Designate Documentation

Release 20.0.0.0rc2.dev2

Designate Developers

Mar 18, 2025

CONTENTS

Designate is a multi-tenant DNSaaS service for OpenStack. It provides a REST API with integrated Keystone authentication. It can be configured to auto-generate records based on Nova and Neutron actions. Designate supports a variety of DNS servers including Bind9 and PowerDNS 4.

CHAPTER

ONE

CONTENTS

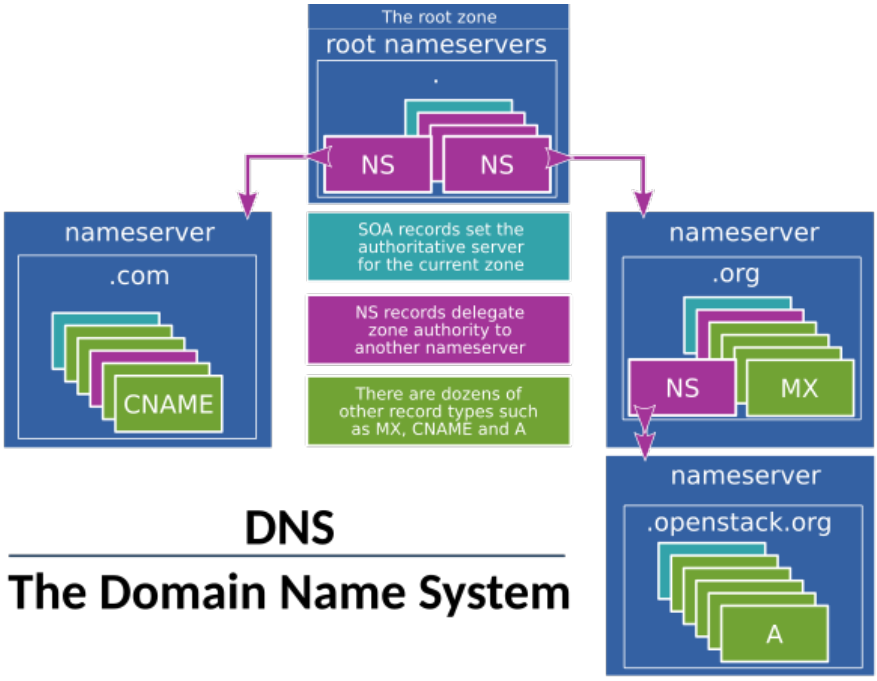
1.1 Introduction to Designate

Designate is an Open Source DNS-as-a-Service implementation and a part of the OpenStack ecosystem of services for running clouds. In order to understand what Designate can do and how it works, its necessary to understand some of the basics of DNS.

1.1.1 What is DNS?

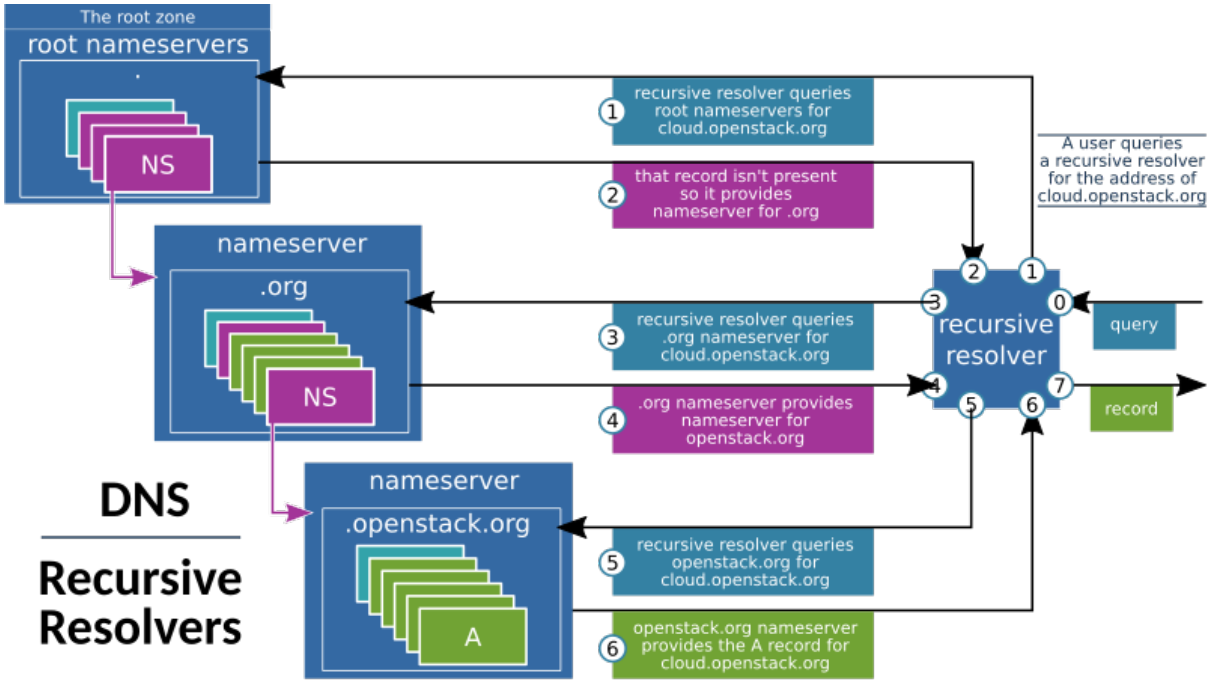
The Domain Name System (DNS) is a system for naming resources connected to a network, and works by storing various types of *record*, such as an IP address associated with a domain name. In practice, this is implemented by *authoritative name servers* which contain these records and *resolvers* which query name servers for records. Names are divided up into a hierarchy of zones, allowing different name servers to be responsible for separate groups of zones by delegating responsibility using records.

The root zone, which is simply ., is comprised entirely of records delegating various top level domains (TLDs) to other nameservers. The TLD name servers will contain records for domains within their TLD, such as the *.com* nameserver having an *example.com* record, as well as records that delegate zones to other nameservers, for example *openstack.org* might have their own nameserver so that they can then create *cloud.openstack.org*.



Resolvers are often formed in two parts: a *stub* resolver which is often merely a library on a users computer, and a *recursive resolver* that will perform queries against nameservers before returning the result to the user. When searching for a domain, the resolver will start at the end of the domain and work its way back to the beginning.

For example in the diagram below, when searching for cloud.openstack.org, it will start with the root nameserver ., which will reply with the location of the .org nameserver. The resolver can then contact the .org nameserver to get the openstack.org nameserver and from there finally get the cloud.openstack.org record and return it to the user.



In order to make this more efficient, the results are cached on the resolver, so after the first user has requested cloud.openstack.org, the resolver can return the cached result for subsequent requests.

Further reading on DNS and how it works is available here:

- https://en.wikipedia.org/wiki/Domain_Name_System

While the system itself is defined via RFCs such as this:

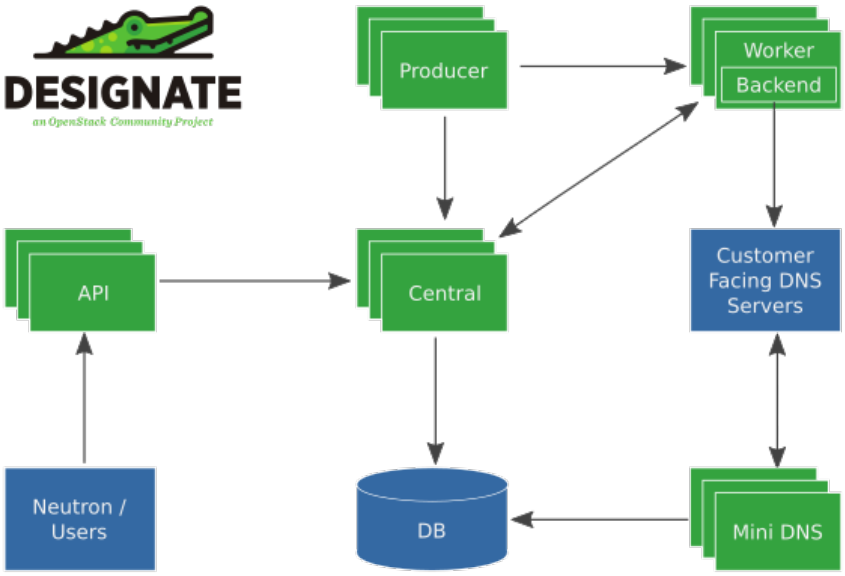
- https://tools.ietf.org/html/rfc1034

1.1.2 Introducing Designate

Designate is an OpenStack service that allows users and operators to manage DNS records, names and zones via a REST API and can configure existing DNS name servers to contain those records. Designate can also be configured by an operator to integrate with both the OpenStack Network Service (Neutron) and the Compute Service (Nova) so that records are automatically created when floating IPs and compute instances are created respectively, and uses the OpenStack Identity Service (Keystone) for user management. Since there are a multitude of software implementations of the DNS name server, Designate has a pluggable backend that can be configured to manage many of them, most notably BIND9 and PowerDNS.

1.1.3 Designate Architecture

Designate is comprised of several different services: the API, Producer, Central, Worker and Mini DNS. It uses an oslo.db compatible database to store state and data, and an oslo.messaging compatible message queue to facilitate communication between services. Multiple copies of all Designate services can be run in tandem to facilitate high availability deployments, with the API process often sitting behind load balancers.



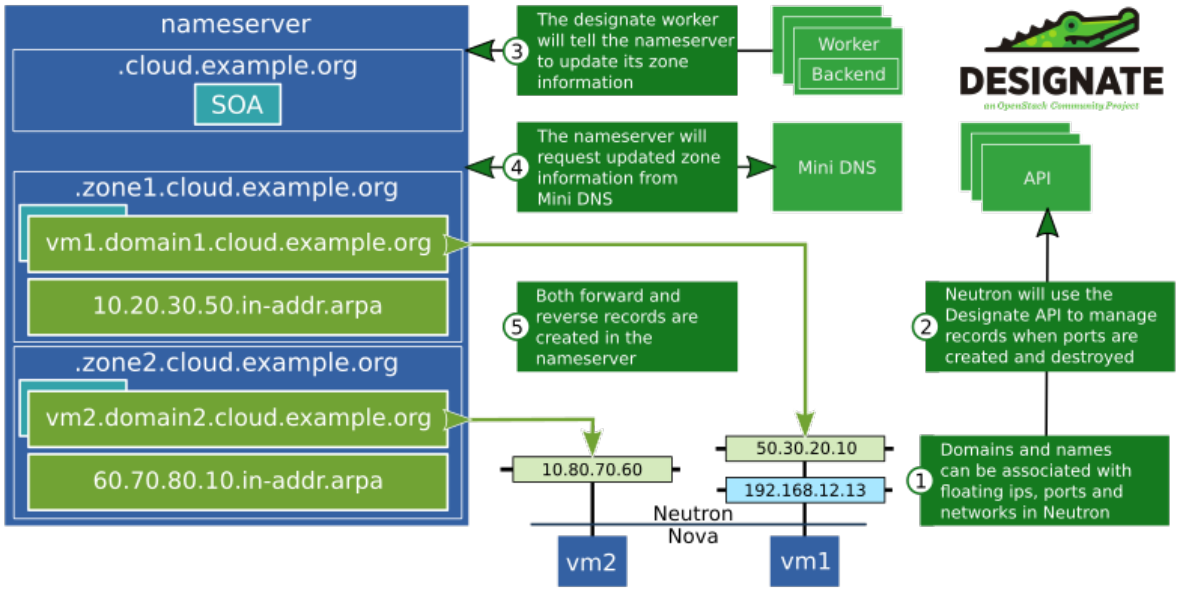
Neutron and other users of Designate only need to be able to access the API server, while administrators should ensure the DNS Nameservers to be configured are able to access Mini DNS from which to request updates.

Below we can see a common deployment scenario:

A user has created two zones in Designate: *zone1.cloud.openstack.org* and *zone2.cloud.openstack.org*. This will result in two new zones being created on the Designate-managed nameserver with SOA records.

The user then created two networks in Neutron: one private network with *zone1.cloud.openstack.org* assigned to it, and one public network with *zone2.cloud.openstack.org*.

They have then created virtual machine *vm1* in Nova, connected to the private network in Neutron and attached to a floating IP, and the virtual machine *vm2* attached directly to the public network. Each of these actions triggers a chain of events that will cause Neutron to request Designate create records on behalf of the user, with the end result being that records are created in the authoritative nameserver mapping the vm names to domains along with PTR records to allow reverse lookups.



More information about configuring Neutron to work with Designate can be found in the Neutron documentation at https://docs.openstack.org/neutron/latest/admin/config-dns-int-ext-serv.html

1.1.4 Using Designate

Designate provides a REST API and that is commonly used by one of three methods. The most common is to use the OpenStack client, a python command-line tool with commands for interacting with OpenStack services. The documentation for the OpenStack client is available at https://docs.openstack.org/python-openstackclient/. The designate plugin for the OpenStack client needs to be installed as well:

```
pip install python-openstackclient
pip install python-designateclient
```

Another popular way to use Designate is via the OpenStack Dashboard, Horizon. Administrators will need to add the Designate Horizon plugin to the dashboard in order to enable Designate features.

Finally, for python developers the aforementioned Designate plugin for the OpenStack client which can be used as a python library. Other languages may have bindings available from one of the third party SDKs for OpenStack.

1.2 Installing OpenStack DNS as a Service

1.2.1 Manual Designate installation

This chapter assumes a working setup of OpenStack following the OpenStack Installation Tutorial.

DNS service overview

The DNS service provides DNS Zone and RecordSet management for OpenStack clouds. The DNS Service includes a REST API, a command-line client, and a Horizon Dashboard plugin.

The DNS service consists of the following components:

`openstack` command-line client plugin
A plugin for the OpenStack Client CLI that communicates with the REST API

`designate-api` component
An OpenStack-native REST API that processes API requests by sending them to the `designate-central` over Remote Procedure Call (RPC).

`designate-central` component
Orchestrates the creation, deletion and update of Zones and RecordSets.

`designate-producer` component
Orchestrates periodic tasks that are run by designate.

`designate-worker` component
Is a generic task runner, that runs both zone create / update and deletes, and periodic tasks, from `designate-producer`

`designate-mdns` component
A small DNS Server that is responsible for pushing DNS Zone information to the customer facing DNS Servers. Can also pull in DNS information about DNS Zones hosted outside of the Designate infrastructure

`Customer Facing DNS Servers`
Serves DNS requests to end users. They are orchestreated by the `designate-worker`, and the supported list is maintained *here*.

Install and configure

This section describes how to install and configure the DNS service, code-named designate, on the controller node.

This section assumes that you already have a working OpenStack environment with at least the Identity service installed.

Note that installation and configuration vary by distribution.

