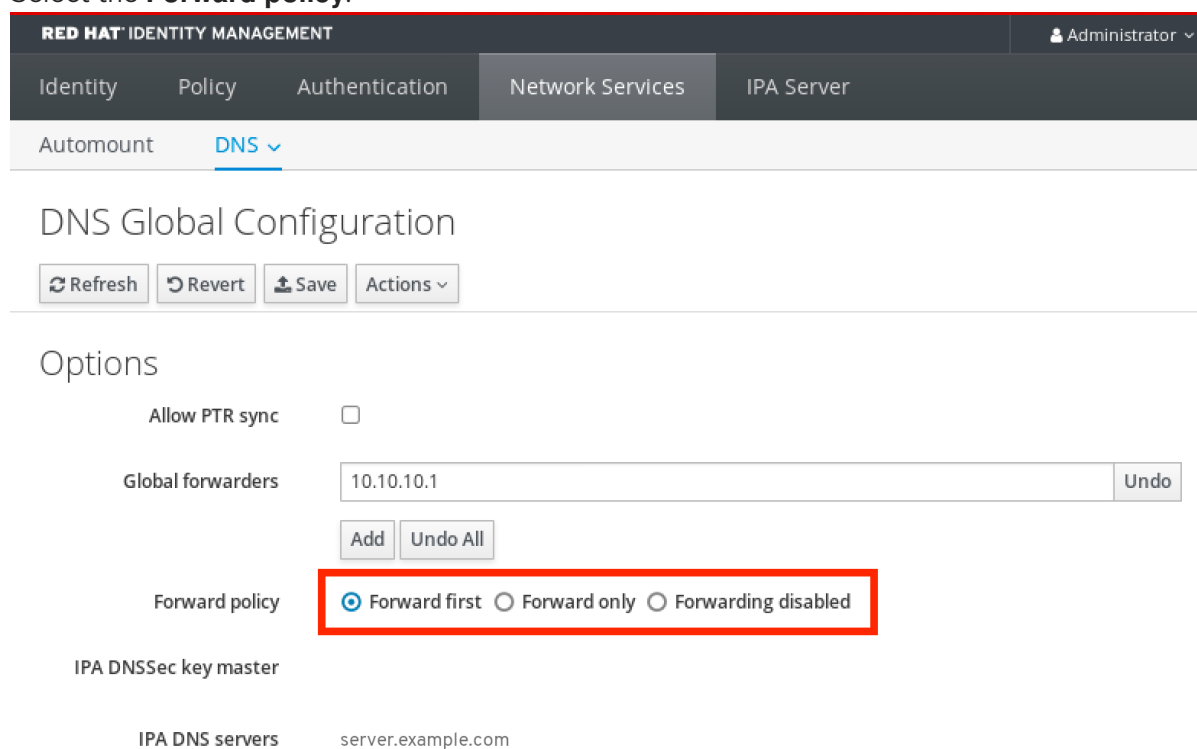
Global forwarders

Forward policy ☒ Forward first ☐ Forward only ☐ Forwarding disabled

IPA DNSSec key master

IPA DNS servers server.example.com

- Select the **Forward policy**.



RED HAT IDENTITY MANAGEMENT Administrator ▾

Identity Policy Authentication **Network Services** IPA Server

Automount **DNS ▾**

DNS Global Configuration

Options

Allow PTR sync ☐

Global forwarders

Forward policy ☒ Forward first ☐ Forward only ☐ Forwarding disabled

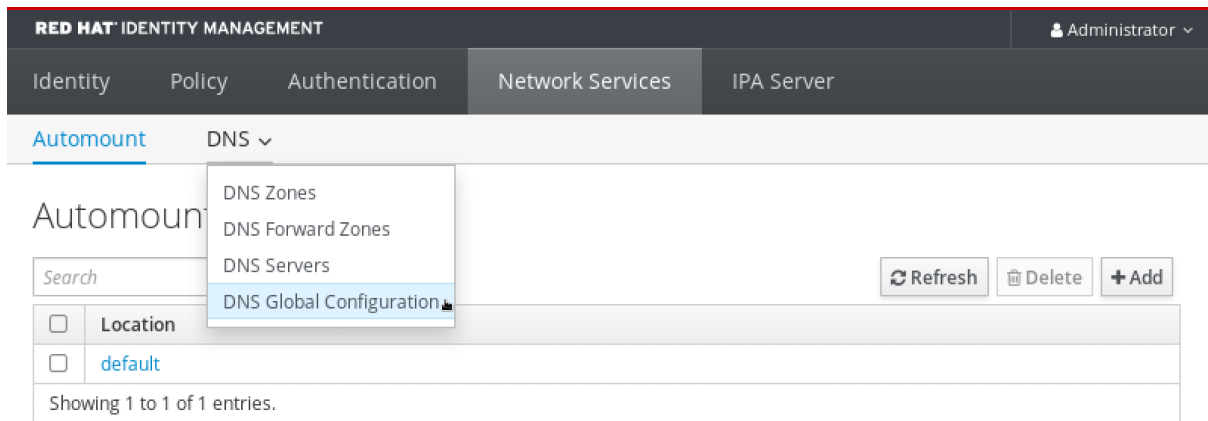
IPA DNSSec key master

IPA DNS servers server.example.com

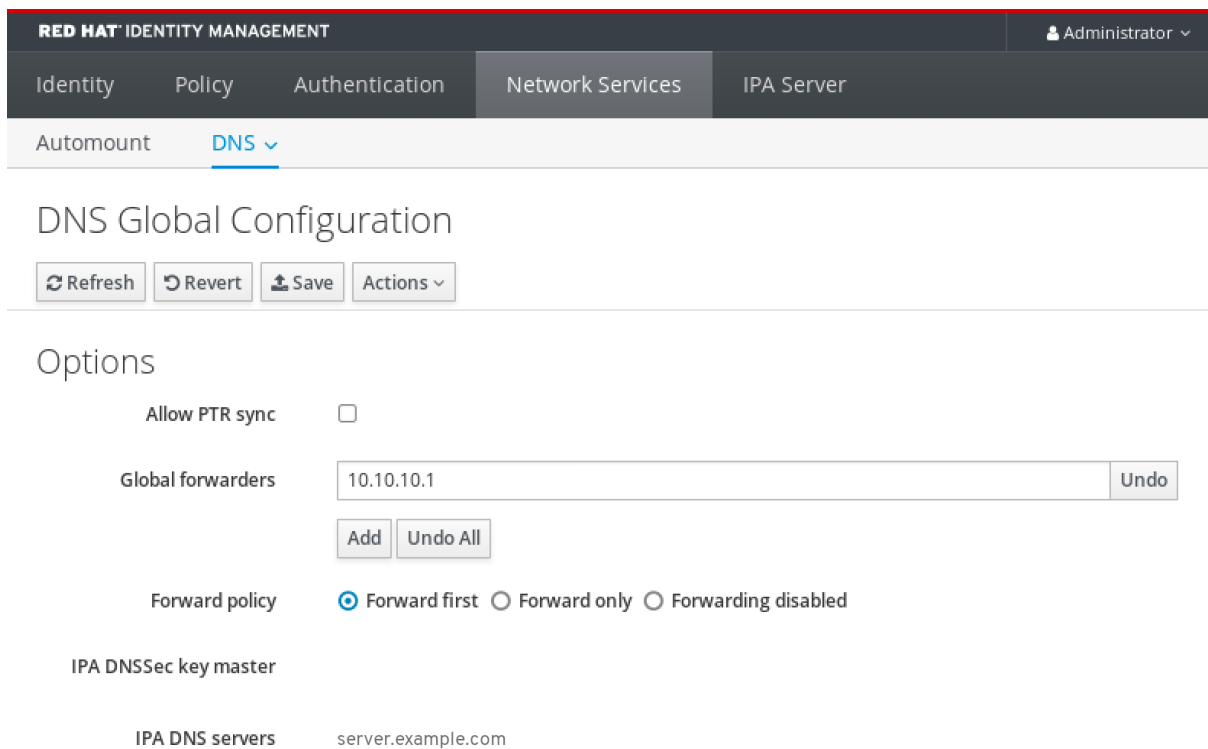
- Click **Save** at the top of the window.

Verification steps

- Select **Network Services** → **DNS Global Configuration** → **DNS**.



2. Verify that the global forwarder, with the forward policy you specified, is present and enabled in the IdM Web UI.



6.4. ADDING A GLOBAL FORWARDER IN THE CLI

Follow this procedure to add a global DNS forwarder by using the command line interface (CLI).

Prerequisites

- You are logged in as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

Procedure

- Use the **ipa dnsconfig-mod** command to add a new global forwarder. Specify the IP address of the DNS forwarder with the **--forwarder** option.

```
[user@server ~]$ ipa dnsconfig-mod --forwarder=10.10.0.1
Server will check DNS forwarder(s).
```

This may take some time, please wait ...
 Global forwarders: 10.10.0.1
 IPA DNS servers: server.example.com

Verification steps

- Use the **dnsconfig-show** command to display global forwarders.

```
[user@server ~]$ ipa dnsconfig-show
Global forwarders: 10.10.0.1
IPA DNS servers: server.example.com
```

6.5. ADDING A DNS FORWARD ZONE IN THE IDM WEB UI

Follow this procedure to add a DNS forward zone in the Identity Management (IdM) Web UI.