Install and configure for openSUSE and SUSE Linux Enterprise

This section describes how to install and configure the DNS service for openSUSE Leap 42.2 and SUSE Linux Enterprise Server 12 SP2.

Prerequisites

Before you install and configure the DNS service, you must create service credentials and API endpoints.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc
```

2. To create the service credentials, complete these steps:
   - Create the `designate` user:

```
$ openstack user create --domain default --password-prompt designate
```

- Add the `admin` role to the `designate` user:

```
$ openstack role add --project service --user designate admin
```

- Create the designate service entities:

```
$ openstack service create --name designate --description "DNS" dns
```

3. Create the DNS service API endpoint:

```
$ openstack endpoint create --region RegionOne \
  dns public http://controller:9001/
```

Install and configure components

Note

Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# zypper install openstack-designate\*
```

2. Create a `designate` database that is accessible by the `designate` user. Replace `DESIGNATE_DBPASS` with a suitable password:

```
# mysql
MariaDB [(none)]> CREATE DATABASE designate CHARACTER SET utf8 COLLATE␣
↪utf8_general_ci;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@
↪'localhost' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@'%' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
```

3. Install the BIND packages:

```
# zypper install bind bind-utils
```

4. Create an RNDC Key:

```
# rndc-confgen -a -k designate -c /etc/designate/rndc.key -r /dev/urandom
```

5. Add the following options in the `/etc/named.conf` file:

```
...
include "/etc/designate/rndc.key";

options {
    ...
    allow-new-zones yes;
    request-ixfr no;
    listen-on port 53 { 127.0.0.1; };
    recursion no;
    allow-query { 127.0.0.1; };
};

controls {
  inet 127.0.0.1 port 953
    allow { 127.0.0.1; } keys { "designate"; };
};
```

6. Start the DNS service and configure it to start when the system boots:

```
# systemctl enable named

# systemctl start named
```

7. Edit the `/etc/designate/designate.conf` file and complete the following actions:

   - In the `[service:api]` section, configure `auth_strategy`:

```
[service:api]
listen = 0.0.0.0:9001
auth_strategy = keystone
enable_api_v2 = True
enable_api_admin = True
enable_host_header = True
enabled_extensions_admin = quotas, reports
```

   - In the `[keystone_authtoken]` section, configure the following options:

```
[keystone_authtoken]
auth_type = password
username = designate
password = DESIGNATE_PASS
project_name = service
project_domain_name = Default
user_domain_name = Default
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
```

     Replace `DESIGNATE_PASS` with the password you chose for the `designate` user in the Identity service.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in RabbitMQ.

- In the `[storage:sqlalchemy]` section, configure database access:

```
[storage:sqlalchemy]
connection = mysql+pymysql://designate:DESIGNATE_DBPASS@controller/
↪designate
```

Replace `DESIGNATE_DBPASS` with the password you chose for the `designate` database.

- Populate the designate database

```
# su -s /bin/sh -c "designate-manage database sync" designate
```

8. Start the designate central and API services and configure them to start when the system boots:

```
# systemctl start openstack-designate-central openstack-designate-api

# systemctl enable openstack-designate-central openstack-designate-api
```

9. Create a pools.yaml file in `/etc/designate/pools.yaml` with the following contents:

```
- name: default
  # The name is immutable. There will be no option to change the name␣
↪after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default Pool

  attributes: {}

  # List out the NS records for zones hosted within this pool
  # This should be a record that is created outside of designate, that
  # points to the public IP of the controller node.
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1

  # List out the nameservers for this pool. These are the actual BIND␣
↪servers.
  # We use these to verify changes have propagated to all nameservers.
  nameservers:
    - host: 127.0.0.1
      port: 53

  # List out the targets for this pool. For BIND there will be one
  # entry for each BIND server, as we have to run rndc command on each␣
```

(continues on next page)

(continued from previous page)

```
↪server
  targets:
    - type: bind9
      description: BIND9 Server 1

      # List out the designate-mdns servers from which BIND servers␣
↪should
      # request zone transfers (AXFRs) from.
      # This should be the IP of the controller node.
      # If you have multiple controllers you can add multiple masters
      # by running designate-mdns on them, and adding them here.
      masters:
        - host: 127.0.0.1
          port: 5354

      # BIND Configuration options
      options:
        host: 127.0.0.1
        port: 53
        rndc_host: 127.0.0.1
        rndc_port: 953
        rndc_key_file: /etc/designate/rndc.key
```

10. Update the pools:

```
# su -s /bin/sh -c "designate-manage pool update" designate
```

11. Start the designate and mDNS services and configure them to start when the system boots:

```
# systemctl start openstack-designate-worker openstack-designate-producer␣
↪openstack-designate-mdns

# systemctl enable openstack-designate-worker openstack-designate-
↪producer openstack-designate-mdns
```

Install and configure for Red Hat Enterprise Linux and CentOS

This section describes how to install and configure the DNS service for Red Hat Enterprise Linux 7 and CentOS 7.

Prerequisites

Before you install and configure the DNS service, you must create service credentials and API endpoints.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc
```

2. To create the service credentials, complete these steps:

   - Create the `designate` user:

```
$ openstack user create --domain default --password-prompt designate
```

- Add the `admin` role to the `designate` user:

```
$ openstack role add --project service --user designate admin
```

- Create the designate service entities:

```
$ openstack service create --name designate --description "DNS" dns
```

3. Create the DNS service API endpoint:

```
$ openstack endpoint create --region RegionOne \
  dns public http://controller:9001/
```

Install and configure components

Note

Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# dnf install openstack-designate\*
```

2. Create a `designate` database that is accessible by the `designate` user. Replace `DESIGNATE_DBPASS` with a suitable password:

```
# mysql
MariaDB [(none)]> CREATE DATABASE designate CHARACTER SET utf8 COLLATE␣
↪utf8_general_ci;
MariaDB [(none)]> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@
↪'localhost' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@'%' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
```

3. Install the BIND packages:

```
# dnf install bind bind-utils
```

4. Create an RNDC Key:

```
# rndc-confgen -a -k designate -c /etc/designate/rndc.key -r /dev/urandom
```

5. Add the following options in the `/etc/named.conf` file:

```
...
include "/etc/designate/rndc.key";

options {
    ...
    allow-new-zones yes;
    request-ixfr no;
    listen-on port 53 { 127.0.0.1; };
    recursion no;
    allow-query { 127.0.0.1; };
};

controls {
  inet 127.0.0.1 port 953
    allow { 127.0.0.1; } keys { "designate"; };
};
```

6. Start the DNS service and configure it to start when the system boots:

```
# systemctl enable named

# systemctl start named
```

7. Edit the `/etc/designate/designate.conf` file and complete the following actions:

   - In the `[service:api]` section, configure `auth_strategy`:

```
[service:api]
listen = 0.0.0.0:9001
auth_strategy = keystone
enable_api_v2 = True
enable_api_admin = True
enable_host_header = True
enabled_extensions_admin = quotas, reports
```

   - In the `[keystone_authtoken]` section, configure the following options:

```
[keystone_authtoken]
auth_type = password
username = designate
password = DESIGNATE_PASS
project_name = service
project_domain_name = Default
user_domain_name = Default
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
```

     Replace `DESIGNATE_PASS` with the password you chose for the `designate` user in the Identity service.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

Replace `RABBIT_PASS` with the password you chose for the `openstack` account in RabbitMQ.

- In the `[storage:sqlalchemy]` section, configure database access:

```
[storage:sqlalchemy]
connection = mysql+pymysql://designate:DESIGNATE_DBPASS@controller/
↪designate
```

Replace `DESIGNATE_DBPASS` with the password you chose for the `designate` database.

- Populate the designate database

```
# su -s /bin/sh -c "designate-manage database sync" designate
```

8. Start the designate central and API services and configure them to start when the system boots:

```
# systemctl start designate-central designate-api

# systemctl enable designate-central designate-api
```

9. Create a pools.yaml file in `/etc/designate/pools.yaml` with the following contents:

```
- name: default
  # The name is immutable. There will be no option to change the name␣
↪after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default Pool

  attributes: {}

  # List out the NS records for zones hosted within this pool
  # This should be a record that is created outside of designate, that
  # points to the public IP of the controller node.
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1

  # List out the nameservers for this pool. These are the actual BIND␣
↪servers.
  # We use these to verify changes have propagated to all nameservers.
  nameservers:
    - host: 127.0.0.1
      port: 53

  # List out the targets for this pool. For BIND there will be one
  # entry for each BIND server, as we have to run rndc command on each␣
```

(continues on next page)

(continued from previous page)

```
↪server
  targets:
    - type: bind9
      description: BIND9 Server 1

      # List out the designate-mdns servers from which BIND servers␣
↪should
      # request zone transfers (AXFRs) from.
      # This should be the IP of the controller node.
      # If you have multiple controllers you can add multiple masters
      # by running designate-mdns on them, and adding them here.
      masters:
        - host: 127.0.0.1
          port: 5354

      # BIND Configuration options
      options:
        host: 127.0.0.1
        port: 53
        rndc_host: 127.0.0.1
        rndc_port: 953
        rndc_key_file: /etc/designate/rndc.key
```

10. Update the pools:

```
# su -s /bin/sh -c "designate-manage pool update" designate
```

11. Start the designate and mDNS services and configure them to start when the system boots:

```
# systemctl start designate-worker designate-producer designate-mdns

# systemctl enable designate-worker designate-producer designate-mdns
```

Install and configure for Ubuntu

This section describes how to install and configure the DNS service for Ubuntu 16.04 (LTS).

Prerequisites

Before you install and configure the DNS service, you must create service credentials and API endpoints.

1. Source the `admin` credentials to gain access to admin-only CLI commands:

```
$ source admin-openrc
```

2. To create the service credentials, complete these steps:

   - Create the `designate` user:

```
$ openstack user create --domain default --password-prompt designate
```

   - Add the `admin` role to the `designate` user:

```
$ openstack role add --project service --user designate admin
```

- Create the designate service entities:

```
$ openstack service create --name designate --description "DNS" dns
```

3. Create the DNS service API endpoint:

```
$ openstack endpoint create --region RegionOne \
  dns public http://controller:9001/
```

Install and configure components

Note

Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (`...`) in the configuration snippets indicates potential default configuration options that you should retain.

1. Install the packages:

```
# apt-get install designate
```

2. Create a `designate` database that is accessible by the `designate` user. Replace `DESIGNATE_DBPASS` with a suitable password:

```
# mysql
mysql> CREATE DATABASE designate CHARACTER SET utf8 COLLATE utf8_general_
↪ci;
mysql> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@'localhost' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON designate.* TO 'designate'@'%' \
IDENTIFIED BY 'DESIGNATE_DBPASS';
```

3. Install the BIND9 packages:

```
# apt-get install bind9 bind9utils bind9-doc
```

4. Create an RNDC Key:

```
# rndc-confgen -a -k designate -c /etc/designate/rndc.key -r /dev/urandom
```

5. Add the following options in the `/etc/bind/named.conf.options` file:

```
...
include "/etc/designate/rndc.key";

options {
    ...
    allow-new-zones yes;
```

(continues on next page)

(continued from previous page)

```
    request-ixfr no;
    listen-on port 53 { 127.0.0.1; };
    recursion no;
    allow-query { 127.0.0.1; };
};

controls {
  inet 127.0.0.1 port 953
    allow { 127.0.0.1; } keys { "designate"; };
};
```

6. Restart the DNS service:

```
# systemctl restart bind9.service
```

7. Edit the `/etc/designate/designate.conf` file and complete the following actions:

   - In the `[service:api]` section, configure `auth_strategy`:

```
[service:api]
listen = 0.0.0.0:9001
auth_strategy = keystone
enable_api_v2 = True
enable_api_admin = True
enable_host_header = True
enabled_extensions_admin = quotas, reports
```

   - In the `[keystone_authtoken]` section, configure the following options:

```
[keystone_authtoken]
auth_type = password
username = designate
password = DESIGNATE_PASS
project_name = service
project_domain_name = Default
user_domain_name = Default
www_authenticate_uri = http://controller:5000/
auth_url = http://controller:5000/
memcached_servers = controller:11211
```

     Replace `DESIGNATE_PASS` with the password you chose for the `designate` user in the Identity service.

   - In the `[DEFAULT]` section, configure `RabbitMQ` message queue access:

```
[DEFAULT]
# ...
transport_url = rabbit://openstack:RABBIT_PASS@controller:5672/
```

     Replace `RABBIT_PASS` with the password you chose for the `openstack` account in RabbitMQ.

   - In the `[storage:sqlalchemy]` section, configure database access:

```
[storage:sqlalchemy]
connection = mysql+pymysql://designate:DESIGNATE_DBPASS@controller/
↪designate
```

Replace `DESIGNATE_DBPASS` with the password you chose for the `designate` database.

- Populate the designate database

```
# su -s /bin/sh -c "designate-manage database sync" designate
```

8. Start the designate central and API services and configure them to start when the system boots:

```
# systemctl start designate-central designate-api

# systemctl enable designate-central designate-api
```

9. Create a pools.yaml file in `/etc/designate/pools.yaml` with the following contents:

```
- name: default
  # The name is immutable. There will be no option to change the name␣
↪after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default Pool

  attributes: {}

  # List out the NS records for zones hosted within this pool
  # This should be a record that is created outside of designate, that
  # points to the public IP of the controller node.
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1

  # List out the nameservers for this pool. These are the actual BIND␣
↪servers.
  # We use these to verify changes have propagated to all nameservers.
  nameservers:
    - host: 127.0.0.1
      port: 53

  # List out the targets for this pool. For BIND there will be one
  # entry for each BIND server, as we have to run rndc command on each␣
↪server
  targets:
    - type: bind9
      description: BIND9 Server 1

      # List out the designate-mdns servers from which BIND servers␣
↪should
      # request zone transfers (AXFRs) from.
```

(continues on next page)

(continued from previous page)

```
    # This should be the IP of the controller node.
    # If you have multiple controllers you can add multiple masters
    # by running designate-mdns on them, and adding them here.
    masters:
      - host: 127.0.0.1
        port: 5354

    # BIND Configuration options
    options:
      host: 127.0.0.1
      port: 53
      rndc_host: 127.0.0.1
      rndc_port: 953
      rndc_key_file: /etc/designate/rndc.key
```

10. Update the pools:

```
# su -s /bin/sh -c "designate-manage pool update" designate
```

11. Install Designate Worker, producer and mini-dns

```
# apt install designate-worker designate-producer designate-mdns
```

12. Start the designate and mDNS services and configure them to start when the system boots:

```
# systemctl start designate-worker designate-producer designate-mdns

# systemctl enable designate-worker designate-producer designate-mdns
```

Verify operation

Verify operation of the DNS service.

Note

Perform these commands on the controller node.

1. Source the `admin` tenant credentials:

```
$ . admin-openrc
```

2. List service components to verify successful launch and registration of each process:

```
$ ps -aux | grep designate

../usr/bin/python /usr/bin/designate-mdns --config-file /etc/designate/
↪designate.conf
../usr/bin/python /usr/bin/designate-central --config-file /etc/designate/
↪designate.conf
../usr/bin/python /usr/bin/designate-api --config-file /etc/designate/
```

(continues on next page)

(continued from previous page)

```
↪designate.conf
../usr/bin/python /usr/bin/designate-worker --config-file /etc/designate/
↪designate.conf
../usr/bin/python /usr/bin/designate-producer --config-file /etc/
↪designate/designate.conf

$ openstack dns service list
+--------------------------------------+-------------------------+-------
↪-------+--------+-------+--------------+
| id                                   | hostname                |␣
↪service_name | status | stats | capabilities |
+--------------------------------------+-------------------------+-------
↪-------+--------+-------+--------------+
| 918a8f6e-9e7e-453e-8583-cbefa7ae7f8f | vagrant-ubuntu-trusty-64 |␣
↪central      | UP     | -     | -            |
| 982f78d5-525a-4c36-af26-a09aa39de5d7 | vagrant-ubuntu-trusty-64 | api  ␣
↪       | UP     | -     | -            |
| eda2dc16-ad27-4ee1-b091-bb75b6ceaffe | vagrant-ubuntu-trusty-64 | mdns ␣
↪       | UP     | -     | -            |
| 00c5c372-e630-49b1-a6b6-17e3fa4544ea | vagrant-ubuntu-trusty-64 |␣
↪worker       | UP     | -     | -            |
| 8cdaf2e9-accd-4665-8e9e-be26f1ccfe4a | vagrant-ubuntu-trusty-64 |␣
↪producer     | UP     | -     | -            |
+--------------------------------------+-------------------------+-------
↪-------+--------+-------+--------------+
```

Note

This output should indicate at least one of each of the `central`, `api`, `producer`, `mdns` and `worker` components on the controller node.

This output may differ slightly depending on the distribution.

Create a Zone

In environments that include the DNS service, you can create a DNS Zone.

1. Source the `demo` credentials to perform the following steps as a non-administrative project:

```
$ . demo-openrc
```

2. Create a DNS Zone called `example.com.`:

```
$ openstack zone create --email dnsmaster@example.com example.com.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| attributes     | {}                                   |
| created_at     | 2016-07-13T14:54:16.000000           |
```

(continues on next page)

(continued from previous page)

```
| description    | None                                 |
| email          | dnsmaster@example.com                |
| id             | 14093115-0f0f-497a-ac69-42235e46c26f |
| masters        |                                      |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 656bc359067844fba6005d400f19df76     |
| serial         | 1468421656                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+
```

3. After a short time, verify successful creation of the DNS Zone:

```
$ openstack zone list
+--------------------------------------+-------------+---------+---------
↪---+--------+--------+
| id                                   | name        | type    |
↪serial | status | action |
+--------------------------------------+-------------+---------+---------
↪---+--------+--------+
| 14093115-0f0f-497a-ac69-42235e46c26f | example.com. | PRIMARY |
↪1468421656 | ACTIVE | NONE   |
+--------------------------------------+-------------+---------+---------
↪---+--------+--------+
```

4. You can now create RecordSets in this DNS Zone:

```
$ openstack recordset create --record '10.0.0.1' --type A example.com. www
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2016-07-13T14:59:32.000000           |
| description | None                                 |
| id          | 07e6f5af-783e-481f-b8df-5972a6174c94 |
| name        | www.example.com.                     |
| project_id  | 656bc359067844fba6005d400f19df76     |
| records     | 10.0.0.1                             |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 14093115-0f0f-497a-ac69-42235e46c26f |
| zone_name   | example.com.                         |
```

(continues on next page)

(continued from previous page)

```
+-------------+--------------------------------------+
```

5. Delete the DNS Zone:

```
$ openstack zone delete example.com.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | DELETE                               |
| attributes     |                                      |
| created_at     | 2017-07-12T03:26:25.000000           |
| description    | None                                 |
| email          | dnsmaster@example.com                |
| id             | 4a21a893-2c58-4797-82ed-19fcef7c418d |
| masters        |                                      |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | d53f80b5a22b4962a176935eea23f9c4     |
| serial         | 1499830029                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | 2017-07-12T03:27:25.000000           |
| version        | 4                                    |
+----------------+--------------------------------------+
```

Next steps

Your OpenStack environment now includes the designate service.

To add additional services, see the OpenStack install guide.

To learn more about the designate service, read the *Designate developer documentation*.

1.2.2 Quickstart with Kolla

Following the Designate in Kolla to quickly install and setup Designate.

1.3 Developer documentation

In this section, you will find documentation relevant to developing Designate.

Contents:

1.3.1 Getting Involved

How to install DNS with DevStack

The Designate source code contains a DevStack plugin that allows to deploy an OpenStack installation with the DNS service enabled.

Instructions

Note

If you want to use local sources for development then you should consider using the contrib/vagrant folder in the repository.

1. Get a clean Ubuntu 20.04 VM (see the DevStack installation instructions for more details). DevStack takes over. Dont use your desktop!
2. Clone DevStack inside the VM

```
$ git clone https://opendev.org/openstack/devstack.git
```

3. Move to `devstack` directory

```
$ cd devstack
```

4. Create a *local.conf* config file

```
[[local|localrc]]
# General DevStack Config
# =======================
ADMIN_PASSWORD=password
MYSQL_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password

# IP Address for services to bind to (Should match IP from Vagrantfile)
SERVICE_HOST=192.168.27.100
HOST_IP=$SERVICE_HOST

# Logging
#LOGFILE=/opt/stack/logs/stack.sh.log
VERBOSE=True
LOG_COLOR=True

# Test a Gerrit Review
# DESIGNATE_REPO=https://review.opendev.org/openstack/designate
# DESIGNATE_BRANCH=refs/changes/41/765541/1

# Test a particular branch
# DESIGNATE_REPO=https://opendev.org/openstack/designate.git
# DESIGNATE_BRANCH=stable/stein

# Disable all services except core ones
disable_all_services
enable_service rabbit mysql key

# Enable designate
```

(continues on next page)

(continued from previous page)

```
enable_plugin designate https://opendev.org/openstack/designate

# Designate Devstack Config
# =========================
# Enable core Designate services
enable_service designate,designate-central,designate-api,designate-worker,
↪designate-producer,designate-mdns

# Optional Designate services
#enable_service designate-agent
#enable_service designate-sink

# Backend Driver (e.g. powerdns, bind9. See designate.backend section of
#                 setup.cfg)
#DESIGNATE_BACKEND_DRIVER=bind9

# Agent Backend Driver (Used only when DESIGNATE_BACKEND_DRIVER=agent)
#DESIGNATE_AGENT_BACKEND_DRIVER=fake

# Pool Manager Cache Driver (e.g. noop, memcache, sqlalchemy. See
#                            designate.backend section of setup.cfg)
#DESIGNATE_POOL_MANAGER_CACHE_DRIVER=memcache

# mDNS Service DNS Port Number
#DESIGNATE_SERVICE_PORT_MDNS=5354

# Designate Backend Config
# ========================
# DynECT Backend
# NOTEs:
# - DynECT requires DESIGNATE_SERVICE_PORT_MDNS is set to "53"
# - DESIGNATE_DYNECT_MASTERS must be a Publicly reachable IP, pointed to␣
↪mDNS
#DESIGNATE_DYNECT_CUSTOMER=
#DESIGNATE_DYNECT_USERNAME=
#DESIGNATE_DYNECT_PASSWORD=
#DESIGNATE_DYNECT_NAMESERVERS=ns1.p13.dynect.net,ns2.p13.dynect.net,ns3.
↪p13.dynect.net,ns4.p13.dynect.net
#DESIGNATE_DYNECT_MASTERS=

# Akamai Backend
#DESIGNATE_AKAMAI_USERNAME=
#DESIGNATE_AKAMAI_PASSWORD=
#DESIGNATE_AKAMAI_NAMESERVERS=a5-64.akam.net,a11-65.akam.net,a13-66.akam.
↪net,a14-64.akam.net,a20-65.akam.net,a22-66.akam.net
#DESIGNATE_AKAMAI_MASTERS=

# Designate D2D Backend
# NOTEs:
```

(continues on next page)

(continued from previous page)

```
# - DESIGNATE_D2D_ALSO_NOTIFIES needs to be set to the source mdns␣
↪ip:port in
#   order for designate to receive the proper NOTIFY
# - DESIGNATE_D2D_* credentials should be setup either to the source␣
↪keystone
#   or the destination
#DESIGNATE_D2D_MASTERS=
#DESIGNATE_D2D_ALSO_NOTIFIES=
#DESIGNATE_D2D_NAMESERVERS=

# Authentication options
#DESIGNATE_D2D_KS_VERSION=3

#DESIGNATE_D2D_AUTH_URL=
#DESIGNATE_D2D_USERNAME=
#DESIGNATE_D2D_PASSWORD=

# Keystone V2
#DESIGNATE_D2D_TENANT_NAME=${DESIGNATE_D2D_TENANT_NAME:-}
#DESIGNATE_D2D_TENANT_NAME=${DESIGNATE_D2D_TENANT_ID:-}

# Keystone V3
#DESIGNATE_D2D_PROJECT_NAME=
#DESIGNATE_D2D_PROJECT_DOMAIN_NAME=
#DESIGNATE_D2D_USER_DOMAIN_NAME=


# Designate Misc Config
# =====================

# Enable a Notification Driver (e.g. for Ceiliometer)
#DESIGNATE_NOTIFICATION_DRIVER=messaging

# Set Notification topics
#DESIGNATE_NOTIFICATION_TOPICS=notifications

# Set coordination service URL (e.g. kazoo://localhost/)
#DESIGNATE_COORDINATION_URL=


# Other Devstack Config
# =====================
# Optional TLS Proxy
#enable_service tls-proxy

# Optional Tempest (Recommended)
enable_service tempest

# Optional Rally
```

(continues on next page)

(continued from previous page)

```
#enable_plugin rally https://opendev.org/openstack/rally.git master

# Optional Horizon
#enable_service horizon

# Optional Glance
#enable_service g-api

# Optional Nova
#enable_service n-api n-cpu n-net n-cond n-sch n-novnc

# Optional Neutron
#disable_service n-net
#enable_service q-svc q-agt q-dhcp q-l3 q-meta
```

5. Run DevStack

```
$ ./stack.sh
```

6. See the status of all Designate processes

```
$ sudo systemctl status devstack@designate-*.service
```

See the Using Systemd in DevStack home page for more options.

7. Querying Logs

```
$ sudo journalctl -f --unit devstack@designate-*.service
```

See the Querying Logs home page for more options.

8. Load credentials into the shell

```
$ export OS_CLOUD=devstack-admin   # For the admin user, admin project
$ export OS_CLOUD=devstack         # For the demo user, demo project
```

9. Try out the openstack client

```
$ openstack zone create --email admin@example.net example.net.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| attributes     |                                      |
| created_at     | 2017-11-15T04:48:40.000000           |
| description    | None                                 |
| email          | admin@example.net                    |
| id             | f34f835b-9acc-4930-b6dd-d045c15da78a |
| masters        |                                      |
| name           | example.net.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 9d0beaef253a4e14bd7025dc30c24f98     |
```

(continues on next page)

(continued from previous page)

```
| serial         | 1510721320                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+

$ openstack recordset create --record '127.0.0.1'  --type A example.net.␣
↪www
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2017-11-15T04:51:27.000000           |
| description | None                                 |
| id          | 7861e600-8d9e-4e13-9ea2-9038a2719b41 |
| name        | www.example.net.                     |
| project_id  | 9d0beaef253a4e14bd7025dc30c24f98     |
| records     | 127.0.0.1                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | f34f835b-9acc-4930-b6dd-d045c15da78a |
| zone_name   | example.net.                         |
+-------------+--------------------------------------+

$ openstack recordset list f34f835b-9acc-4930-b6dd-d045c15da78a
+--------------------------------------+------------------+------+--------
↪-------------------------------------------------------------+--------+-
↪-------+
| id                                   | name             | type |␣
↪records                                                     |␣
↪status | action |
+--------------------------------------+------------------+------+--------
↪-------------------------------------------------------------+--------+-
↪-------+
| d0630d94-94d8-43fc-93e8-973fbec7531e | example.net.     | SOA  | ns1.
↪devstack.org. admin.example.net. 1510721487 3510 600 86400 3600 |␣
↪ACTIVE | NONE   |
| 31a313dc-c322-4dc0-ba53-79c039d7f09f | example.net.     | NS   | ns1.
↪devstack.org.                                               |␣
↪ACTIVE | NONE   |
| 7861e600-8d9e-4e13-9ea2-9038a2719b41 | www.example.net. | A    | 127.0.
↪0.1                                                         | ACTIVE␣
↪| NONE   |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+------------------+------+--------
↪-------------------------------------------------------------+--------+-
↪-------+

$ openstack recordset show f34f835b-9acc-4930-b6dd-d045c15da78a 7861e600-
↪8d9e-4e13-9ea2-9038a2719b41
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | NONE                                 |
| created_at  | 2017-11-15T04:51:27.000000           |
| description | None                                 |
| id          | 7861e600-8d9e-4e13-9ea2-9038a2719b41 |
| name        | www.example.net.                     |
| project_id  | 9d0beaef253a4e14bd7025dc30c24f98     |
| records     | 127.0.0.1                            |
| status      | ACTIVE                               |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | f34f835b-9acc-4930-b6dd-d045c15da78a |
| zone_name   | example.net.                         |
+-------------+--------------------------------------+
```

10. Verify that the `recordset` is in DNS

```
$ dig www.example.net @${SERVICE_HOST}

; <<>> DiG 9.16.1-Ubuntu <<>> www.example.net @192.168.27.100
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34315
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: f10292dba9100bbf010000005f749e3586096307a693d0fe (good)
;; QUESTION SECTION:
;www.example.net.               IN      A

;; ANSWER SECTION:
www.example.net.        3600    IN      A       127.0.0.1

;; Query time: 0 msec
;; SERVER: 192.168.27.100#53(192.168.27.100)
;; WHEN: Wed Sep 30 15:03:17 UTC 2020
;; MSG SIZE  rcvd: 88
```

where `SERVICE_HOST` is the IP address used in `local.conf`.

#openstack-dns IRC channel

There is an active IRC channel at irc://oftc.net/#openstack-dns, where many of the designate contributors can be found, as well as users from various organisations.

Contributing

For general information on contributing to OpenStack please see the contributor guide to get started. It covers all the basics that are common to all OpenStack projects: the accounts you need, the basics of interacting with our Gerrit review system, how we communicate as a community, etc.