IMPORTANT

Do not use forward zones unless absolutely required. Forward zones are not a standard solution, and using them can lead to unexpected and problematic behavior. If you must use forward zones, limit their use to overriding a global forwarding configuration.

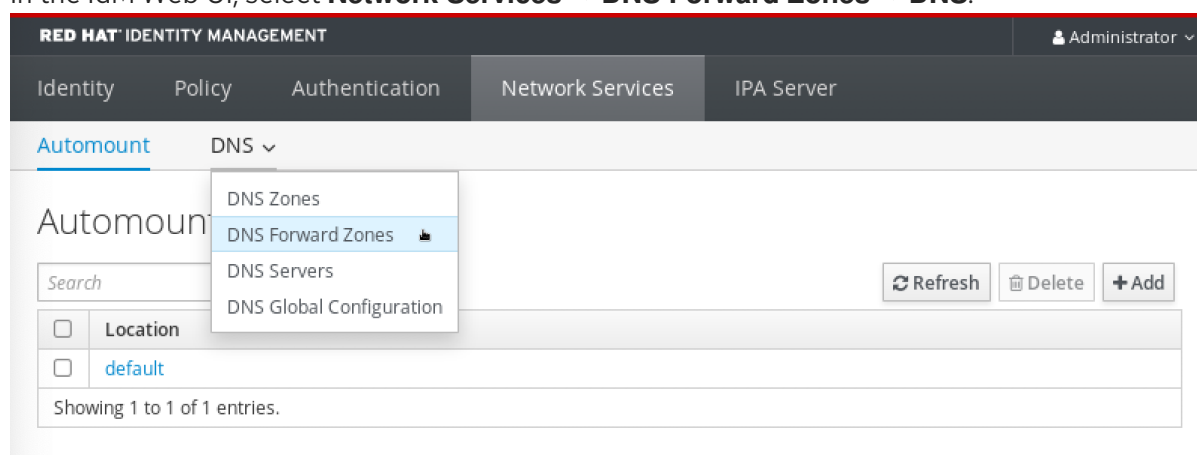
When creating a new DNS zone, Red Hat recommends to always use standard DNS delegation using nameserver (NS) records and to avoid forward zones. In most cases, using a global forwarder is sufficient, and forward zones are not necessary.

Prerequisites

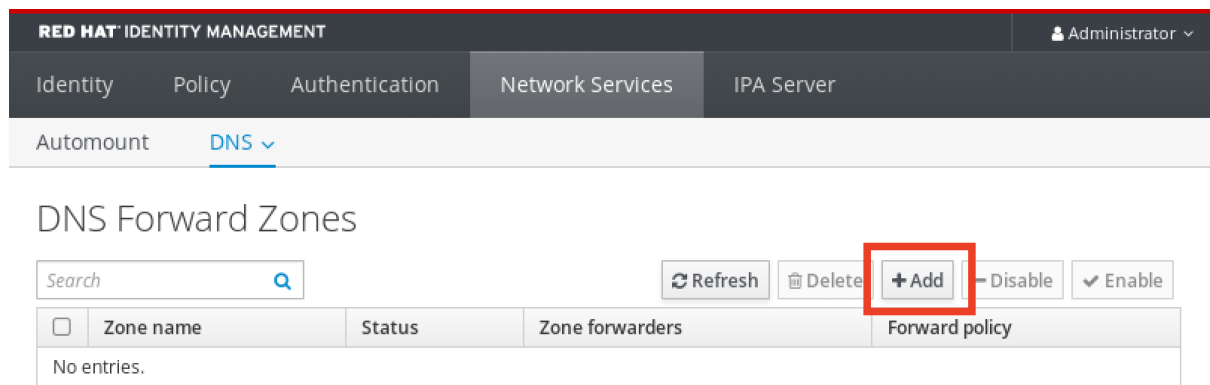
- You are logged in to the IdM WebUI as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

Procedure

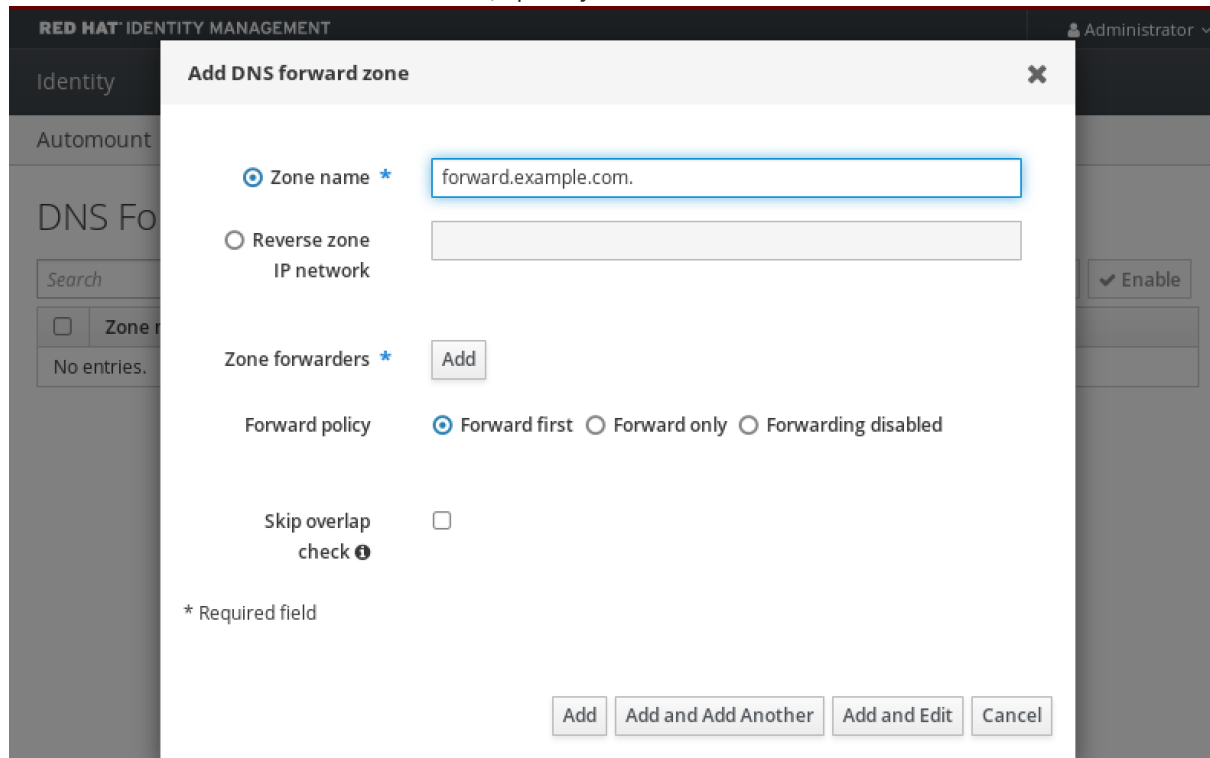
1. In the IdM Web UI, select **Network Services → DNS Forward Zones → DNS**.



2. In the **DNS Forward Zones** section, click **Add**.



3. In the **Add DNS forward zone** window, specify the forward zone name.



4. Click the **Add** button and specify the IP address of a DNS server to receive the forwarding request. You can specify multiple forwarders per forward zone.

Add DNS forward zone

☒ Zone name * forward.example.com.