We welcome fixes, extensions, documentation, pretty much anything that helps improve Designate, contributing is easy & follows the standard OpenStack Gerrit workflow, if youre looking for something to do, you could always checkout the blueprint & bug lists.

The designate git repo is available at https://opendev.org/openstack/designate, though all contributions should be done via the Gerrit review system.

Task Tracking

We track our tasks in Launchpad

https://bugs.launchpad.net/designate

If youre looking for some smaller, easier work item to pick up and get started on, search for the low-hanging-fruit tag.

Reporting a Bug

You found an issue and want to make sure we are aware of it? You can do so on Launchpad.

Development Environment and Developer Workflow

Assuming youve already got a working *Development Environment*, heres a quick summary:

Install the git-review package to make life easier, some distros have it as native package, otherwise use pip

```
pip install git-review
```

Branch, work, & submit:

```
# cut a new branch, tracking master
git checkout --track -b bug/id origin/master
# work work work
git add stuff
git commit
# rebase/squash to a single commit before submitting
git rebase -i
# submit
git-review
```

Coding Standards

Designate uses the OpenStack flake8 coding standards guidelines. These are stricter than pep8, and are run by gerrit on every commit.

You can use tox to check your code locally by running

```
# For just flake8 tests
tox -e flake8
# For tests + flake8
tox
```

Example DNS Names and IP Space

The IANA has allocated several special purpose domains and IP blocks for use as examples in code and documentation. Where possible, these domains and IP blocks should be preferred. There are some cases where it will not be possible to follow this guidance, for example, there is currently no reserved IDN domain name.

We prefer to use these names and IP blocks to avoid causing any unexpected collateral damage to the rightful owners of the non-reserved names and IP space. For example, publishing an email address in our codebase will more than likely be picked up by spammers, while published URLs etc using non-reserved names or IP space will likely trigger search indexers etc to begin crawling.

Reserved Domains

Reserved DNS domains are documented here: IANA Special Use Domain Names.

Several common reserved domains:

- example.com.
- example.net.
- example.org.

Reserved IP Space

Reserved IP space is documented here: IANA IPv4 Special Registry, and IANA IPv6 Special Registry.

Several common reserved IP blocks:

- 192.0.2.0/24
- 198.51.100.0/24
- 203.0.113.0/24
- 2001:db8::/32

Style Guide

Follow OpenStack Style Guidelines

File header

Start new files with the following. Replace where needed:

```
# Copyright <year> <company>
#
# Author: <name> <email addr>
#
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
# License for the specific language governing permissions and limitations
# under the License.

"""
<package.module>
~~~~~~~~~~~~~~~
<Describe what the module should do, especially interactions with
other components and caveats>

<Optional links>
`Specs: Refer to a spec document if relevant`_

`User documentation <FILL_THIS.html>`_ <Refer to files under doc/>
<This is useful to remind developers to keep the docs up to date>
"""
```

Example:

```
backend.impl_akamai
~~~~~~~~~~~~~~~~~~~
Akamai backend. Create and delete zones on Akamai. Blah Blah...

`Specs: Keystone Session <https://opendev.org/openstack/designate-specs/src/
↪branch/master/specs/kilo/switch-to-keystone-session.rst>`_

`User documentation <backend.html>`_
```

When updating a module, please ensure that the related user documentation is updated as well.

Docstrings

Use the Sphinx markup. Here is an example:

```
class MyClass(object):
    """<description>
```

(continues on next page)

(continued from previous page)

```
    mention a function :func:`foo` or a class :class:`Bar`
    """

    def function(self, foo):
        """<describe what the function does>
        :param foo: <description>
        :type foo: <type>
        :returns: <describe the returned value>
        :rtype: <returned type>
        :raises: <list raised exceptions>

        :Example:

        >>> a = b - c
        >>> <more Python code>

        .. note:: <add a note here>
        .. seealso:: <blah>
        .. warning:: <use sparingly>
        """
```

Logging

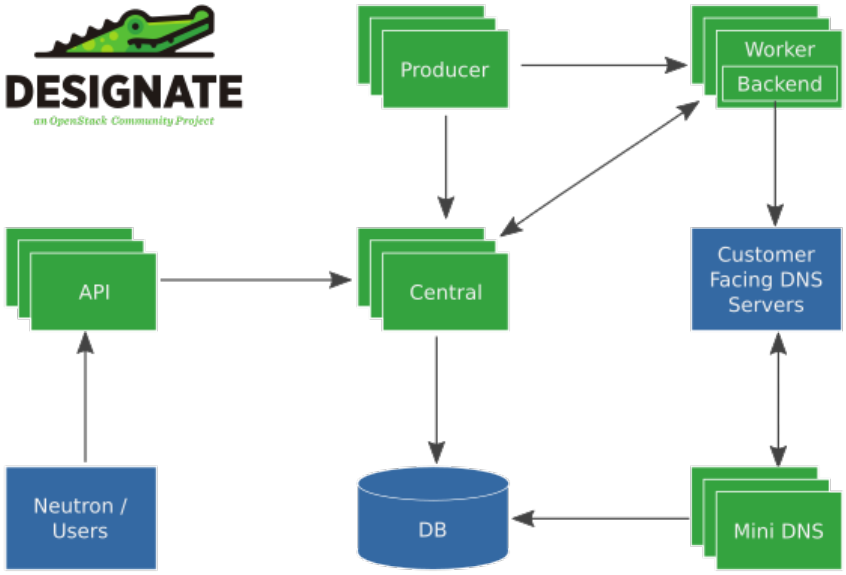
See https://docs.openstack.org/oslo.i18n/latest/user/guidelines.html

```
# Do not use "%" string formatting
# No localization for log messages
LOG.debug("... %s", variable)
# Use named interpolation when more than one replacement is done
LOG.info("... %(key)s ...", {'key': 'value', ...})
LOG.warning("... %(key)s", {'key': 'value'})
LOG.error("... %(key)s", {'key': 'value'})
LOG.critical("... %(key)s", {'key': 'value'})
```

1.3.2 Architecture

Designate provides multi-tenant DNS as a Service. Designate provides a REST API, applies business logic, persists DNS data to a database, and orchestrates the propagation of the DNS data to configured pools of DNS servers. For a more detailed breakdown of responsibilities and components, see the components below.

High Level Topology



Designate API

designate-api provides the standard OpenStack style REST API service, accepting HTTP requests, validating authentication tokens with Keystone and passing them to the *Designate Central* service over AMQP. Multiple versions of the API can be hosted, as well as API extensions, allowing for pluggable extensions to the core API.

Although designate-api is capable of handling HTTPS traffic, its typical to terminate HTTPS elsewhere, for example by placing nginx in front of designate-api or by letting the external facing load balancers terminate HTTPS.

Designate Central

designate-central is the service that handles RPC requests via the MQ, it coordinates the persistent storage of data and applies business logic to data from the API. Storage is provided via plugins, typically SQLAlchemy, although MongoDB or other storage drivers should be possible.

Designate MiniDNS

designate-mdns is the service that sends DNS NOTIFY and answers zone transfer (AXFR) requests. This allows Designate to integrate with any DNS server that supports these very standard methods of communicating. designate-mdns also encapsulates all other forms of DNS protocol that Designate performs. For example, sending SOA queries to check that a change is live.

Designate Worker

designate-worker is a service that manages state of the DNS servers Designate manages, and any other long-running or otherwise complicated piece of work. The worker reads configuration for DNS servers from the Designate database, which is populated via the pools.yaml file. These DNS server backends are loaded into the worker so it understands how to create, update, and delete zones and recordsets on each DNS server. The Worker is fully aware of DNS Server Pools, so a single worker process can manage many pools of DNS servers.

Designate Producer

designate-producer is a service that handles the invocation of long-running and potentially large jobs. Producer processes start work for an automatically assigned shard of the zones Designate manages. Shards are allocated based on the first three characters of the zone ID (a UUID field). The number of shards under management of a single producer process is equal to the total number of shards divided by the number of producer processes. This means the more producer processes are started, the less work is created at any one time.

The current implemented tasks in producer include emitting dns.zone.exists events for Ceilometer, purging deleted zones from database, polling secondary zones at their refresh intervals, generating delayed NOTIFY transactions, and invoking a periodic recovery of zones in an error state.

Designate Sink

designate-sink is an optional service which listens for event *Notifications*, such as compute.instance.create.end, handlers are available for Nova and Neutron. Notification events can then be used to trigger record creation & deletion.

The current sink implementations generate simple forward lookup A records, using a format specified in handler-nova configuration. Any field in the event notification can be used to generate a record.

DNS Backend

Backends are drivers for a particular DNS server. Designate supports multiple backend implementations, PowerDNS, BIND, NSD, DynECT, you are also free to implement your own backend to fit your needs, as well as extensions to provide extra functionality to complement existing backends.

Message Queue

Designate uses oslo.rpc for messaging between components, therefore it inherits a requirement for a supported messaging bus (such as RabbitMQ, Qpid or ZeroMQ). Typically this means a RabbitMQ setup is dedicated to Designate, but as only a single virtualhost is required for a normal installation, youre free to use other RabbitMQ instances as you see fit.

Database/Storage

Storage drivers are drivers for a particular SQL/NoSQL server. Designate needs a SQLAlchemy-supported storage engine for the persistent storage of data. The recommended driver is MySQL.

1.3.3 Guru Meditation Reports

A Guru Meditation Report (GMR) is generated by the Designate services when service processes receiving SIGUSR2 signal. The report is a general-purpose debug report for developers and system admins which contains the current state of a running Designate service process.

Structure of a GMR

Package
: Shows information about the package to which this process belongs, including version information

Threads
: Shows stack traces and thread ids for each of the threads within this process

Green Threads
Shows stack traces for each of the green threads within this process (green threads dont have thread ids)

Processes
Shows information about this process, including pid, ppid, uid and process state

Configuration
Lists all the configuration options currently accessible via the CONF object for the current process

Generate a GMR

A GMR can be generated by sending the USR2 signal to any Designate processes.

For example, suppose `designate-central` has pid `15097`, `kill -USR2 15097` will trigger a GMR.

If option `logdir` has been set in `designate.conf`, the GMR will be saved in the folder which `logdir` specified. Otherwise, the GMR will be printed to the stderr.

Reference

For more information about GMR, see GMR wiki.

GMR Example

```
========================================================================
====                        Guru Meditation                         ====
========================================================================
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||


========================================================================
====                            Package                             ====
========================================================================
product = OpenStack Designate
vendor = OpenStack Foundation
version = 2015.1
========================================================================
====                            Threads                             ====
========================================================================
------                  Thread #140098874533632                   ------

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/hub.py:346 in run
    `self.wait(sleep_time)`

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/poll.py:85 in wait
    `presult = self.do_poll(seconds)`

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/epolls.py:62 in do_poll
    `return self.poll.poll(seconds)`

========================================================================
====                         Green Threads                          ====
```

(continues on next page)

(continued from previous page)

```
==========================================================================
------                        Green Thread                        ------

/usr/local/lib/python2.7/dist-packages/eventlet/greenthread.py:214 in main
    `result = function(*args, **kwargs)`

/opt/stack/designate/designate/openstack/common/service.py:492 in run_service
    `done.wait()`

/usr/local/lib/python2.7/dist-packages/eventlet/event.py:121 in wait
    `return hubs.get_hub().switch()`

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/hub.py:294 in switch
    `return self.greenlet.switch()`

------                        Green Thread                        ------

/usr/local/lib/python2.7/dist-packages/eventlet/greenthread.py:214 in main
    `result = function(*args, **kwargs)`

/usr/local/lib/python2.7/dist-packages/oslo_utils/excutils.py:95 in inner_func
    `return infunc(*args, **kwargs)`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_executors/impl_
↪eventlet.py:96 in _executor_thread
    `incoming = self.listener.poll()`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/amqpdriver.
↪py:121 in poll
    `self.conn.consume(limit=1, timeout=timeout)`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/impl_rabbit.
↪py:867 in consume
    `six.next(it)`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/impl_rabbit.
↪py:782 in iterconsume
    `yield self.ensure(_error_callback, _consume)`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/impl_rabbit.
↪py:688 in ensure
    `ret, channel = autoretry_method()`

/usr/local/lib/python2.7/dist-packages/kombu/connection.py:436 in _ensured
    `return fun(*args, **kwargs)`

/usr/local/lib/python2.7/dist-packages/kombu/connection.py:508 in __call__
    `return fun(*args, channel=channels[0], **kwargs), channels[0]`
```

(continues on next page)

(continued from previous page)

```
/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/impl_rabbit.
↪py:675 in execute_method
    `method()`

/usr/local/lib/python2.7/dist-packages/oslo_messaging/_drivers/impl_rabbit.
↪py:774 in _consume
    `return self.connection.drain_events(timeout=poll_timeout)`

/usr/local/lib/python2.7/dist-packages/kombu/connection.py:275 in drain_events
    `return self.transport.drain_events(self.connection, **kwargs)`

/usr/local/lib/python2.7/dist-packages/kombu/transport/pyamqp.py:91 in drain_
↪events
    `return connection.drain_events(**kwargs)`

/usr/local/lib/python2.7/dist-packages/amqp/connection.py:302 in drain_events
    `chanmap, None, timeout=timeout,`

/usr/local/lib/python2.7/dist-packages/amqp/connection.py:365 in _wait_
↪multiple
    `channel, method_sig, args, content = read_timeout(timeout)`

/usr/local/lib/python2.7/dist-packages/amqp/connection.py:336 in read_timeout
    `return self.method_reader.read_method()`

/usr/local/lib/python2.7/dist-packages/amqp/method_framing.py:186 in read_
↪method
    `self._next_method()`

/usr/local/lib/python2.7/dist-packages/amqp/method_framing.py:107 in _next_
↪method
    `frame_type, channel, payload = read_frame()`

/usr/local/lib/python2.7/dist-packages/amqp/transport.py:154 in read_frame
    `frame_header = read(7, True)`

/usr/local/lib/python2.7/dist-packages/amqp/transport.py:277 in _read
    `s = recv(n - len(rbuf))`

/usr/local/lib/python2.7/dist-packages/eventlet/greenio/base.py:326 in recv
    `timeout_exc=socket.timeout("timed out"))`

/usr/local/lib/python2.7/dist-packages/eventlet/greenio/base.py:201 in _
↪trampoline
    `mark_as_closed=self._mark_as_closed)`

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/__init__.py:162 in␣
↪trampoline
    `return hub.switch()`
```

(continues on next page)