☐ Reverse zone IP network

Zone forwarders * 10.10.0.14 Undo

Add

Forward policy ☒ Forward first ☐ Forward only ☐ Forwarding disabled

Skip overlap check ☐

* Required field

Add Add and Add Another Add and Edit Cancel

5. Select the **Forward policy**.

Add DNS forward zone

☒ Zone name * forward.example.com

☐ Reverse zone IP network

Zone forwarders * 10.10.0.14 Undo

Add

Forward policy ☒ Forward first ☐ Forward only ☐ Forwarding disabled

Skip overlap check ☐

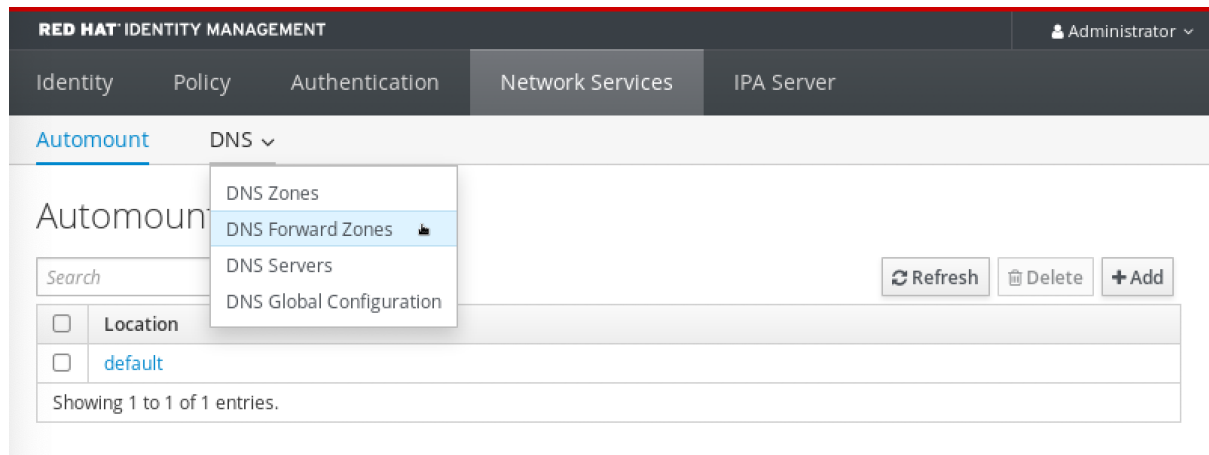
* Required field

Add Add and Add Another Add and Edit Cancel

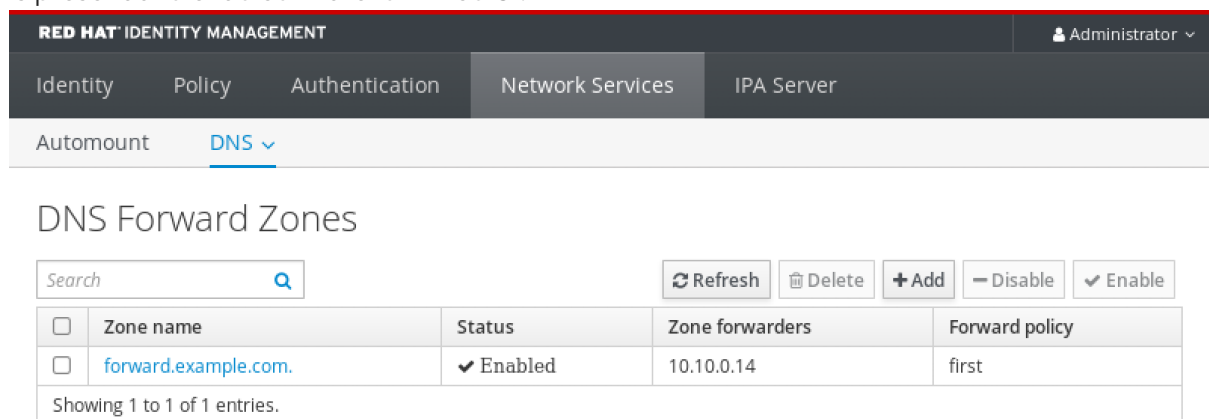
6. Click **Add** at the bottom of the window to add the new forward zone.

Verification steps

1. In the IdM Web UI, select **Network Services** → **DNS Forward Zones** → **DNS**.

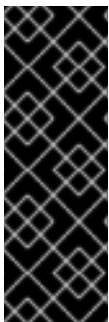


2. Verify that the forward zone you created, with the forwarders and forward policy you specified, is present and enabled in the IdM Web UI.



6.6. ADDING A DNS FORWARD ZONE IN THE CLI

Follow this procedure to add a DNS forward zone by using the command line interface (CLI).



IMPORTANT

Do not use forward zones unless absolutely required. Forward zones are not a standard solution, and using them can lead to unexpected and problematic behavior. If you must use forward zones, limit their use to overriding a global forwarding configuration.

When creating a new DNS zone, Red Hat recommends to always use standard DNS delegation using nameserver (NS) records and to avoid forward zones. In most cases, using a global forwarder is sufficient, and forward zones are not necessary.

Prerequisites

- You are logged in as IdM administrator.
- You know the Internet Protocol (IP) address of the DNS server to forward queries to.

Procedure

- Use the **dnsforwardzone-add** command to add a new forward zone. Specify at least one forwarder with the **--forwarder** option if the forward policy is not **none**, and specify the forward policy with the **--forward-policy** option.

■

```
[user@server ~]$ ipa dnsforwardzone-add forward.example.com. --
forwarder=10.10.0.14 --forwarder=10.10.1.15 --forward-policy=first
```

```
Zone name: forward.example.com.
Zone forwarders: 10.10.0.14, 10.10.1.15
Forward policy: first
```

Verification steps

- Use the **dnsforwardzone-show** command to display the DNS forward zone you just created.

```
[user@server ~]$ ipa dnsforwardzone-show forward.example.com.
```

```
Zone name: forward.example.com.
Zone forwarders: 10.10.0.14, 10.10.1.15
Forward policy: first
```

6.7. ESTABLISHING A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to establish a DNS Global Forwarder in IdM.

In the example procedure below, the IdM administrator creates a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of **8.8.6.6** and IPv6 address of **2001:4860:4860::8800** on port **53**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```


3. Make a copy of the **set-configuration.yml** Ansible playbook file. For example:

```
$ cp set-configuration.yml establish-global-forwarder.yml
```

4. Open the **establish-global-forwarder.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the **name** variable for the playbook to **Playbook to establish a global forwarder in IdM DNS**.
 - b. In the **tasks** section, change the **name** of the task to **Create a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800**.
 - c. In the **forwarders** section of the **ipadnsconfig** portion:
 - i. Change the first **ip_address** value to the IPv4 address of the global forwarder: **8.8.6.6**.
 - ii. Change the second **ip_address** value to the IPv6 address of the global forwarder: **2001:4860:4860::8800**.
 - iii. Verify the **port** value is set to **53**.
 - d. Change the **forward_policy** to **first**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to establish a global forwarder in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Create a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800
    ipadnsconfig:
      forwarders:
        - ip_address: 8.8.6.6
        - ip_address: 2001:4860:4860::8800
      port: 53
      forward_policy: first
      allow_sync_ptr: yes
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file establish-global-forwarder.yml
```

Additional resources

- See the **README-dnsconfig.md** file in the **/usr/share/doc/ansible-freeipa/** directory.

6.8. ENSURING THE PRESENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the presence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS global forwarder to a DNS server with an Internet Protocol (IP) v4 address of **7.7.9.9** and IP v6 address of **2001:db8::1:0** on port **53**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `forwarders-absent.yml` Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-of-a-global-forwarder.yml
```

4. Open the `ensure-presence-of-a-global-forwarder.yml` file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the `name` variable for the playbook to **Playbook to ensure the presence of a global forwarder in IdM DNS**.
 - b. In the `tasks` section, change the `name` of the task to **Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port 53**.
 - c. In the `forwarders` section of the `ipadnsconfig` portion:
 - i. Change the first `ip_address` value to the IPv4 address of the global forwarder: **7.7.9.9**.

- ii. Change the second **ip_address** value to the IPv6 address of the global forwarder:
2001:db8::1:0.
- iii. Verify the **port** value is set to **53**.
- d. Change the **state** to **present**.
This the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the presence of a global forwarder in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure the presence of a DNS global forwarder to 7.7.9.9 and 2001:db8::1:0 on port
    53
    ipadnsconfig:
      forwarders:
        - ip_address: 7.7.9.9
        - ip_address: 2001:db8::1:0
      port: 53
      state: present
```

- 6. Save the file.
- 7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-presence-
of-a-global-forwarder.yml
```

Additional resources

- See the **README-dnsconfig.md** file in the **/usr/share/doc/ansible-freeipa/** directory.

6.9. ENSURING THE ABSENCE OF A DNS GLOBAL FORWARDER IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the absence of a DNS global forwarder in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS global forwarder with an Internet Protocol (IP) v4 address of **8.8.6.6** and IP v6 address of **2001:4860:4860::8800** on port **53**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package on the Ansible controller.
 - The example assumes that in the **~/MyPlaybooks/** directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.

- The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the **/usr/share/doc/ansible-freeipa/playbooks/dnsconfig** directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-absence-of-a-global-forwarder.yml
```

4. Open the **ensure-absence-of-a-global-forwarder.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the **name** variable for the playbook to **Playbook to ensure the absence of a global forwarder in IdM DNS**.
 - b. In the **tasks** section, change the **name** of the task to **Ensure the absence of a DNS global forwarder to 8.8.6.6 and 2001:4860:4860::8800 on port 53**.
 - c. In the **forwarders** section of the **ipadnsconfig** portion:
 - i. Change the first **ip_address** value to the IPv4 address of the global forwarder: **8.8.6.6**.
 - ii. Change the second **ip_address** value to the IPv6 address of the global forwarder: **2001:4860:4860::8800**.
 - iii. Verify the **port** value is set to **53**.
 - d. Set the **action** variable to **member**.
 - e. Verify the **state** is set to **absent**.

This the modified Ansible playbook file for the current example:

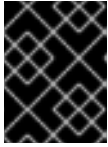
```
---
- name: Playbook to ensure the absence of a global forwarder in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure the absence of a DNS global forwarder to 8.8.6.6 and
    2001:4860:4860::8800 on port 53
```

```

ipadnsconfig:
  forwarders:
    - ip_address: 8.8.6.6
    - ip_address: 2001:4860:4860::8800
  port: 53
  action: member
  state: absent

```



IMPORTANT

If you only use the **state: absent** option in your playbook without also using **action: member**, the playbook fails.

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-absence-of-a-global-forwarder.yml
```

Additional resources

- The **README-dnsconfig.md** file in the `/usr/share/doc/ansible-freeipa/` directory
- The **action: member** option in `ipadnsconfig` `ansible-freeipa` modules

6.10. ENSURING DNS GLOBAL FORWARDERS ARE DISABLED IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure DNS Global Forwarders are disabled in IdM. In the example procedure below, the IdM administrator ensures that the forwarding policy for the global forwarder is set to **none**, which effectively disables the global forwarder.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipadmin_password`.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Verify the contents of the **disable-global-forwarders.yml** Ansible playbook file which is already configured to disable all DNS global forwarders. For example:

```
$ cat disable-global-forwarders.yml
---
- name: Playbook to disable global DNS forwarders
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Disable global forwarders.
    ipadnsconfig:
      forward_policy: none
```

4. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file disable-global-forwarders.yml
```

Additional resources

- See the **README-dnsconfig.md** file in the **/usr/share/doc/ansible-freeipa/** directory.

6.11. ENSURING THE PRESENCE OF A DNS FORWARD ZONE IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the presence of a DNS Forward Zone in IdM. In the example procedure below, the IdM administrator ensures the presence of a DNS forward zone for **example.com** to a DNS server with an Internet Protocol (IP) address of **8.8.8.8**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package on the Ansible controller.
 - The example assumes that in the **~/MyPlaybooks/** directory, you have created an **Ansible inventory file** with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-forwardzone.yml
```

4. Open the **ensure-presence-forwardzone.yml** file for editing.

5. Adapt the file by setting the following variables:

- a. Change the **name** variable for the playbook to **Playbook to ensure the presence of a dnsforwardzone in IdM DNS**.
- b. In the **tasks** section, change the **name** of the task to **Ensure presence of a dnsforwardzone for example.com to 8.8.8.8**.
- c. In the **tasks** section, change the **ipadnsconfig** heading to **ipadnsforwardzone**.
- d. In the **ipadnsforwardzone** section:
 - i. Add the **ipaadmin_password** variable and set it to your IdM administrator password.
 - ii. Add the **name** variable and set it to **example.com**.
 - iii. In the **forwarders** section:
 - A. Remove the **ip_address** and **port** lines.
 - B. Add the IP address of the DNS server to receive forwarded requests by specifying it after a dash:

```
- 8.8.8.8
```

- iv. Add the **forwardpolicy** variable and set it to **first**.
- v. Add the **skip_overlap_check** variable and set it to **true**.
- vi. Change the **state** variable to **present**.

This the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the presence of a dnsforwardzone in IdM DNS
  hosts: ipaserver
```

```
vars_files:
- /home/user_name/MyPlaybooks/secret.yml
tasks:
- name: Ensure the presence of a dnsforwardzone for example.com to 8.8.8.8
  ipadnsforwardzone:
    ipadmin_password: "{{ ipadmin_password }}"
    name: example.com
    forwarders:
      - 8.8.8.8
    forwardpolicy: first
    skip_overlap_check: true
    state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-presence-forwardzone.yml
```

Additional resources

- See the **README-dnsforwardzone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.

6.12. ENSURING A DNS FORWARD ZONE HAS MULTIPLE FORWARDERS IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure a DNS Forward Zone in IdM has multiple forwarders. In the example procedure below, the IdM administrator ensures the DNS forward zone for **example.com** is forwarding to **8.8.8.8** and **4.4.4.4**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```


2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-presence-multiple-forwarders.yml
```

4. Open the **ensure-presence-multiple-forwarders.yml** file for editing.

5. Adapt the file by setting the following variables:

- a. Change the **name** variable for the playbook to **Playbook to ensure the presence of multiple forwarders in a dnsforwardzone in IdM DNS**.
- b. In the **tasks** section, change the **name** of the task to **Ensure presence of 8.8.8.8 and 4.4.4.4 forwarders in dnsforwardzone for example.com**.
- c. In the **tasks** section, change the **ipadnsconfig** heading to **ipadnsforwardzone**.
- d. In the **ipadnsforwardzone** section:
 - i. Add the **ipaadmin_password** variable and set it to your IdM administrator password.
 - ii. Add the **name** variable and set it to **example.com**.
 - iii. In the **forwarders** section:
 - A. Remove the **ip_address** and **port** lines.
 - B. Add the IP address of the DNS servers you want to ensure are present, preceded by a dash:


```
- 8.8.8.8
- 4.4.4.4
```
 - iv. Change the state variable to present.

This the modified Ansible playbook file for the current example:

```
---
- name: name: Playbook to ensure the presence of multiple forwarders in a dnsforwardzone
  in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure presence of 8.8.8.8 and 4.4.4.4 forwarders in dnsforwardzone for
    example.com
    ipadnsforwardzone:
      ipaadmin_password: "{{ ipaadmin_password }}"
      name: example.com
```

```
forwarders:
  - 8.8.8.8
  - 4.4.4.4
state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-presence-
multiple-forwarders.yml
```

Additional resources

- See the **README-dnsforwardzone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.

6.13. ENSURING A DNS FORWARD ZONE IS DISABLED IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure a DNS Forward Zone is disabled in IdM. In the example procedure below, the IdM administrator ensures the DNS forward zone for **example.com** is disabled.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipaadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsconfig` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-disabled-forwardzone.yml
```

4. Open the **ensure-disabled-forwardzone.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the **name** variable for the playbook to **Playbook to ensure a dnsforwardzone is disabled in IdM DNS**.
 - b. In the **tasks** section, change the **name** of the task to **Ensure a dnsforwardzone for example.com is disabled**.
 - c. In the **tasks** section, change the **ipadnsconfig** heading to **ipadnsforwardzone**.
 - d. In the **ipadnsforwardzone** section:
 - i. Add the **ipaadmin_password** variable and set it to your IdM administrator password.
 - ii. Add the **name** variable and set it to **example.com**.
 - iii. Remove the entire **forwarders** section.
 - iv. Change the **state** variable to **disabled**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure a dnsforwardzone is disabled in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure a dnsforwardzone for example.com is disabled
    ipadnsforwardzone:
      ipaadmin_password: "{{ ipaadmin_password }}"
      name: example.com
      state: disabled
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-disabled-forwardzone.yml
```

Additional resources

- See the **README-dnsforwardzone.md** file in the **/usr/share/doc/ansible-freeipa/** directory.

6.14. ENSURING THE ABSENCE OF A DNS FORWARD ZONE IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure the absence of a DNS Forward Zone in IdM. In the example procedure below, the IdM administrator ensures the absence of a DNS forward zone for **example.com**.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the **ansible-freeipa** package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipaadmin_password**.
- You know the IdM administrator password.

Procedure

1. Navigate to the **/usr/share/doc/ansible-freeipa/playbooks/dnsconfig** directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsconfig
```

2. Open your inventory file and make sure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **forwarders-absent.yml** Ansible playbook file. For example:

```
$ cp forwarders-absent.yml ensure-absence-forwardzone.yml
```

4. Open the **ensure-absence-forwardzone.yml** file for editing.
5. Adapt the file by setting the following variables:
 - a. Change the **name** variable for the playbook to **Playbook to ensure the absence of a dnsforwardzone in IdM DNS**.
 - b. In the **tasks** section, change the **name** of the task to **Ensure the absence of a dnsforwardzone for example.com**.
 - c. In the **tasks** section, change the **ipadnsconfig** heading to **ipadnsforwardzone**.
 - d. In the **ipadnsforwardzone** section:
 - i. Add the **ipaadmin_password** variable and set it to your IdM administrator password.
 - ii. Add the **name** variable and set it to **example.com**.
 - iii. Remove the entire **forwarders** section.

- iv. Leave the **state** variable as **absent**.

This the modified Ansible playbook file for the current example:

```
---
- name: Playbook to ensure the absence of a dnsforwardzone in IdM DNS
  hosts: ipaserver