(continued from previous page)

```
/usr/local/lib/python2.7/dist-packages/eventlet/hubs/hub.py:294 in switch
    `return self.greenlet.switch()`

------                          Green Thread                          ------

/usr/local/bin/designate-central:10 in <module>
    `sys.exit(main())`

/opt/stack/designate/designate/cmd/central.py:37 in main
    `service.wait()`

/opt/stack/designate/designate/service.py:356 in wait
    `_launcher.wait()`

/opt/stack/designate/designate/openstack/common/service.py:187 in wait
    `status, signo = self._wait_for_exit_or_signal(ready_callback)`

/opt/stack/designate/designate/openstack/common/service.py:170 in _wait_for_
↪exit_or_signal
    `super(ServiceLauncher, self).wait()`

/opt/stack/designate/designate/openstack/common/service.py:133 in wait
    `self.services.wait()`

/opt/stack/designate/designate/openstack/common/service.py:473 in wait
    `self.tg.wait()`

/opt/stack/designate/designate/openstack/common/threadgroup.py:145 in wait
    `x.wait()`

/opt/stack/designate/designate/openstack/common/threadgroup.py:47 in wait
    `return self.thread.wait()`

/usr/local/lib/python2.7/dist-packages/eventlet/greenthread.py:175 in wait
    `return self._exit_event.wait()`

/usr/local/lib/python2.7/dist-packages/eventlet/event.py:121 in wait
    `return hubs.get_hub().switch()`

/usr/local/lib/python2.7/dist-packages/eventlet/hubs/hub.py:294 in switch
    `return self.greenlet.switch()`

------                          Green Thread                          ------

No Traceback!

============================================================================
====                            Processes                               ====
```

(continues on next page)

(continued from previous page)

```
================================================================================
Process 15097 (under 7312) [ run by: stanzgy (1000), state: running ]

================================================================================
====                          Configuration                               ====
================================================================================

backend:bind9:
  masters =
    127.0.0.1:5354
  rndc-config-file = None
  rndc-host = 127.0.0.1
  rndc-key-file = None
  rndc-port = 953
  server_ids =

backend:fake:
  masters =
    127.0.0.1:5354
  server_ids =

backend:powerdns:
  backend = sqlalchemy
  connection = ***
  connection_debug = 0
  connection_trace = False
  db_inc_retry_interval = True
  db_max_retries = 20
  db_max_retry_interval = 10
  db_retry_interval = 1
  idle_timeout = 3600
  masters =
    10.180.64.117:5354
  max_overflow = None
  max_pool_size = None
  max_retries = 10
  min_pool_size = 1
  mysql_sql_mode = TRADITIONAL
  pool_timeout = None
  retry_interval = 10
  server_ids =
    f26e0b32-736f-4f0a-831b-039a415c481e
  slave_connection = ***
  sqlite_db = oslo.sqlite
  sqlite_synchronous = True
  use_db_reconnect = False

backend:powerdns:f26e0b32-736f-4f0a-831b-039a415c481e:
  backend = None
```

(continues on next page)

(continued from previous page)

```
  connection = ***
  connection_debug = None
  connection_trace = None
  db_inc_retry_interval = None
  db_max_retries = None
  db_max_retry_interval = None
  db_retry_interval = None
  host = 10.180.64.117
  idle_timeout = None
  masters = None
  max_overflow = None
  max_pool_size = None
  max_retries = None
  min_pool_size = None
  mysql_sql_mode = None
  pool_timeout = None
  port = 53
  retry_interval = None
  slave_connection = ***
  sqlite_db = None
  sqlite_synchronous = None
  tsig-key = None
  use_db_reconnect = None

default:
  allowed_remote_exmods =
  backdoor_port = None
  backlog = 4096
  central-topic = central
  config-dir = None
  config-file =
    /etc/designate/designate.conf
  control_exchange = designate
  debug = True
  default-soa-expire = 86400
  default-soa-minimum = 3600
  default-soa-refresh-min = 3500
  default-soa-refresh-max = 3600
  default-soa-retry = 600
  default-ttl = 3600
  default_log_levels =
    amqp=WARN
    amqplib=WARN
    boto=WARN
    eventlet.wsgi.server=WARN
    keystone=INFO
    keystonemiddleware.auth_token=INFO
    oslo.messaging=WARN
    sqlalchemy=WARN
```

(continues on next page)

(continued from previous page)

```
    stevedore=WARN
    suds=INFO
  fatal_deprecations = False
  host = cns-dev2
  instance_format = [instance: %(uuid)s]
  instance_uuid_format = [instance: %(uuid)s]
  log-config-append = None
  log-date-format = %Y-%m-%d %H:%M:%S
  log-dir = /opt/stack/logs/designate
  log-file = None
  log-format = None
  logging_context_format_string = %(asctime)s.%(msecs)03d %(color)s
↪%(levelname)s %(name)s [[01;36m%(request_id)s [00;36m%(user)s %(tenant)s
↪%(color)s] [01;35m%(instance)s%(color)s%(message)s[00m
  logging_debug_format_suffix = [00;33mfrom (pid=%(process)d) %(funcName)s
↪%(pathname)s:%(lineno)d[00m
  logging_default_format_string = %(asctime)s.%(msecs)03d %(color)s
↪%(levelname)s %(name)s [[00;36m-%(color)s] [01;35m%(instance)s%(color)s
↪%(message)s[00m
  logging_exception_prefix = %(color)s%(asctime)s.%(msecs)03d TRACE %(name)s␣
↪[01;35m%(instance)s[00m
  mdns-topic = mdns
  network_api = neutron
  notification_driver =
  notification_topics =
    notifications
  policy_default_rule = default
  policy_dirs =
    policy.d
  policy_file = /etc/designate/policy.yaml
  pool-manager-topic = pool_manager
  publish_errors = False
  pybasedir = /opt/stack/designate
  quota-domain-records = 500
  quota-domain-recordsets = 500
  quota-domains = 10
  quota-driver = storage
  quota-recordset-records = 20
  root-helper = sudo designate-rootwrap /etc/designate/rootwrap.conf
  rpc_backend = rabbit
  rpc_thread_pool_size = 64
  state-path = /opt/stack/data/designate
  syslog-log-facility = LOG_USER
  tcp_keepidle = 600
  transport_url = None
  use-syslog = False
  use-syslog-rfc-format = False
  use_stderr = True
  verbose = True
```

(continues on next page)

(continued from previous page)

```
network_api:neutron:
  ca_certificates_file = None
  endpoint_type = publicURL
  endpoints = None
  insecure = False
  timeout = 30

oslo_concurrency:
  disable_process_locking = False
  lock_path = None

oslo_messaging_rabbit:
  amqp_auto_delete = False
  amqp_durable_queues = False
  fake_rabbit = False
  kombu_reconnect_delay = 1.0
  kombu_ssl_ca_certs =
  kombu_ssl_certfile =
  kombu_ssl_keyfile =
  kombu_ssl_version =
  rabbit_ha_queues = False
  rabbit_host = localhost
  rabbit_hosts =
    127.0.0.1
  rabbit_login_method = AMQPLAIN
  rabbit_max_retries = 0
  rabbit_password = ***
  rabbit_port = 5672
  rabbit_retry_backoff = 2
  rabbit_retry_interval = 1
  rabbit_use_ssl = False
  rabbit_userid = stackrabbit
  rabbit_virtual_host = /
  rpc_conn_pool_size = 30

proxy:
  http_proxy = None
  https_proxy = None
  no_proxy =

service:central:
  default_pool_id = 794ccc2c-d751-44fe-b57f-8894c9f5c842
  enabled-notification-handlers =
  managed_resource_email = hostmaster@example.com
  managed_resource_tenant_id = None
  max_domain_name_len = 255
  max_recordset_name_len = 255
  min_ttl = None
```

(continues on next page)

(continued from previous page)

```
  storage-driver = sqlalchemy
  workers = None

service:pool_manager:
  backends =
    powerdns
  cache-driver = sqlalchemy
  enable-recovery-timer = True
  enable-sync-timer = True
  periodic-recovery-interval = 120
  periodic-sync-interval = 300
  periodic-sync-seconds = None
  poll-delay = 1
  poll-max-retries = 3
  poll-retry-interval = 2
  poll-timeout = 30
  pool-id = 794ccc2c-d751-44fe-b57f-8894c9f5c842
  threshold-percentage = 100
  workers = None

ssl:
  ca_file = None
  cert_file = None
  key_file = None

storage:sqlalchemy:
  backend = sqlalchemy
  connection = ***
  connection_debug = 0
  connection_trace = False
  db_inc_retry_interval = True
  db_max_retries = 20
  db_max_retry_interval = 10
  db_retry_interval = 1
  idle_timeout = 3600
  max_overflow = None
  max_pool_size = None
  max_retries = 10
  min_pool_size = 1
  mysql_sql_mode = TRADITIONAL
  pool_timeout = None
  retry_interval = 10
  slave_connection = ***
  sqlite_db = oslo.sqlite
  sqlite_synchronous = True
  use_db_reconnect = False
```

1.3.4 Source Code Documentation

API

API Middleware

class designate.api.middleware.**APIv2ValidationErrorMiddleware**(*application*)

Bases: Middleware

class designate.api.middleware.**ContextMiddleware**(*application*, *conf=None*)

Bases: Middleware

make_context(*request*, **args*, ***kwargs*)

class designate.api.middleware.**FaultWrapperMiddleware**(*application*)

Bases: Middleware

class designate.api.middleware.**KeystoneContextMiddleware**(*application*)

Bases: *ContextMiddleware*

process_request(*request*)

Called on each request.

If this returns None, the next application down the stack will be executed. If it returns a response then that response will be returned and execution will stop here.

class designate.api.middleware.**MaintenanceMiddleware**(*application*)

Bases: Middleware

process_request(*request*)

Called on each request.

If this returns None, the next application down the stack will be executed. If it returns a response then that response will be returned and execution will stop here.

class designate.api.middleware.**NoAuthContextMiddleware**(*application*)

Bases: *ContextMiddleware*

process_request(*request*)

Called on each request.

If this returns None, the next application down the stack will be executed. If it returns a response then that response will be returned and execution will stop here.

class designate.api.middleware.**NormalizeURIMiddleware**(*application*, *conf=None*)

Bases: Middleware

class designate.api.middleware.**TestContextMiddleware**(*application*, *tenant_id=None*, *user_id=None*)

Bases: *ContextMiddleware*

process_request(*request*)

Called on each request.

If this returns None, the next application down the stack will be executed. If it returns a response then that response will be returned and execution will stop here.

designate.api.middleware.**auth_pipeline_factory**(*loader*, *global_conf*, ***local_conf*)

A paste pipeline replica that keys off of auth_strategy.

Code nabbed from cinder.

API Service

class designate.api.service.**Service**

Bases: WSGIService

property service_name

start()

Start a service.

stop(*graceful=True*)

Stop a service.

Parameters

graceful indicates whether to wait for all threads to finish or terminate them instantly

property wsgi_application

Backend

Backend Base

class designate.backend.base.**Backend**(*target*)

Bases: DriverPlugin

Base class for backend implementations

abstract create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

abstract delete_zone(*context*, *zone*, *zone_params*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

update_zone(*context*, *zone*)

Update a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

Backend Bind9

Bind 9 backend. Create and delete zones by executing rndc

class designate.backend.impl_bind9.**Bind9Backend**(*target*)

Bases: *Backend*

create_zone(*context*, *zone*)

Create a new Zone by executin rndc, then notify mDNS Do not raise exceptions if the zone already exists.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a new Zone by executin rndc Do not raise exceptions if the zone does not exist.

get_zone(*context*, *zone*)

Returns True if zone exists and False if not

update_zone(*context*, *zone*)

Update a DNS zone.

This will execute a rndc modzone if the zone already exists but masters might need to be refreshed. Or, will create the zone if it does not exist.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

Backend Designate

class designate.backend.impl_designate.**DesignateBackend**(*target*)

Bases: *Backend*

Support for Designate to Designate using Secondary zones.

property client

create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

Backend Dynect

class designate.backend.impl_dynect.**DynClient**(*customer_name*, *user_name*, *password*, *timeout*, *timings*, *verify=True*, *retries=1*, *pool_maxsize=10*, *pool_connections=10*)

Bases: object

DynECT service client.

https://help.dynect.net/rest/

delete(**args*, ***kwargs*)

login()

logout()

post(**args*, ***kwargs*)

put(**args*, ***kwargs*)

request(*method*, *url*, *retries=2*, ***kwargs*)

exception designate.backend.impl_dynect.**DynClientAuthError**(*data=None*, *job_id=None*, *msgs=None*, *http_status=None*, *url=None*, *method=None*, *details=None*)

Bases: *DynClientError*

exception designate.backend.impl_dynect.**DynClientError**(*data=None*, *job_id=None*, *msgs=None*, *http_status=None*, *url=None*, *method=None*, *details=None*)

Bases: Backend

The base exception class for all HTTP exceptions.

static from_response(*response*, *details=None*)

exception designate.backend.impl_dynect.**DynClientOperationBlocked**(**args*, ***kwargs*)

Bases: BadRequest, *DynClientError*

error_type = 'operation_blocked'

class designate.backend.impl_dynect.**DynECTBackend**(*target*)

Bases: *Backend*

Support for DynECT as a secondary DNS.

create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.

- **zone** the DNS zone.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

get_client()

exception designate.backend.impl_dynect.**DynTimeoutError**

Bases: Backend

A job timedout.

error_code = 408

error_type = 'dyn_timeout'

Backend Infoblox

class designate.backend.impl_infoblox.**InfobloxBackend**(**args*, ***kwargs*)

Bases: *Backend*

Provides a Designate Backend for Infoblox

create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

get_network_view(*project_id*)

get_or_create_dns_view(*net_view*, *create_if_missing=True*)

get_or_create_network_view(*project_id*)

property is_multi_project

static parse_wapi_url(*wapi_url*)

restart_if_needed()

Backend Nsd4

class designate.backend.impl_nsd4.**NSD4Backend**(*target*)

Bases: *Backend*

NSDCT_VERSION = 'NSDCT1'

create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

Backend Fake

class designate.backend.impl_fake.**FakeBackend**(*target*)

Bases: *Backend*

create_zone(*context*, *zone*)

Create a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone.

Parameters

- **context** Security context information.
- **zone** the DNS zone.

Backend PowerDNS 4

class designate.backend.impl_pdns4.**PDNS4Backend**(*target*)

Bases: *Backend*

create_zone(*context*, *zone*)

Create a DNS zone

delete_zone(*context*, *zone*, *zone_params=None*)

Delete a DNS zone

Central

Central RPC API

class designate.central.rpcapi.**CentralAPI**(*topic=None*)

Bases: object

Client side of the central RPC API.

API version history:

1.0 - Initial version 1.1 - Add new finder methods 1.2 - Add get_tenant and get_tenants 1.3 - Add get_absolute_limits 2.0 - Renamed most get_resources to find_resources 2.1 - Add quota methods 3.0 - RecordSet Changes 3.1 - Add floating ip ptr methods 3.2 - TLD Api changes 3.3 - Add methods for blacklisted domains 4.0 - Create methods now accept designate objects 4.1 - Add methods for server pools 4.2 - Add methods for pool manager integration 4.3 - Added Zone Transfer Methods 5.0 - Remove dead server code 5.1 - Add xfr_zone 5.2 - Add Zone Import methods 5.3 - Add Zone Export method 5.4 - Add asynchronous Zone Export methods 5.5 - Add deleted zone purging task 5.6 - Changed purge_zones function args 6.0 - Renamed domains to zones 6.1 - Add ServiceStatus methods 6.2 - Changed find_recordsets method args 6.3 - Changed update_status method args 6.4 - Removed unused record and diagnostic methods 6.5 - Removed additional unused methods 6.6 - Add methods for shared zones 6.7 - Add increment_zone_serial 6.8 - Add managed recordset methods 6.9 - Removed unused methods 6.10 - Add Zone Pool Move method

RPC_API_VERSION = '6.10'

RPC_LOGGING_DISALLOW = ['update_service_status', 'get_instance', '__init__']

count_report(*context*, *criterion=None*)

create_blacklist(*context*, *blacklist*)

create_managed_records(*context*, *zone_id*, *records_values*, *recordset_values*)

create_pool(*context*, *pool*)

create_recordset(*context*, *zone_id*, *recordset*)

create_tld(*context*, *tld*)

create_tsigkey(*context*, *tsigkey*)

create_zone(*context*, *zone*)

create_zone_export(*context*, *zone_id*)

create_zone_import(*context*, *request_body*)

create_zone_transfer_accept(*context*, *zone_transfer_accept*)

create_zone_transfer_request(*context*, *zone_transfer_request*)

delete_blacklist(*context*, *blacklist_id*)

delete_managed_records(*context*, *zone_id=None*, *criterion=None*)

delete_pool(*context*, *pool_id*)

delete_recordset(*context*, *zone_id*, *recordset_id*, *increment_serial=True*)

delete_tld(*context*, *tld_id*)

delete_tsigkey(*context*, *tsigkey_id*)

delete_zone(*context*, *zone_id*)

delete_zone_export(*context*, *zone_export_id*)

delete_zone_import(*context*, *zone_import_id*)

delete_zone_transfer_request(*context*, *zone_transfer_request_id*)

export_zone(*context*, *zone_id*)

find_blacklists(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_pool(*context*, *criterion=None*)

find_pools(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_recordsets(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*, *force_index=False*)

find_service_status(*context*, *criterion=None*)

find_service_statuses(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_shared_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_tenants(*context*)

find_tlds(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_tsigkeys(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_exports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_imports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_transfer_accepts(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_transfer_requests(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

get_absolute_limits(*context*)

get_blacklist(*context*, *blacklist_id*)

get_floatingip(*context*, *region*, *floatingip_id*)

classmethod get_instance()

The rpc.get_client() which is called upon the API object initialization will cause a assertion error if the designate.rpc.TRANSPORT isnt setup by rpc.init() before.

This fixes that by creating the rpcapi when demanded.

get_pool(*context*, *pool_id*)

get_quotas(*context*, *tenant_id*)

get_recordset(*context*, *zone_id*, *recordset_id*)

get_shared_zone(*context*, *zone_id*, *zone_share_id*)

get_tenant(*context*, *tenant_id*)

get_tld(*context*, *tld_id*)

get_tsigkey(*context*, *tsigkey_id*)

get_zone(*context*, *zone_id*)

get_zone_export(*context*, *zone_export_id*)

get_zone_import(*context*, *zone_import_id*)

get_zone_ns_records(*context*, *zone_id*)

get_zone_transfer_accept(*context*, *zone_transfer_accept_id*)

get_zone_transfer_request(*context*, *zone_transfer_request_id*)

increment_zone_serial(*context*, *zone*)

list_floatingips(*context*)

pool_move_zone(*context*, *zone_id*, *target_pool_id*)

purge_zones(*context*, *criterion*, *limit=None*)

reset_quotas(*context*, *tenant_id*)

set_quota(*context*, *tenant_id*, *resource*, *hard_limit*)

share_zone(*context*, *zone_id*, *shared_zone*)

unshare_zone(*context*, *zone_id*, *zone_share_id*)

update_blacklist(*context*, *blacklist*)

update_floatingip(*context*, *region*, *floatingip_id*, *values*)

update_pool(*context*, *pool*)

update_recordset(*context*, *recordset*, *increment_serial=True*)

update_service_status(*context*, *service_status*)

update_status(*context*, *zone_id*, *status*, *serial*, *action=None*)

update_tld(*context*, *tld*)

update_tsigkey(*context*, *tsigkey*)

update_zone(*context*, *zone*, *increment_serial=True*)

update_zone_export(*context*, *zone_export*)

update_zone_transfer_request(*context*, *zone_transfer_request*)

xfr_zone(*context*, *zone_id*)

designate.central.rpcapi.**reset**()

Central Service

class designate.central.service.**Service**

Bases: RPCService

RPC_API_VERSION = '6.10'

count_records(*context*, *criterion=None*)

count_recordsets(*context*, *criterion=None*)

count_report(*context*, *criterion=None*)

count_tenants(*context*)

count_zones(*context*, *criterion=None*)

create_blacklist(*context*, *blacklist*)

create_managed_records(*context*, *zone_id*, *records_values*, *recordset_values*)

create_pool(*context*, *pool*)

create_recordset(*context*, *zone_id*, *recordset*, *increment_serial=True*)

create_tld(*context*, *tld*)

create_tsigkey(*context*, *tsigkey*)

create_zone(*context*, *zone*)

Create zone: perform checks and then call _create_zone()

create_zone_export(*context*, *zone_id*)

create_zone_import(*context*, *request_body*)

create_zone_transfer_accept(*context*, *zone_transfer_accept*)

create_zone_transfer_request(*context*, *zone_transfer_request*)

delete_blacklist(*context*, *blacklist_id*)

delete_managed_records(*context*, *zone_id*, *criterion*)

delete_pool(*context*, *pool_id*)

delete_recordset(*context*, *zone_id*, *recordset_id*, *increment_serial=True*)

delete_tld(*context*, *tld_id*)

delete_tsigkey(*context*, *tsigkey_id*)

delete_zone(*context*, *zone_id*)

Delete or abandon a zone On abandon, delete the zone from the DB immediately. Otherwise, set action to DELETE and status to PENDING and poke Pool Managers delete_zone to update the resolvers. PM will then poke back to set action to NONE and status to DELETED

delete_zone_export(*context*, *zone_export_id*)

delete_zone_import(*context*, *zone_import_id*)

delete_zone_transfer_request(*context*, *zone_transfer_request_id*)

export_zone(*context*, *zone_id*)

find_blacklists(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_pool(*context*, *criterion=None*)

find_pools(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_records(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_recordset(*context*, *criterion=None*)

find_recordsets(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*, *force_index=False*)

find_service_status(*context*, *criterion=None*)

find_service_statuses(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

List service statuses.

find_shared_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_tenants(*context*)

find_tlds(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_tsigkeys(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_exports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_imports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_transfer_accepts(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_transfer_requests(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

List existing zones including the ones flagged for deletion.

get_absolute_limits(*context*)

get_blacklist(*context*, *blacklist_id*)

get_floatingip(*context*, *region*, *floatingip_id*)

Get Floating IP PTR

get_pool(*context*, *pool_id*)

get_quotas(*context*, *tenant_id*)

get_recordset(*context*, *zone_id*, *recordset_id*)

get_shared_zone(*context*, *zone_id*, *zone_share_id*)

get_tenant(*context*, *tenant_id*)

get_tld(*context*, *tld_id*)

get_tsigkey(*context*, *tsigkey_id*)

get_zone(*context*, *zone_id*, *apply_tenant_criteria=True*)

Get a zone, even if flagged for deletion

get_zone_export(*context*, *zone_export_id*)

get_zone_import(*context*, *zone_import_id*)

get_zone_ns_records(*context*, *zone_id=None*, *criterion=None*)

get_zone_transfer_accept(*context*, *zone_transfer_accept_id*)

get_zone_transfer_request(*context*, *zone_transfer_request_id*)

increment_zone_serial(*context*, *zone*)

list_floatingips(*context*)

List Floating IPs PTR

A) **We have service_catalog in the context and do a lookup using the** token pr Neutron in the SC

B) We lookup FIPs using the configured values for this deployment.

pool_move_zone(*context*, *zone_id*, *target_pool_id=None*)

Move zone. Perform checks and then create zone in destination pool

Returns

moved zone

purge_zones(*context*, *criterion*, *limit=None*)

Purge deleted zones. :returns: number of purged zones

property quota

reset_quotas(*context*, *tenant_id*)

property scheduler

property service_name

set_quota(*context*, *tenant_id*, *resource*, *hard_limit*)

share_zone(*context*, *zone_id*, *shared_zone*)

start()

Start a service.

stop(*graceful=True*)

Stop a service.

Parameters

graceful indicates whether to wait for all threads to finish or terminate them instantly

property storage

target = <Target version=6.10>

unshare_zone(*context*, *zone_id*, *zone_share_id*)

update_blacklist(*context*, *blacklist*)

update_floatingip(*context*, *region*, *floatingip_id*, *values*)

We strictly see if values[ptrdname] is str or None and set / unset the requested FloatingIPs PTR record based on that.

update_pool(*context*, *pool*)

update_recordset(*context*, *recordset*, *increment_serial=True*)

update_service_status(*context*, *service_status*)

update_status(*context*, *zone_id*, *status*, *serial*, *action=None*)

Parameters

- **context** Security context information.
- **zone_id** The ID of the designate zone.
- **status** The status, SUCCESS or ERROR.
- **serial** The consensus serial number for the zone.
- **action** The action, CREATE, UPDATE, DELETE or NONE.

Returns
updated zone

update_tld(*context*, *tld*)

update_tsigkey(*context*, *tsigkey*)

update_zone(*context*, *zone*, *increment_serial=True*)

Update zone. Perform checks and then call _update_zone()

Returns
updated zone

update_zone_export(*context*, *zone_export*)

update_zone_import(*context*, *zone_import*)

update_zone_transfer_request(*context*, *zone_transfer_request*)

property worker_api

xfr_zone(*context*, *zone_id*)

MDNS

MDNS Handler

class designate.mdns.handler.**RequestHandler**(*storage*, *tg*)

Bases: object

property worker_api

MDNS Service

class designate.mdns.service.**Service**

Bases: Service

property dns_application

property service_name

start()

Start a service.

stop(*graceful=True*)

Stop a service.

Parameters

graceful indicates whether to wait for all threads to finish or terminate them instantly

property storage

Objects

Objects Base

class designate.objects.base.**AttributeListObjectMixin**(**args*, ***kwargs*)

Bases: *ListObjectMixin*

Mixin class for Attribute objects.

Attribute objects are ListObjects, whos memebers have a key and value property, which should be exposed on the list itself as list.<key>.

classmethod from_dict(*_dict*)

get(*key*, *default=None*)

to_dict()

class designate.objects.base.**DesignateObject**(**args*, ***kwargs*)

Bases: VersionedObject

OBJ_PROJECT_NAMESPACE = 'designate'

OBJ_SERIAL_NAMESPACE = 'designate_object'

STRING_KEYS = []

classmethod from_dict(*_dict*)

classmethod from_list(*_list*)

classmethod from_primitive(*primitive*, *context=None*)

nested_sort(*key*, *value*)

This function ensure that change fields list is sorted. :param key: :param value: :return:

obj_attr_is_set(*name*)

Return True or False depending of if a particular attribute has had an attributes value explicitly set.

classmethod obj_cls_from_name(*name*)

property obj_fields

obj_get_original_value(*field*)

Returns the original value of a field.

obj_reset_changes(*fields=None*, *recursive=False*)

Reset the list of fields that have been changed.

Parameters

- **fields** List of fields to reset, or all if None.
- **recursive** Call obj_reset_changes(recursive=True) on any sub-objects within the list of fields being reset.

This is NOT revert to previous values.

Specifying fields on recursive resets will only be honored at the top level. Everything below the top will reset all.

to_dict()

Convert the object to a simple dictionary.

to_primitive()

update(*values*)

Update a objects fields with the supplied key/value pairs

validate()

class designate.objects.base.**DesignateRegistry**(**args*, ***kwargs*)

Bases: VersionedObjectRegistry

registration_hook(*cls*, *index*)

class designate.objects.base.**ListObjectMixin**(**args*, ***kwargs*)

Bases: ObjectListBase

LIST_ITEM_TYPE

alias of *DesignateObject*

append(*value*)

Append a value to the list

count(*value*)

List count of value occurrences

extend(*values*)

Extend the list by appending all the items in the given list

classmethod from_list(*_list*)

index(*value*)

List index of value

insert(*index*, *value*)

Insert a value into the list at the given index

pop(*index*)

Pop a value from the list

remove(*value*)

Remove a value from the list

to_list()

class designate.objects.base.PagedListObjectMixin

Bases: object

Mixin class for List objects.

This adds fields that would populate API metadata for collections.

fields = {'total_count': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

class designate.objects.base.PersistentObjectMixin

Bases: object

Mixin class for Persistent objects.

This adds the fields that we use in common for all persistent objects.

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'version': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

class designate.objects.base.SoftDeleteObjectMixin

Bases: object

Mixin class for Soft-Deleted objects.