  vars_files:
  - /home/user_name/MyPlaybooks/secret.yml
  tasks:
  - name: Ensure the absence of a dnsforwardzone for example.com
    ipadsforwardzone:
      ipaadmin_password: "{{ ipaadmin_password }}"
      name: example.com
      state: absent
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-absence-forwardzone.yml
```

Additional resources

- See the **README-dnsforwardzone.md** file in the `/usr/share/doc/ansible-freeipa/` directory.

CHAPTER 7. MANAGING DNS RECORDS IN IDM

This chapter describes how to manage DNS records in Identity Management (IdM). As an IdM administrator, you can add, modify and delete DNS records in IdM. The chapter contains the following sections:

- [DNS records in IdM](#)
- [Adding DNS resource records from the IdM Web UI](#)
- [Adding DNS resource records from the IdM CLI](#)
- [Common ipa dnsrecord-add options](#)
- [Deleting DNS records in the IdM Web UI](#)
- [Deleting an entire DNS record in the IdM Web UI](#)
- [Deleting DNS records in the IdM CLI](#)

Prerequisites

- Your IdM deployment contains an integrated DNS server. For information how to install IdM with integrated DNS, see one of the following links:
 - [Installing an IdM server: With integrated DNS, with an integrated CA as the root CA](#) .
 - [Installing an IdM server: With integrated DNS, with an external CA as the root CA](#) .

7.1. DNS RECORDS IN IDM

Identity Management (IdM) supports many different DNS record types. The following four are used most frequently:

A

This is a basic map for a host name and an IPv4 address. The record name of an A record is a host name, such as **www**. The **IP Address** value of an A record is an IPv4 address, such as **192.0.2.1**. For more information about A records, see [RFC 1035](#).

AAAA

This is a basic map for a host name and an IPv6 address. The record name of an AAAA record is a host name, such as **www**. The **IP Address** value is an IPv6 address, such as **2001:DB8::1111**. For more information about AAAA records, see [RFC 3596](#).

SRV

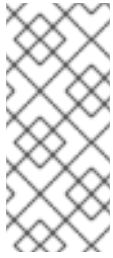
Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the server which manages it.

The record name of an SRV record has the format **_service._protocol**, such as **_ldap._tcp**. The configuration options for SRV records include priority, weight, port number, and host name for the target service.

For more information about SRV records, see [RFC 2782](#).

PTR

A pointer record (PTR) adds a reverse DNS record, which maps an IP address to a domain name.



NOTE

All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the **in-addr.arpa.** domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the **in-addr.arpa.** domain appended to it. For example, for the network address **192.0.2.0/24**, the reverse zone is **2.0.192.in-addr.arpa.**

The record name of a PTR must be in the standard format specified in [RFC 1035](#), extended in [RFC 2317](#), and [RFC 3596](#). The host name value must be a canonical host name of the host for which you want to create the record.



NOTE

Reverse zones can also be configured for IPv6 addresses, with zones in the **.ip6.arpa.** domain. For more information about IPv6 reverse zones, see [RFC 3596](#).

When adding DNS resource records, note that many of the records require different data. For example, a CNAME record requires a host name, while an A record requires an IP address. In the IdM Web UI, the fields in the form for adding a new record are updated automatically to reflect what data is required for the currently selected type of record.

7.2. ADDING DNS RESOURCE RECORDS IN THE IDM WEB UI

Follow this procedure to add DNS resource records in the Identity Management (IdM) Web UI.

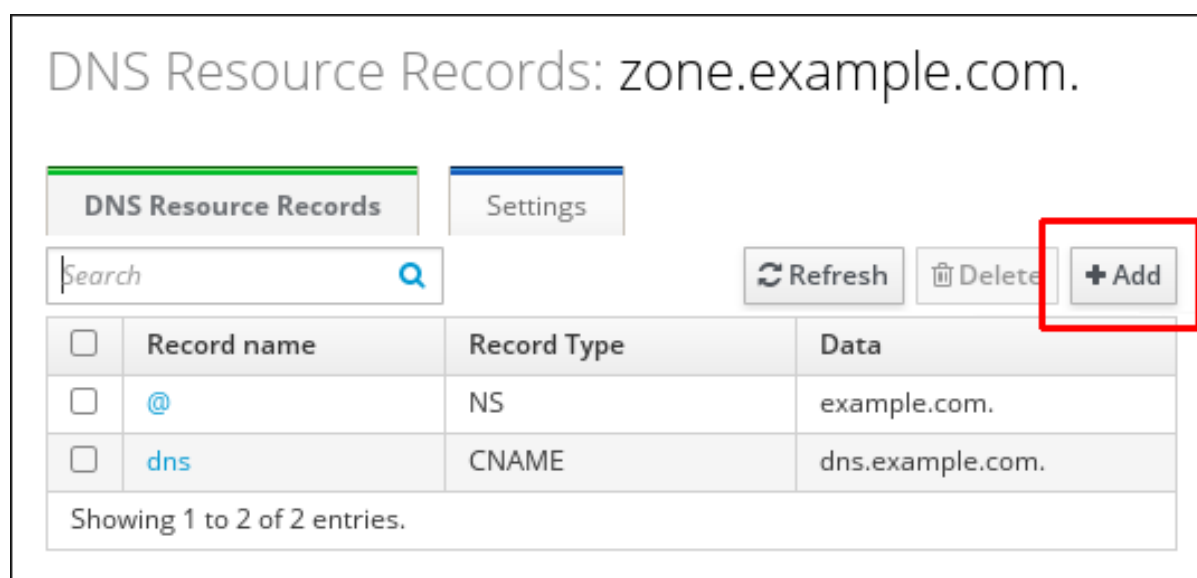
Prerequisites

- The DNS zone to which you want to add a DNS record exists and is managed by IdM. For more information about creating a DNS zone in IdM DNS, see [Managing DNS zones in IdM](#).
- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.
2. Click the DNS zone to which you want to add a DNS record.
3. In the **DNS Resource Records** section, click **Add** to add a new record.

Figure 7.1. Adding a New DNS Resource Record



4. Select the type of record to create and fill out the other fields as required.

Figure 7.2. Defining a New DNS Resource Record

Add DNS Resource Record

Record name *

Record Type

CNAME

Hostname *

* Required field

5. Click **Add** to confirm the new record.

7.3. ADDING DNS RESOURCE RECORDS FROM THE IDM CLI

Follow this procedure to add a DNS resource record of any type from the command line interface (CLI).

Prerequisites

- The DNS zone to which you want to add a DNS records exists. For more information about creating a DNS zone in IdM DNS, see [Managing DNS zones in IdM](#).
- You are logged in as IdM administrator.

Procedure

1. To add a DNS resource record, use the **ipa dnsrecord-add** command. The command follows this syntax:

```
$ ipa dnsrecord-add zone_name record_name --record_type_option=data
```

In the command above:

- The *zone_name* is the name of the DNS zone to which the record is being added.
- The *record_name* is an identifier for the new DNS resource record.

For example, to add an A type DNS record of **host1** to the **idm.example.com** zone, enter:

```
$ ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123
```

7.4. COMMON IPA DNSRECORD-* OPTIONS

You can use the following options when adding, modifying and deleting the most common DNS resource record types in Identity Management (IdM):

- A (IPv4)
- AAAA (IPv6)
- SRV
- PTR

In **Bash**, you can define multiple entries by listing the values in a comma-separated list inside curly braces, such as **--option={val1,val2,val3}**.

Table 7.1. General Record Options

Option	Description
--ttl=number	Sets the time to live for the record.
--structured	Parses the raw DNS records and returns them in a structured format.

Table 7.2. "A" record options

Option	Description	Examples
--a-rec=ARECORD	Passes a single A record or a list of A records.	ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123

Option	Description	Examples
	Can create a wildcard A record with a given IP address.	ipa dnsrecord-add idm.example.com "*" --a-rec=192.168.122.123 ^[a]
--a-ip-address=string	Gives the IP address for the record. When creating a record, the option to specify the A record value is --a-rec . However, when modifying an A record, the --a-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option.	ipa dnsrecord-mod idm.example.com --a-rec 192.168.122.123 --a-ip-address 192.168.122.124
[a] The example creates a wildcard A record with the IP address of 192.0.2.123.		

Table 7.3. "AAAA" record options

Option	Description	Example
--aaaa-rec=AAAARECORD	Passes a single AAAA (IPv6) record or a list of AAAA records.	ipa dnsrecord-add idm.example.com www --aaaa-rec 2001:db8::1231:5675
--aaaa-ip-address=string	Gives the IPv6 address for the record. When creating a record, the option to specify the A record value is --aaaa-rec . However, when modifying an A record, the --aaaa-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option.	ipa dnsrecord-mod idm.example.com --aaaa-rec 2001:db8::1231:5675 --aaaa-ip-address 2001:db8::1231:5676

Table 7.4. "PTR" record options

Option	Description	Example
--ptr-rec=PTRRECORD	Passes a single PTR record or a list of PTR records. When adding the reverse DNS record, the zone name used with the ipa dnsrecord-add command is reversed, compared to the usage for adding other DNS records. Typically, the host IP address is the last octet of the IP address in a given network. The first example on the right adds a PTR record for server4.idm.example.com with IPv4 address 192.168.122.4 . The second example adds a reverse DNS entry to the 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa . IPv6 reverse zone for the host server2.example.com with the IP address 2001:DB8::1111 .	ipa dnsrecord-add 122.168.192.in-addr.arpa 4 --ptr-rec server4.idm.example.com. \$ ipa dnsrecord-add 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. 1.1.1.0.0.0.0.0.0.0.0.0.0.0 --ptr-rec server2.idm.example.com.

Option	Description	Example
--ptr-hostname=string	Gives the host name for the record.	

Table 7.5. "SRV" Record Options

Option	Description	Example
--srv-rec=SRVRECORD	Passes a single SRV record or a list of SRV records. In the examples on the right, <code>_ldap._tcp</code> defines the service type and the connection protocol for the SRV record. The --srv-rec option defines the priority, weight, port, and target values. The weight values of 51 and 49 in the examples add up to 100 and represent the probability, in percentages, that a particular record is used.	<pre># ipa dnsrecord-add idm.example.com _ldap._tcp --srv-rec="0 51 389 server1.idm.example.com."</pre> <pre># ipa dnsrecord-add server.idm.example.com _ldap._tcp --srv-rec="1 49 389 server2.idm.example.com."</pre>
--srv-priority=number	Sets the priority of the record. There can be multiple SRV records for a service type. The priority (0 - 65535) sets the rank of the record; the lower the number, the higher the priority. A service has to use the record with the highest priority first.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="1 49 389 server2.idm.example.com." --srv-priority=0</pre>
--srv-weight=number	Sets the weight of the record. This helps determine the order of SRV records with the same priority. The set weights should add up to 100, representing the probability (in percentages) that a particular record is used.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="0 49 389 server2.idm.example.com." --srv-weight=60</pre>
--srv-port=number	Gives the port for the service on the target host.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="0 60 389 server2.idm.example.com." --srv-port=636</pre>
--srv-target=string	Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.	

Additional resources

- Run **ipa dnsrecord-add --help**.

7.5. DELETING DNS RECORDS IN THE IDM WEB UI

Follow this procedure to delete DNS records in Identity Management (IdM) using the IdM Web UI.

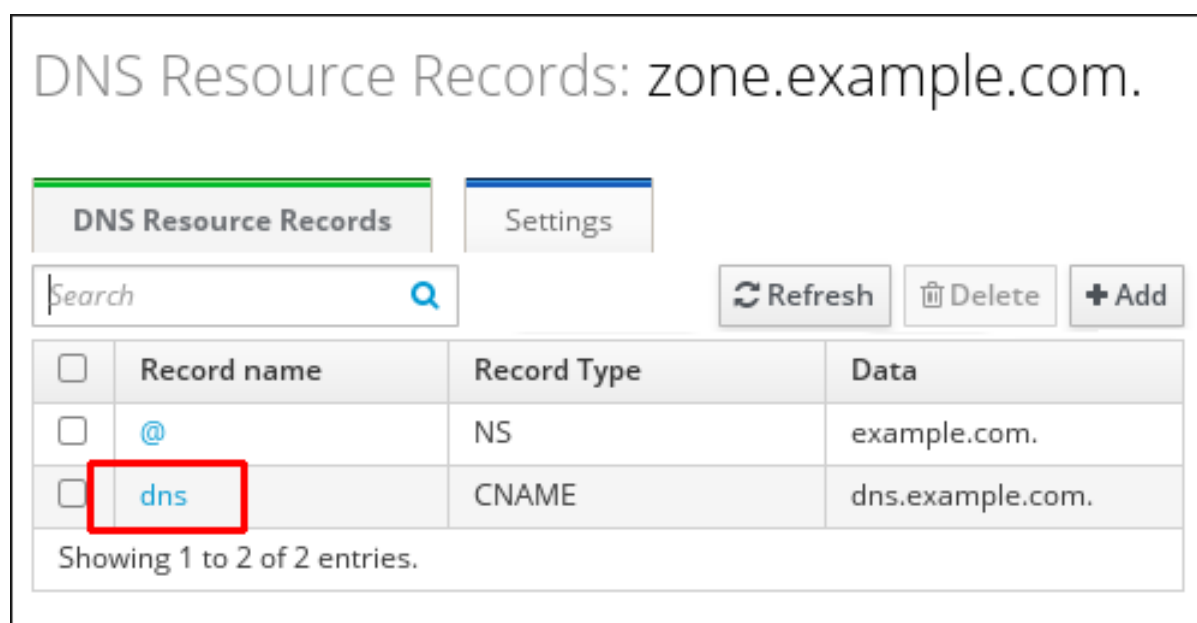
Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.
2. Click the zone from which you want to delete a DNS record, for example **example.com..**
3. In the **DNS Resource Records** section, click the name of the resource record.

Figure 7.3. Selecting a DNS Resource Record



4. Select the check box by the name of the record type to delete.
5. Click **Delete**.

Figure 7.4. Deleting a DNS Resource Record

The screenshot shows a web interface titled "Standard Record Types". It contains three main sections for different record types: A, AAAA, and CNAME. Each section has a table with columns for a checkbox, a description, and action buttons. The CNAME section is expanded, showing a table with one record: "dns.example.com.". The "Delete" button for this record is highlighted with a red rectangle.

Record Type	Checkbox	Description	Actions
A	<input type="checkbox"/>	IP Address	Delete + Add
AAAA	<input type="checkbox"/>	IP Address	Delete + Add
CNAME	<input type="checkbox"/>	Hostname	Delete + Add
	<input checked="" type="checkbox"/>	dns.example.com.	Edit

The selected record type is now deleted. The other configuration of the resource record is left intact.

Additional resources

- See [Deleting an entire DNS record in the IdM Web UI](#) .

7.6. DELETING AN ENTIRE DNS RECORD IN THE IDM WEB UI

Follow this procedure to delete all the records for a particular resource in a zone using the Identity Management (IdM) Web UI.

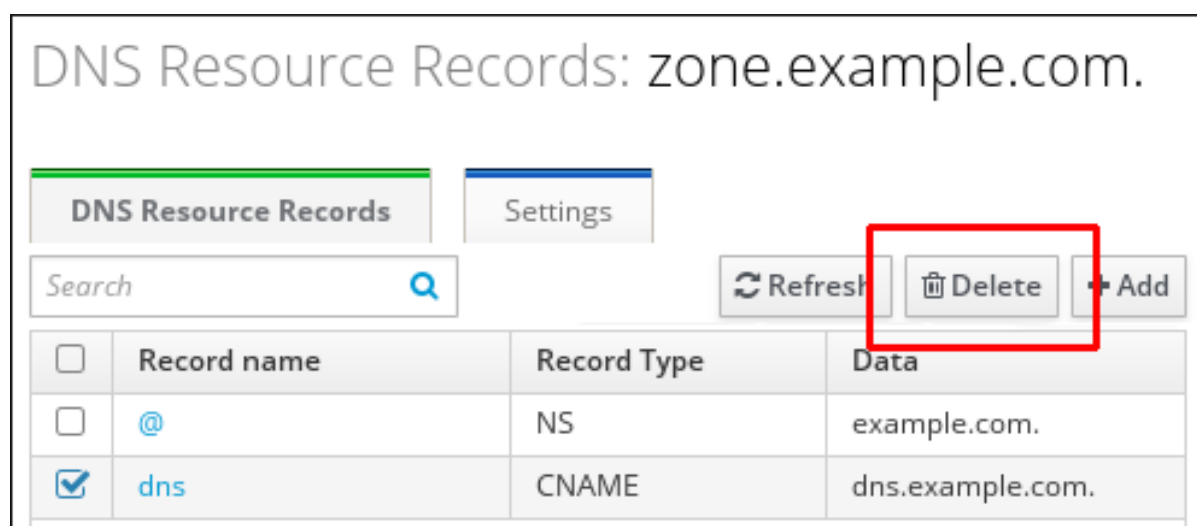
Prerequisites

- You are logged in as IdM administrator.

Procedure

1. In the IdM Web UI, click **Network Services** → **DNS** → **DNS Zones**.
2. Click the zone from which you want to delete a DNS record, for example **zone.example.com.**.
3. In the **DNS Resource Records** section, select the check box of the resource record to delete.
4. Click **Delete**.

Figure 7.5. Deleting an Entire Resource Record



The entire resource record is now deleted.

7.7. DELETING DNS RECORDS IN THE IDM CLI

Follow this procedure to remove DNS records from a zone managed by the Identity Management (IdM) DNS.

Prerequisites

- You are logged in as IdM administrator.

Procedure

- To remove records from a zone, use the **ipa dnsrecord-del** command and add the **-recordType-rec** option together with the record value. For example, to remove an A type record:

```
$ ipa dnsrecord-del example.com www --a-rec 192.0.2.1
```

If you run **ipa dnsrecord-del** without any options, the command prompts for information about the record to delete. Note that passing the **--del-all** option with the command removes all associated records for the zone.

Additional resources

- Run the **ipa dnsrecord-del --help** command.

7.8. ADDITIONAL RESOURCES

- See [Using Ansible to manage DNS records in IdM](#) .

CHAPTER 8. USING ANSIBLE TO MANAGE DNS RECORDS IN IDM

This chapter describes how to manage DNS records in Identity Management (IdM) using an Ansible playbook. As an IdM administrator, you can add, modify, and delete DNS records in IdM. The chapter contains the following sections:

- [Ensuring the presence of A and AAAA DNS records in IdM using Ansible](#)
- [Ensuring the presence of A and PTR DNS records in IdM using Ansible](#)
- [Ensuring the presence of multiple DNS records in IdM using Ansible](#)
- [Ensuring the presence of multiple CNAME records in IdM using Ansible](#)
- [Ensuring the presence of an SRV record in IdM using Ansible](#)

8.1. DNS RECORDS IN IDM

Identity Management (IdM) supports many different DNS record types. The following four are used most frequently:

A

This is a basic map for a host name and an IPv4 address. The record name of an A record is a host name, such as **www**. The **IP Address** value of an A record is an IPv4 address, such as **192.0.2.1**. For more information about A records, see [RFC 1035](#).

AAAA

This is a basic map for a host name and an IPv6 address. The record name of an AAAA record is a host name, such as **www**. The **IP Address** value is an IPv6 address, such as **2001:DB8::1111**. For more information about AAAA records, see [RFC 3596](#).

SRV

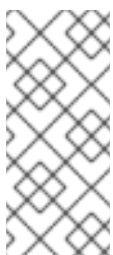
Service (SRV) resource records map service names to the DNS name of the server that is providing that particular service. For example, this record type can map a service like an LDAP directory to the server which manages it.

The record name of an SRV record has the format **_service._protocol**, such as **_ldap._tcp**. The configuration options for SRV records include priority, weight, port number, and host name for the target service.

For more information about SRV records, see [RFC 2782](#).

PTR

A pointer record (PTR) adds a reverse DNS record, which maps an IP address to a domain name.



NOTE

All reverse DNS lookups for IPv4 addresses use reverse entries that are defined in the **in-addr.arpa** domain. The reverse address, in human-readable form, is the exact reverse of the regular IP address, with the **in-addr.arpa** domain appended to it. For example, for the network address **192.0.2.0/24**, the reverse zone is **2.0.192.in-addr.arpa**.

The record name of a PTR must be in the standard format specified in [RFC 1035](#), extended in [RFC 2317](#), and [RFC 3596](#). The host name value must be a canonical host name of the host for which you want to create the record.



NOTE

Reverse zones can also be configured for IPv6 addresses, with zones in the **.ip6.arpa.** domain. For more information about IPv6 reverse zones, see [RFC 3596](#).

When adding DNS resource records, note that many of the records require different data. For example, a CNAME record requires a host name, while an A record requires an IP address. In the IdM Web UI, the fields in the form for adding a new record are updated automatically to reflect what data is required for the currently selected type of record.

8.2. COMMON IPA DNSRECORD-* OPTIONS

You can use the following options when adding, modifying and deleting the most common DNS resource record types in Identity Management (IdM):

- A (IPv4)
- AAAA (IPv6)
- SRV
- PTR

In **Bash**, you can define multiple entries by listing the values in a comma-separated list inside curly braces, such as **--option={val1,val2,val3}**.

Table 8.1. General Record Options

Option	Description
--ttl=number	Sets the time to live for the record.
--structured	Parses the raw DNS records and returns them in a structured format.

Table 8.2. "A" record options

Option	Description	Examples
--a-rec=ARECORD	Passes a single A record or a list of A records.	ipa dnsrecord-add idm.example.com host1 --a-rec=192.168.122.123
	Can create a wildcard A record with a given IP address.	ipa dnsrecord-add idm.example.com "*" --a-rec=192.168.122.123 ^[a]

Option	Description	Examples
--a-ip-address=string	Gives the IP address for the record. When creating a record, the option to specify the A record value is --a-rec . However, when modifying an A record, the --a-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option.	ipa dnsrecord-mod idm.example.com --a-rec 192.168.122.123 --a-ip- address 192.168.122.124
[a] The example creates a wildcard A record with the IP address of 192.0.2.123.		

Table 8.3. "AAAA" record options

Option	Description	Example
--aaaa-rec=AAAARECORD	Passes a single AAAA (IPv6) record or a list of AAAA records.	ipa dnsrecord-add idm.example.com www -- aaaa-rec 2001:db8::1231:5675
--aaaa-ip-address=string	Gives the IPv6 address for the record. When creating a record, the option to specify the A record value is --aaaa-rec . However, when modifying an A record, the --aaaa-rec option is used to specify the current value for the A record. The new value is set with the --a-ip-address option.	ipa dnsrecord-mod idm.example.com --aaaa-rec 2001:db8::1231:5675 --aaaa- ip-address 2001:db8::1231:5676

Table 8.4. "PTR" record options

Option	Description	Example
--ptr-rec=PTRRECORD	Passes a single PTR record or a list of PTR records. When adding the reverse DNS record, the zone name used with the ipa dnsrecord-add command is reversed, compared to the usage for adding other DNS records. Typically, the host IP address is the last octet of the IP address in a given network. The first example on the right adds a PTR record for server4.idm.example.com with IPv4 address 192.168.122.4 . The second example adds a reverse DNS entry to the 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa . IPv6 reverse zone for the host server2.example.com with the IP address 2001:DB8::1111 .	ipa dnsrecord-add 122.168.192.in-addr.arpa 4 -- ptr-rec server4.idm.example.com. \$ ipa dnsrecord-add 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.i p6.arpa. 1.1.1.0.0.0.0.0.0.0.0.0.0.0 -- ptr-rec server2.idm.example.com.
--ptr-hostname=string	Gives the host name for the record.	

Table 8.5. "SRV" Record Options

Option	Description	Example
--srv-rec=SRVRECORD	Passes a single SRV record or a list of SRV records. In the examples on the right, <code>_ldap._tcp</code> defines the service type and the connection protocol for the SRV record. The --srv-rec option defines the priority, weight, port, and target values. The weight values of 51 and 49 in the examples add up to 100 and represent the probability, in percentages, that a particular record is used.	<pre># ipa dnsrecord-add idm.example.com _ldap._tcp --srv-rec="0 51 389 server1.idm.example.com."</pre> <pre># ipa dnsrecord-add server.idm.example.com _ldap._tcp --srv-rec="1 49 389 server2.idm.example.com."</pre>
--srv-priority=number	Sets the priority of the record. There can be multiple SRV records for a service type. The priority (0 - 65535) sets the rank of the record; the lower the number, the higher the priority. A service has to use the record with the highest priority first.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="1 49 389 server2.idm.example.com." --srv-priority=0</pre>
--srv-weight=number	Sets the weight of the record. This helps determine the order of SRV records with the same priority. The set weights should add up to 100, representing the probability (in percentages) that a particular record is used.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="0 49 389 server2.idm.example.com." --srv-weight=60</pre>
--srv-port=number	Gives the port for the service on the target host.	<pre># ipa dnsrecord-mod server.idm.example.com _ldap._tcp --srv-rec="0 60 389 server2.idm.example.com." --srv-port=636</pre>
--srv-target=string	Gives the domain name of the target host. This can be a single period (.) if the service is not available in the domain.	

Additional resources

- Run **ipa dnsrecord-add --help**.

8.3. ENSURING THE PRESENCE OF A AND AAAA DNS RECORDS IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure that A and AAAA records for a particular IdM host are present. In the example used in the procedure below, an IdM administrator ensures the presence of A and AAAA records for **host1** in the **idm.example.com** DNS zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipadmin_password`.
- You know the IdM administrator password.
- The `idm.example.com` zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see [Using Ansible playbooks to manage IdM DNS zones](#) .

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `ensure-A-and-AAAA-records-are-present.yml` Ansible playbook file. For example:

```
$ cp ensure-A-and-AAAA-records-are-present.yml ensure-A-and-AAAA-records-are-present-copy.yml
```

4. Open the `ensure-A-and-AAAA-records-are-present-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipadnsrecord` task section:
 - Set the `ipadmin_password` variable to your IdM administrator password.
 - Set the `zone_name` variable to `idm.example.com`.
 - In the `records` variable, set the `name` variable to `host1`, and the `a_ip_address` variable to `192.168.122.123`.
 - In the `records` variable, set the `name` variable to `host1`, and the `aaaa_ip_address` variable to `::1`.
 This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure A and AAAA records are present
  hosts: ipaserver
  become: true
  gather_facts: false
```

```

tasks:
# Ensure A and AAAA records are present
- name: Ensure that 'host1' has A and AAAA records.
  ipadsrecord:
    ipadmin_password: "{{ ipadmin_password }}"
    zone_name: idm.example.com
  records:
    - name: host1
      a_ip_address: 192.168.122.123
    - name: host1
      aaaa_ip_address: ::1

```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-A-and-AAAA-records-are-present-copy.yml
```

Additional resources

- See [DNS records in IdM](#).
- See the **README-dnsrecord.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

8.4. ENSURING THE PRESENCE OF A AND PTR DNS RECORDS IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure that an A record for a particular IdM host is present, with a corresponding PTR record. In the example used in the procedure below, an IdM administrator ensures the presence of A and PTR records for **host1** with an IP address of **192.168.122.45** in the **idm.example.com** zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.
- The **idm.example.com** DNS zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see [Using Ansible playbooks to manage IdM DNS zones](#).

Procedure

1. Navigate to the **/usr/share/doc/ansible-freeipa/playbooks/dnsrecord** directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **ensure-dnsrecord-with-reverse-is-present.yml** Ansible playbook file. For example:

```
$ cp ensure-dnsrecord-with-reverse-is-present.yml ensure-dnsrecord-with-reverse-is-present-copy.yml
```

4. Open the **ensure-dnsrecord-with-reverse-is-present-copy.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnsrecord** task section:

- Set the **ipaadmin_password** variable to your IdM administrator password.
- Set the **name** variable to **host1**.
- Set the **zone_name** variable to **idm.example.com**.
- Set the **ip_address** variable to **192.168.122.45**.
- Set the **create_reverse** variable to **yes**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure DNS Record is present.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
    # Ensure that dns record is present
    - ipadnsrecord:
        ipaadmin_password: "{{ ipaadmin_password }}"
        name: host1
        zone_name: idm.example.com
        ip_address: 192.168.122.45
        create_reverse: yes
        state: present
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-
dnsrecord-with-reverse-is-present-copy.yml
```

Additional resources

- See [DNS records in IdM](#).
- See the **README-dnsrecord.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

8.5. ENSURING THE PRESENCE OF MULTIPLE DNS RECORDS IN IDM USING ANSIBLE

Follow this procedure to use an Ansible playbook to ensure that multiple values are associated with a particular IdM DNS record. In the example used in the procedure below, an IdM administrator ensures the presence of multiple A records for **host1** in the **idm.example.com** DNS zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the **secret.yml** Ansible vault stores your **ipadmin_password**.
- You know the IdM administrator password.
- The **idm.example.com** zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see [Using Ansible playbooks to manage IdM DNS zones](#).