This adds the fields that we use in common for all soft-deleted objects.

fields = {'deleted': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'deleted_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

designate.objects.base.get_dict_attr(*klass*, *attr*)

Objects Backlist

class designate.objects.blacklist.Blacklist(**args*, ***kwargs*)

Bases: VersionedObjectDictCompat, *PersistentObjectMixin*, *DesignateObject*

STRING_KEYS = ['id', 'pattern']

property created_at

property description

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'description': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'pattern': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'version': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property pattern

property updated_at

property version

class designate.objects.blacklist.**BlacklistList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *Blacklist*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects Zone

class designate.objects.zone.**Zone**(**args*, ***kwargs*)

Bases: *DesignateObject*, VersionedObjectDictCompat, *PersistentObjectMixin*, *SoftDeleteObjectMixin*

STRING_KEYS = ['id', 'type', 'name', 'pool_id', 'serial', 'action', 'status', 'shard']

property action

property attributes

property created_at

property delayed_notify

property deleted

property deleted_at

property description

```
property email

property expire
```

```
fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'attributes': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'delayed_notify': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'deleted': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'deleted_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'email': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'expire': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'increment_serial': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'masters': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'minimum': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'parent_zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'pool_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordsets': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'refresh': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'retry': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shared': Boolean(default=False,nullable=True), 'status':
Enum(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,
nullable=True,valid_values=['ACTIVE', 'PENDING', 'ERROR', 'DELETED',
'SUCCESS', 'NO_ZONE']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'transferred_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'ttl':
Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'type':
Enum(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,
nullable=True,valid_values=['SECONDARY', 'PRIMARY', 'CATALOG']),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

get_master_by_ip(*host*)

Utility to get the master by its ip for this zone.

property id

property increment_serial

property masters

property minimum

property name

property parent_zone_id

property pool_id

property recordsets

property refresh

property retry

property serial

property shard

property shared

property status

property tenant_id

property transferred_at

property ttl

property type

property updated_at

validate()

property version

class designate.objects.zone.ZoneList(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*, *PagedListObjectMixin*

LIST_ITEM_TYPE

alias of *Zone*

```
fields = {'objects':  List(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'total_count':  Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

property objects

property total_count

Objects Pool

class designate.objects.pool.**Pool**(**args*, ***kwargs*)

Bases: VersionedObjectDictCompat, *PersistentObjectMixin*, *DesignateObject*

STRING_KEYS = ['id', 'name']

property also_notifies

property attributes

property catalog_zone

property created_at

property description

fields = {'also_notifies': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'attributes': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'catalog_zone': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'nameservers': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'ns_records': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'provisioner': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'targets': Object(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property name

property nameservers

property ns_records

property provisioner

property targets

property tenant_id

property updated_at

property version

class designate.objects.pool.PoolList(*args, **kwargs)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *Pool*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects Quota

class designate.objects.quota.Quota(*args, **kwargs)

Bases: VersionedObjectDictCompat, *PersistentObjectMixin*, *DesignateObject*

STRING_KEYS = ['resource', 'tenant_id', 'hard_limit']

property created_at

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hard_limit': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'resource': Enum(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True,valid_values=['api_export_size', 'recordset_records', 'zone_records', 'zone_recordsets', 'zones']), 'tenant_id': Any(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'version': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property hard_limit

property id

property resource

property tenant_id

property updated_at

property version

class designate.objects.quota.QuotaList(*args, **kwargs)

Bases: ListObjectMixin, DesignateObject

LIST_ITEM_TYPE

alias of Quota

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

classmethod from_dict(_dict)

property objects

to_dict()

Convert the object to a simple dictionary.

Objects Record

class designate.objects.record.Record(*args, **kwargs)

Bases: DesignateObject, PersistentObjectMixin, VersionedObjectDictCompat

STRING_KEYS = ['id', 'recordset_id', 'data']

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

classmethod get_recordset_schema_changes()

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.record.**RecordList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *Record*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects Recordset

class designate.objects.recordset.**RecordSet**(**args*, ***kwargs*)

Bases: *DesignateObject*, VersionedObjectDictCompat, *PersistentObjectMixin*

STRING_KEYS = ['id', 'type', 'name', 'zone_id', 'shard']

property action

property created_at

property description

fields = {'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'records': PolymorphicObject(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'ttl':
Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'type':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property managed

property name

property records

property shard

property status

property tenant_id

property ttl

property type

property updated_at

validate()

property version

property zone_id

property zone_name

class designate.objects.recordset.**RecordSetList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*, *PagedListObjectMixin*

LIST_ITEM_TYPE

alias of *RecordSet*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'total_count': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property objects

property total_count

Objects Tenant

class designate.objects.tenant.**Tenant**(**args*, ***kwargs*)

Bases: *DesignateObject*, VersionedObjectDictCompat

STRING_KEYS = ['id']

fields = {'id': Any(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'zone_count': Any(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'zones': Any(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property zone_count

property zones

class designate.objects.tenant.**TenantList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *Tenant*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects TLD

class designate.objects.tld.**Tld**(**args*, ***kwargs*)

Bases: VersionedObjectDictCompat, *PersistentObjectMixin*, *DesignateObject*

STRING_KEYS = ['id', 'name']

property created_at

property description

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'description': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'version': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property name

property updated_at

property version

class designate.objects.tld.**TldList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *Tld*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects TSigKey

class designate.objects.tsigkey.**TsigKey**(**args*, ***kwargs*)

Bases: VersionedObjectDictCompat, *PersistentObjectMixin*, *DesignateObject*

STRING_KEYS = ['id', 'name', 'algorithm', 'scope', 'resource_id']

property algorithm

property created_at

fields = {'algorithm': Enum(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False,valid_values=['hmac-md5', 'hmac-sha1', 'hmac-sha224', 'hmac-sha256', 'hmac-sha384', 'hmac-sha512']), 'created_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'name': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'resource_id': UUID(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'scope': Enum(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False,valid_values=['POOL', 'ZONE']), 'secret': String(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False), 'updated_at': DateTime(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'version': Integer(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

property id

property name

property resource_id

property scope

property secret

property updated_at

validate()

property version

class designate.objects.tsigkey.**TsigKeyList**(**args*, ***kwargs*)

Bases: *ListObjectMixin*, *DesignateObject*

LIST_ITEM_TYPE

alias of *TsigKey*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects A Record

class designate.objects.rrdata_a.**A**(**args*, ***kwargs*)

Bases: *Record*

A Resource Record Type Defined in: RFC1035

```
RECORD_TYPE = 1

property action

property address

property created_at

property data

property description
```

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'address': IPV4Address(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_a.AList(*args, **kwargs)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *A*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects AAAA Record

class designate.objects.rrdata_aaaa.AAAA(*args, **kwargs)

Bases: *Record*

AAAA Resource Record Type Defined in: RFC3596

RECORD_TYPE = 28

property action

property address

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'address': IPV6Address(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_aaaa.**AAAAList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *AAAA*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects CNAME Record

class designate.objects.rrdata_cname.**CNAME**(**args*, ***kwargs*)

Bases: *Record*

CNAME Resource Record Type Defined in: RFC1035

RECORD_TYPE = 5

property action

```
property cname

property created_at

property data

property description
```

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'cname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_cname.**CNAMEList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *CNAME*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects MX Record

class designate.objects.rrdata_mx.**MX**(**args*, ***kwargs*)

Bases: *Record*

MX Resource Record Type Defined in: RFC1035

RECORD_TYPE = 15

property action

property created_at

property data

property description

property exchange

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'exchange': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'hash': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'priority': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property priority

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_mx.**MXList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *MX*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects NS Record

class designate.objects.rrdata_ns.**NS**(**args*, ***kwargs*)

Bases: *Record*

NS Resource Record Type Defined in: RFC1035

RECORD_TYPE = 2

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'nsdname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

classmethod get_recordset_schema_changes()

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property nsdname

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_ns.NSList(*args, **kwargs)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *NS*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects PTR Record

class designate.objects.rrdata_ptr.PTR(*args, **kwargs)

Bases: *Record*

PTR Resource Record Type Defined in: RFC1035

RECORD_TYPE = 12

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'ptrdname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property ptrdname

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_ptr.**PTRList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *PTR*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects SOA Record

class designate.objects.rrdata_soa.**SOA**(**args*, ***kwargs*)

Bases: *Record*

SOA Resource Record Type Defined in: RFC1035

```
    RECORD_TYPE = 6

    property action

    property created_at

    property data

    property description

    property expire
```

```
fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'expire': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'hash': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'minimum': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'mname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'refresh': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'retry': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'rname': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property minimum

property mname

property recordset_id

property refresh

property retry

property rname

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_soa.SOAList(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *SOA*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects SPF Record

class designate.objects.rrdata_spf.SPF(*args, **kwargs)

Bases: *Record*

SPF Resource Record Type Defined in: RFC4408

RECORD_TYPE = 99

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'txt_data': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property txt_data

property updated_at

property version

property zone_id

class designate.objects.rrdata_spf.SPFList(*args, **kwargs)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *SPF*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects SRV Record

class designate.objects.rrdata_srv.SRV(*args, **kwargs)

Bases: *Record*

SRV Resource Record Type Defined in: RFC2782

RECORD_TYPE = 33

property action

property created_at

property data

property description

```
fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'port':
Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'priority': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'target': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'weight': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

from_string(*value*)

classmethod get_recordset_schema_changes()

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property port

property priority

property recordset_id

property serial

property shard

property status

property target

property tenant_id

property updated_at

property version

property weight

property zone_id

class designate.objects.rrdata_srv.SRVList(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *SRV*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects TXT Record

class designate.objects.rrdata_txt.**TXT**(**args*, ***kwargs*)

Bases: *Record*

TXT Resource Record Type Defined in: RFC1035

RECORD_TYPE = 16

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'hash':
String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'txt_data': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property txt_data

property updated_at

property version

property zone_id

class designate.objects.rrdata_txt.**TXTList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *TXT*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects SSHFP Record

class designate.objects.rrdata_sshfp.**SSHFP**(**args*, ***kwargs*)

Bases: *Record*

SSHFP Resource Record Type Defined in: RFC4255

RECORD_TYPE = 44

property action

property algorithm

property created_at

property data

property description

```
fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'algorithm': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'fingerprint': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'fp_type': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'hash': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

property fingerprint

property fp_type

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property recordset_id

property serial

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_sshfp.SSHFPList(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *SSHFP*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects NAPTR Record

class designate.objects.rrdata_naptr.**NAPTR**(**args*, ***kwargs*)

Bases: *Record*

NAPTR Resource Record Type Defined in: RFC2915

RECORD_TYPE = 35

property action

property created_at

property data

property description