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the **ensure-presence-multiple-records.yml** Ansible playbook file. For example:

```
$ cp ensure-presence-multiple-records.yml ensure-presence-multiple-records-
copy.yml
```

4. Open the **ensure-presence-multiple-records-copy.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnsrecord** task section:
 - Set the **ipaadmin_password** variable to your IdM administrator password.
 - In the **records** section, set the **name** variable to **host1**.
 - In the **records** section, set the **zone_name** variable to **idm.example.com**.
 - In the **records** section, set the **a_rec** variable to **192.168.122.112** and to **192.168.122.122**.
 - Define a second record in the **records** section:
 - Set the **name** variable to **host1**.
 - Set the **zone_name** variable to **idm.example.com**.
 - Set the **aaaa_rec** variable to **::1**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Test multiple DNS Records are present.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
    # Ensure that multiple dns records are present
    - ipadnsrecord:
        ipaadmin_password: "{{ ipaadmin_password }}"
        records:
          - name: host1
            zone_name: idm.example.com
            a_rec: 192.168.122.112
            a_rec: 192.168.122.122
          - name: host1
            zone_name: idm.example.com
            aaaa_rec: ::1
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-
presence-multiple-records-copy.yml
```

Additional resources

- See [DNS records in IdM](#).
- See the **README-dnsrecord.md** file in the **/usr/share/doc/ansible-freeipa/** directory.

- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

8.6. ENSURING THE PRESENCE OF MULTIPLE CNAME RECORDS IN IDM USING ANSIBLE

A Canonical Name record (CNAME record) is a type of resource record in the Domain Name System (DNS) that maps one domain name, an alias, to another name, the canonical name.

You may find CNAME records useful when running multiple services from a single IP address: for example, an FTP service and a web service, each running on a different port.

Follow this procedure to use an Ansible playbook to ensure that multiple CNAME records are present in IdM DNS. In the example used in the procedure below, **host03** is both an HTTP server and an FTP server. The IdM administrator ensures the presence of the **www** and **ftp** CNAME records for the **host03** A record in the **idm.example.com** zone.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipaadmin_password`.
- You know the IdM administrator password.
- The **idm.example.com** zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see [Using Ansible playbooks to manage IdM DNS zones](#).
- The **host03** A record exists in the **idm.example.com** zone.

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the **[ipaserver]** section. For example, to instruct Ansible to configure **server.idm.example.com**, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `ensure-CNAME-record-is-present.yml` Ansible playbook file. For example:

```
$ cp ensure-CNAME-record-is-present.yml ensure-CNAME-record-is-present-copy.yml
```


4. Open the **ensure-CNAME-record-is-present-copy.yml** file for editing.
5. Adapt the file by setting the following variables in the **ipadnsrecord** task section:
 - (Optional) Adapt the description provided by the **name** of the play.
 - Set the **ipaadmin_password** variable to your IdM administrator password.
 - Set the **zone_name** variable to **idm.example.com**.
 - In the **records** variable section, set the following variables and values:
 - Set the **name** variable to **www**.
 - Set the **cname_hostname** variable to **host03**.
 - Set the **name** variable to **ftp**.
 - Set the **cname_hostname** variable to **host03**.

This is the modified Ansible playbook file for the current example:

```
---
- name: Ensure that 'www.idm.example.com' and 'ftp.idm.example.com' CNAME records
  point to 'host03.idm.example.com'.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
  - ipadnsrecord:
    ipaadmin_password: "{{ ipaadmin_password }}"
    zone_name: idm.example.com
    records:
    - name: www
      cname_hostname: host03
    - name: ftp
      cname_hostname: host03
```

6. Save the file.
7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-
CNAME-record-is-present.yml
```

Additional resources

- See the **README-dnsrecord.md** file in the **/usr/share/doc/ansible-freeipa/** directory.
- See sample Ansible playbooks in the **/usr/share/doc/ansible-freeipa/playbooks/dnsrecord** directory.

8.7. ENSURING THE PRESENCE OF AN SRV RECORD IN IDM USING ANSIBLE

A DNS service (SRV) record defines the hostname, port number, transport protocol, priority and weight of a service available in a domain. In Identity Management (IdM), you can use SRV records to locate IdM servers and replicas.

Follow this procedure to use an Ansible playbook to ensure that an SRV record is present in IdM DNS. In the example used in the procedure below, an IdM administrator ensures the presence of the `_kerberos._udp.idm.example.com` SRV record with the value of `10 50 88 idm.example.com`. This sets the following values:

- It sets the priority of the service to 10.
- It sets the weight of the service to 50.
- It sets the port to be used by the service to 88.

Prerequisites

- You have configured your Ansible control node to meet the following requirements:
 - You are using Ansible version 2.14 or later.
 - You have installed the [ansible-freeipa](#) package on the Ansible controller.
 - The example assumes that in the `~/MyPlaybooks/` directory, you have created an [Ansible inventory file](#) with the fully-qualified domain name (FQDN) of the IdM server.
 - The example assumes that the `secret.yml` Ansible vault stores your `ipadmin_password`.
- You know the IdM administrator password.
- The `idm.example.com` zone exists and is managed by IdM DNS. For more information about adding a primary DNS zone in IdM DNS, see [Using Ansible playbooks to manage IdM DNS zones](#) .

Procedure

1. Navigate to the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory:

```
$ cd /usr/share/doc/ansible-freeipa/playbooks/dnsrecord
```

2. Open your inventory file and ensure that the IdM server that you want to configure is listed in the `[ipaserver]` section. For example, to instruct Ansible to configure `server.idm.example.com`, enter:

```
[ipaserver]
server.idm.example.com
```

3. Make a copy of the `ensure-SRV-record-is-present.yml` Ansible playbook file. For example:

```
$ cp ensure-SRV-record-is-present.yml ensure-SRV-record-is-present-copy.yml
```

4. Open the `ensure-SRV-record-is-present-copy.yml` file for editing.
5. Adapt the file by setting the following variables in the `ipadnsrecord` task section:
 - Set the `ipadmin_password` variable to your IdM administrator password.

- Set the **name** variable to `_kerberos._udp.idm.example.com`.
 - Set the **srv_rec** variable to `'10 50 88 idm.example.com'`.
 - Set the **zone_name** variable to `idm.example.com`.
- This the modified Ansible playbook file for the current example:

```
---
- name: Test multiple DNS Records are present.
  hosts: ipaserver
  become: true
  gather_facts: false

  tasks:
    # Ensure a SRV record is present
    - ipadsrecord:
        ipaadmin_password: "{{ ipaadmin_password }}"
        name: _kerberos._udp.idm.example.com
        srv_rec: '10 50 88 idm.example.com'
        zone_name: idm.example.com
        state: present
```

6. Save the file.

7. Run the playbook:

```
$ ansible-playbook --vault-password-file=password_file -v -i inventory.file ensure-SRV-
record-is-present.yml
```

Additional resources

- See [DNS records in IdM](#).
- See the **README-dnsrecord.md** file in the `/usr/share/doc/ansible-freeipa/` directory.
- See sample Ansible playbooks in the `/usr/share/doc/ansible-freeipa/playbooks/dnsrecord` directory.

CHAPTER 9. USING CANONICALIZED DNS HOST NAMES IN IDM

DNS canonicalization is disabled by default on Identity Management (IdM) clients to avoid potential security risks. For example, if an attacker controls the DNS server and a host in the domain, the attacker can cause the short host name, such as **demo**, to resolve to a compromised host, such as **malicious.example.com**. In this case, the user connects to a different server than expected.

This procedure describes how to use canonicalized host names on IdM clients.

9.1. ADDING AN ALIAS TO A HOST PRINCIPAL

By default, Identity Management (IdM) clients enrolled by using the **ipa-client-install** command do not allow to use short host names in service principals. For example, users can use only **host/demo.example.com@EXAMPLE.COM** instead of **host/demo@EXAMPLE.COM** when accessing a service.

Follow this procedure to add an alias to a Kerberos principal. Note that you can alternatively enable canonicalization of host names in the **/etc/krb5.conf** file. For details, see [Enabling canonicalization of host names in service principals on clients](#).

Prerequisites

- The IdM client is installed.
- The host name is unique in the network.

Procedure

1. Authenticate to IdM as the **admin** user:

```
$ kinit admin
```

2. Add the alias to the host principal. For example, to add the **demo** alias to the **demo.example.com** host principal:

```
$ ipa host-add-principal demo.example.com --principal=demo
```

9.2. ENABLING CANONICALIZATION OF HOST NAMES IN SERVICE PRINCIPALS ON CLIENTS

Follow this procedure to enable canonicalization of host names in services principals on clients.

Note that if you use host principal aliases, as described in [Adding an alias to a host principal](#), you do not need to enable canonicalization.

Prerequisites

- The Identity Management (IdM) client is installed.
- You are logged in to the IdM client as the **root** user.
- The host name is unique in the network.

Procedure

1. Set the **dns_canonicalize_hostname** parameter in the **[libdefaults]** section in the **/etc/krb5.conf** file to **false**:

```
[libdefaults]
...
dns_canonicalize_hostname = true
```

9.3. OPTIONS FOR USING HOST NAMES WITH DNS HOST NAME CANONICALIZATION ENABLED

If you set **dns_canonicalize_hostname = true** in the **/etc/krb5.conf** file as explained in [Enabling canonicalization of host names in service principals on clients](#), you have the following options when you use a host name in a service principal:

- In Identity Management (IdM) environments, you can use the full host name in a service principal, such as **host/demo.example.com@EXAMPLE.COM**.
- In environments without IdM, but if the RHEL host is a member of an Active Directory (AD) domain, no further considerations are required, because AD domain controllers (DC) automatically create service principals for NetBIOS names of the machines enrolled into AD.