```
fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['CREATE', 'DELETE',
'UPDATE', 'NONE']), 'created_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'data':
Any(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'description': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'flags': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'hash': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True), 'id':
UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed': Boolean(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_extra': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_name': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_plugin_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_region': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_resource_type': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'managed_tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'order': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'preference': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'recordset_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'regexp': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'replacement': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'serial': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'service': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False),
'shard': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'status': Enum(default=<class 'oslo_versionedobjects.fields.
UnspecifiedDefault'>,nullable=True,valid_values=['ACTIVE', 'PENDING',
'ERROR', 'DELETED']), 'tenant_id': String(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'updated_at': DateTime(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'version': Integer(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True),
'zone_id': UUID(default=<class
'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=True)}
```

property flags

from_string(*value*)

property hash

property id

property managed

property managed_extra

property managed_plugin_name

property managed_plugin_type

property managed_resource_id

property managed_resource_region

property managed_resource_type

property managed_tenant_id

property order

property preference

property recordset_id

property regexp

property replacement

property serial

property service

property shard

property status

property tenant_id

property updated_at

property version

property zone_id

class designate.objects.rrdata_naptr.**NAPTRList**(**args*, ***kwargs*)

Bases: *RecordList*

LIST_ITEM_TYPE

alias of *NAPTR*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.UnspecifiedDefault'>,nullable=False)}

property objects

Objects CAA Record

members
undoc-members
show-inheritance

Objects CERT Record

members
undoc-members
show-inheritance

Quota

Quota Base

class designate.quota.base.Quota

Bases: DriverPlugin

Base class for quota plugins

get_default_quotas(*context*)

abstract get_quota(*context*, *tenant_id*, *resource*)

Get Quota

get_quotas(*context*, *tenant_id*)

limit_check(*context*, *tenant_id*, ***values*)

abstract reset_quotas(*context*, *tenant_id*)

Reset Quotas

abstract set_quota(*context*, *tenant_id*, *resource*, *hard_limit*)

Set Quota

Quota Storage

class designate.quota.impl_storage.StorageQuota

Bases: *Quota*

get_quota(*context*, *tenant_id*, *resource*)

Get Quota

reset_quotas(*context*, *tenant_id*)

Reset Quotas

set_quota(*context*, *tenant_id*, *resource*, *hard_limit*)

Set Quota

Sink

Sink Service

class designate.sink.service.**Service**

Bases: Service

static get_allowed_event_types(*handlers*)

Build a list of all allowed event types.

info(*context*, *publisher_id*, *event_type*, *payload*, *metadata*)

Processes an incoming notification, offering each extension the opportunity to handle it.

static init_extensions()

Loads and prepares all enabled extensions

property service_name

start()

Start a service.

stop(*graceful=True*)

Stop a service.

Parameters

graceful indicates whether to wait for all threads to finish or terminate them instantly

Storage

Storage Base

class designate.storage.sqlalchemy.**SQLAlchemyStorage**

Bases: SQLAlchemy

SQLAlchemy connection

count_records(*context*, *criterion=None*)

Count records

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

count_recordsets(*context*, *criterion=None*)

Count recordsets

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

count_tenants(*context*)

Count tenants

Parameters
context RPC Context.

count_zone_tasks(*context*, *criterion=None*)

count_zone_transfer_accept(*context*, *criterion=None*)

count_zones(*context*, *criterion=None*)
Count zones

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

create_blacklist(*context*, *blacklist*)
Create a Blacklist.

Parameters

- **context** RPC Context.
- **blacklist** Blacklist object with the values to be created.

create_pool(*context*, *pool*)
Create a Pool.

Parameters

- **context** RPC Context.
- **pool** Pool object with the values to be created.

create_pool_also_notify(*context*, *pool_id*, *pool_also_notify*)

create_pool_attribute(*context*, *pool_id*, *pool_attribute*)
Create a PoolAttribute.

Parameters

- **context** RPC Context.
- **pool_id** The ID of the pool to which the attribute belongs.
- **pool_attribute** PoolAttribute object with the values created.

create_pool_nameserver(*context*, *pool_id*, *pool_nameserver*)

create_pool_ns_record(*context*, *pool_id*, *pool_ns_record*)

create_pool_target(*context*, *pool_id*, *pool_target*)

create_pool_target_master(*context*, *pool_target_id*, *pool_target_master*)

create_pool_target_option(*context*, *pool_target_id*, *pool_target_option*)

create_quota(*context*, *quota*)
Create a Quota.

Parameters

- **context** RPC Context.

- **quota** Quota object with the values to be created.

create_record(*context*, *zone_id*, *recordset_id*, *record*)

Create a record on a given Zone ID

Parameters

- **context** RPC Context.
- **zone_id** Zone ID to create the record in.
- **recordset_id** RecordSet ID to create the record in.
- **record** Record object with the values to be created.

create_recordset(*context*, *zone_id*, *recordset*)

Create a recordset on a given Zone ID

Parameters

- **context** RPC Context.
- **zone_id** Zone ID to create the recordset in.
- **recordset** RecordSet object with the values to be created.

create_service_status(*context*, *service_status*)

Create a Service status for a service.

Parameters

- **context** RPC Context.
- **service_status** The status of a service.

create_tld(*context*, *tld*)

Create a TLD.

Parameters

- **context** RPC Context.
- **tld** Tld object with the values to be created.

create_tsigkey(*context*, *tsigkey*)

Create a TSIG Key.

Parameters

- **context** RPC Context.
- **tsigkey** TsigKey object with the values to be created.

create_zone(*context*, *zone*)

Create a new Zone.

Parameters

- **context** RPC Context.
- **zone** Zone object with the values to be created.

create_zone_attribute(*context*, *zone_id*, *zone_attribute*)

create_zone_export(*context*, *zone_export*)

Create a Zone Export.

Parameters

- **context** RPC Context.
- **zone_export** Zone Export object with the values to be created.

create_zone_import(*context*, *zone_import*)

Create a Zone Import.

Parameters

- **context** RPC Context.
- **zone_import** Zone Import object with the values to be created.

create_zone_master(*context*, *zone_id*, *zone_master*)

create_zone_transfer_accept(*context*, *zone_transfer_accept*)

create_zone_transfer_request(*context*, *zone_transfer_request*)

delete_blacklist(*context*, *blacklist_id*)

Delete a Blacklist via ID.

Parameters

- **context** RPC Context.
- **blacklist_id** Delete a Blacklist via ID

delete_pool(*context*, *pool_id*)

Delete the pool with the matching id

Parameters

- **context** RPC Context.
- **pool_id** The ID of the pool to be deleted

delete_pool_also_notify(*context*, *pool_also_notify_id*)

delete_pool_attribute(*context*, *pool_attribute_id*)

Delete the pool with the matching id

Parameters

- **context** RPC Context.
- **pool_attribute_id** The ID of the PoolAttribute to be deleted

delete_pool_nameserver(*context*, *pool_nameserver_id*)

delete_pool_ns_record(*context*, *pool_ns_record_id*)

delete_pool_target(*context*, *pool_target_id*)

delete_pool_target_master(*context*, *pool_target_master_id*)

delete_pool_target_option(*context*, *pool_target_option_id*)

delete_quota(*context*, *quota_id*)

Delete a Quota via ID.

Parameters

- **context** RPC Context.
- **quota_id** Delete a Quota via ID

delete_record(*context*, *record_id*)

Delete a record

Parameters

- **context** RPC Context.
- **record_id** Record ID to delete

delete_recordset(*context*, *recordset_id*)

Delete a recordset

Parameters

- **context** RPC Context.
- **recordset_id** RecordSet ID to delete

delete_tld(*context*, *tld_id*)

Delete a TLD via ID.

Parameters

- **context** RPC Context.
- **tld_id** Delete a TLD via ID

delete_tsigkey(*context*, *tsigkey_id*)

Delete a TSIG Key via ID.

Parameters

- **context** RPC Context.
- **tsigkey_id** Delete a TSIG Key via ID

delete_zone(*context*, *zone_id*)

Delete a Zone

Parameters

- **context** RPC Context.
- **zone_id** Zone ID to delete.

delete_zone_attribute(*context*, *zone_attribute_id*)

delete_zone_export(*context*, *zone_export_id*)

Delete a Zone Export via ID.

Parameters

- **context** RPC Context.
- **zone_export_id** Delete a Zone Export via ID

delete_zone_import(*context*, *zone_import_id*)

Delete a Zone Import via ID.

Parameters

- **context** RPC Context.
- **zone_import_id** Delete a Zone Import via ID

delete_zone_master(*context*, *zone_master_id*)

delete_zone_shares(*zone_id*)

Delete all of the zone shares for a specific zone.

Parameters

zone_id The zone ID to check.

delete_zone_transfer_accept(*context*, *zone_transfer_accept_id*)

delete_zone_transfer_request(*context*, *zone_transfer_request_id*)

find_blacklist(*context*, *criterion*)

Find a single Blacklist.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_blacklists(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find Blacklists

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_pool(*context*, *criterion*)

Find a single Pool.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_pool_also_notifies(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_pool_also_notify(*context*, *criterion*)

find_pool_attribute(*context*, *criterion*)

Find a single PoolAttribute

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_pool_attributes(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find all PoolAttributes

Parameters

- **context** RPC Context
- **criterion** Criteria by which to filer
- **marker** Resource ID used by paging. The next page will start at the next resource after the marker
- **limit** Integer limit of objects on the page
- **sort_key** Key used to sort the returned list
- **sort_dir** Directions to sort after using sort_key

find_pool_nameserver(*context*, *criterion*)

find_pool_nameservers(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_pool_target(*context*, *criterion*)

find_pool_targets(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_pools(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find all Pools

Parameters

- **context** RPC Context.
- **criterion** Criteria by which to filter
- **marker** Resource ID used by paging. The next page will start at the next resource after the marker
- **limit** Integer limit of objects on the page
- **sort_key** Key used to sort the returned list
- **sort_dir** Directions to sort after using sort_key

find_quota(*context*, *criterion*)

Find a single Quota.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_quotas(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find Quotas

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_record(*context*, *criterion*)

Find a single Record.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_records(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find Records.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_recordset(*context*, *criterion*, *apply_tenant_criteria=True*)

Find a single RecordSet.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **apply_tenant_criteria** Whether to filter results by project_id.

find_recordsets(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*, *force_index=False*, *apply_tenant_criteria=True*)

Find RecordSets.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.
- **apply_tenant_criteria** Whether to filter results by project_id.

find_recordsets_axfr(*context*, *criterion=None*)

Find RecordSets.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_recordsets_export(*context*, *criterion=None*)

find_service_status(*context*, *criterion*)

Find a single Service Status.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_service_statuses(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Retrieve status for services

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_shared_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find shared zones

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after

- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_tenants(*context*)

Find all Tenants.

Parameters
context RPC Context.

find_tld(*context*, *criterion*)

Find a single TLD.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_tlds(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find TLDs

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_tsigkey(*context*, *criterion*)

Find TSIG Key.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_tsigkeys(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find TSIG Keys.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.

- **sort_dir** Direction to sort after using sort_key.

find_zone(*context*, *criterion*)

Find a single Zone.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_zone_attributes(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_export(*context*, *criterion*)

Find a single Zone Export.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_zone_exports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find Zone Exports

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_zone_import(*context*, *criterion*)

Find a single Zone Import.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.

find_zone_imports(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find Zone Imports

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker

- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

find_zone_transfer_accept(*context*, *criterion*)

find_zone_transfer_accepts(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zone_transfer_request(*context*, *criterion*)

find_zone_transfer_requests(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

find_zones(*context*, *criterion=None*, *marker=None*, *limit=None*, *sort_key=None*, *sort_dir=None*)

Find zones

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **marker** Resource ID from which after the requested page will start after
- **limit** Integer limit of objects of the page size after the marker
- **sort_key** Key from which to sort after.
- **sort_dir** Direction to sort after using sort_key.

get_blacklist(*context*, *blacklist_id*)

Get a Blacklist via ID.

Parameters

- **context** RPC Context.
- **blacklist_id** Blacklist ID to get.

get_catalog_zone(*context*, *pool*)

get_catalog_zone_records(*context*, *pool*)

get_inspector()

get_pool(*context*, *pool_id*)

Get a Pool via the id

Parameters

- **context** RPC Context.
- **pool_id** The ID of the pool to get

get_pool_also_notify(*context*, *pool_also_notify_id*)

get_pool_attribute(*context*, *pool_attribute_id*)

Get a PoolAttribute via the ID

Parameters

- **context** RPC Context.
- **pool_attribute_id** The ID of the PoolAttribute to get

get_pool_nameserver(*context*, *pool_nameserver_id*)

get_pool_target(*context*, *pool_target_id*)

get_quota(*context*, *quota_id*)

Get a Quota via ID.

Parameters

- **context** RPC Context.
- **quota_id** Quota ID to get.

get_record(*context*, *record_id*)

Get a record via ID

Parameters

- **context** RPC Context.
- **record_id** Record ID to get

get_shared_zone(*context*, *zone_id*, *shared_zone_id*)

Get a shared zone via ID

Parameters

- **context** RPC Context.
- **shared_zone_id** Shared Zone Id

get_tenant(*context*, *tenant_id*)

Get Tenant.

Parameters

- **context** RPC Context.
- **tenant_id** ID of the Tenant.

get_tld(*context*, *tld_id*)

Get a TLD via ID.

Parameters

- **context** RPC Context.
- **tld_id** TLD ID to get.

get_tsigkey(*context*, *tsigkey_id*)

Get a TSIG Key via ID.

Parameters

- **context** RPC Context.
- **tsigkey_id** Server ID to get.

get_zone(*context*, *zone_id*, *apply_tenant_criteria=True*)

Get a Zone via its ID.

Parameters

- **context** RPC Context.
- **zone_id** ID of the Zone.
- **apply_tenant_criteria** Whether to filter results by project_id.

get_zone_attributes(*context*, *zone_attribute_id*)

get_zone_export(*context*, *zone_export_id*)

Get a Zone Export via ID.

Parameters

- **context** RPC Context.
- **zone_export_id** Zone Export ID to get.

get_zone_import(*context*, *zone_import_id*)

Get a Zone Import via ID.

Parameters

- **context** RPC Context.
- **zone_import_id** Zone Import ID to get.

get_zone_transfer_accept(*context*, *zone_transfer_accept_id*)

get_zone_transfer_request(*context*, *zone_transfer_request_id*)

increment_serial(*context*, *zone_id*)

Increment the zones serial number.

is_zone_shared_with_project(*zone_id*, *project_id*)

Checks if a zone is shared with a project.

Parameters

- **zone_id** The zone ID to check.
- **project_id** The project ID to check.

Returns

Boolean True/False if the zone is shared with the project.

purge_zone(*context*, *zone*)

Purge a Zone, effectively removing the zone database record.

Parameters

- **context** RPC Context.
- **zone** Zone to delete.

purge_zones(*context*, *criterion*, *limit*)

Purge Zones, effectively removing the zones database records.

Reparent orphan childrens, if any. Transactions/locks are not needed.

Parameters

- **context** RPC Context.
- **criterion** Criteria to filter by.
- **limit** Integer limit of objects of the page size after the marker

share_zone(*context*, *shared_zone*)

Share zone

Parameters

- **context** RPC Context.
- **shared_zone** Shared Zone dict

unshare_zone(*context*, *zone_id*, *shared_zone_id*)

Unshare zone

Parameters

- **context** RPC Context.
- **shared_zone_id** Shared Zone Id

update_blacklist(*context*, *blacklist*)

Update a Blacklist

Parameters

- **context** RPC Context.
- **blacklist** Blacklist to update.

update_pool(*context*, *pool*)

Update the specified pool

Parameters

- **context** RPC Context.
- **pool** Pool to update.

update_pool_also_notify(*context*, *pool_also_notify*)

update_pool_attribute(*context*, *pool_attribute*)

Update the specified pool

Parameters

- **context** RPC Context.
- **pool_attribute** PoolAttribute to update

update_pool_nameserver(*context*, *pool_nameserver*)

update_pool_ns_record(*context*, *pool_ns_record*)

update_pool_target(*context*, *pool_target*)

update_pool_target_master(*context*, *pool_target_master*)

update_pool_target_option(*context*, *pool_target_option*)

update_quota(*context*, *quota*)

Update a Quota

Parameters

- **context** RPC Context.
- **quota** Quota to update.

update_record(*context*, *record*)

Update a record

Parameters

- **context** RPC Context.
- **record** Record to update

update_recordset(*context*, *recordset*)

Update a recordset

Parameters

- **context** RPC Context.
- **recordset** RecordSet to update

update_service_status(*context*, *service_status*)

Update the Service status for a service.

Parameters

- **context** RPC Context.
- **service_status** Set the status for a service.

update_tld(*context*, *tld*)

Update a TLD

Parameters

- **context** RPC Context.
- **tld** TLD to update.

update_tsigkey(*context*, *tsigkey*)

Update a TSIG Key

Parameters

- **context** RPC Context.
- **tsigkey** TSIG Keyto update.

update_zone(*context*, *zone*)

Update a Zone

Parameters

- **context** RPC Context.
- **zone** Zone object.

update_zone_attribute(*context*, *zone_attribute*)

update_zone_export(*context*, *zone_export*)

Update a Zone Export

Parameters

- **context** RPC Context.
- **zone_export** Zone Export to update.

update_zone_import(*context*, *zone_import*)

Update a Zone Import

Parameters

- **context** RPC Context.
- **zone_import** Zone Import to update.

update_zone_master(*context*, *zone_master*)

update_zone_transfer_accept(*context*, *zone_transfer_accept*)

update_zone_transfer_request(*context*, *zone_transfer_request*)

1.3.5 Development Environment on Ubuntu

Designate is comprised of four main components *Designate API*, *Designate Central*, designate-mdns, and designate-pool-manager, supported by a few standard open source components. For more information see *Architecture*.

There are many different options for customizing Designate, and two of these options have a major impact on the installation process:

- The storage backend used (SQLite or MySQL)
- The DNS backend used (PowerDNS or BIND9)

This guide will walk you through setting up a typical development environment for Designate, using BIND9 as the DNS backend and MySQL as the storage backend. For a more complete discussion on installation & configuration options, please see *Architecture*.

For this guide you will need access to an Ubuntu Server (16.04).

Development Environment

Installing Designate

1. Install system package dependencies (Ubuntu)

```
$ sudo apt update
$ sudo apt install python-pip python-virtualenv libssl-dev libffi-dev git
$ sudo apt build-dep python-lxml
```

2. Clone the Designate repo

```
$ mkdir openstack
$ cd openstack
$ git clone https://opendev.org/openstack/designate.git
$ cd designate
```

3. Setup a virtualenv

Note

This step is necessary to allow the installation of an up-to-date pip, independent of the version packaged for Ubuntu. it is also useful in isolating the remainder of Designates dependencies from the rest of the system.

```
$ virtualenv .venv
$ . .venv/bin/activate
```

4. Install an up-to-date pip

```
$ pip install -U pip
```

5. Install Designate and its dependencies

```
$ pip install -e .
```

6. Change directories to the etc/designate folder.

Note

Everything from here on out should take place in or below your etc/designate folder

```
$ cd etc/designate
```

7. Create Designates config files by copying the sample config files

```
$ cp -a rootwrap.conf.sample rootwrap.conf
```

8. Make the directory for Designates state files

```
$ mkdir -p ../../state
```

Configuring Designate

Refer to *Designate Configuration Guide* for a sample configuration options.

Installing RabbitMQ

Install the RabbitMQ package

```
$ sudo apt install rabbitmq-server
```

Create a user:

```
$ăsudo rabbitmqctl add_user designate designate
```

Give the user access to the / vhost:

```
$ăsudo rabbitmqctl set_permissions -p "/" designate ".*" ".*" ".*"
```

Installing MySQL

Install the MySQL server package

```
$ sudo apt install mysql-server
```

If you do not have MySQL previously installed, you will be prompted to change the root password. By default, the MySQL root password for Designate is password. You can:

- Change the root password to password
- If you want your own password, edit the designate.conf file and change any instance of mysql+pymysql://root:password@127.0.0.1/designate?charset=utf8 to mysql+pymysql://root:YOUR_PASSWORD@127.0.0.1/designate?charset=utf8

You can change your MySQL password anytime with the following command:

```
$ mysqladmin -u root -p password NEW_PASSWORD
Enter password <enter your old password>
```

Create the Designate tables

```
$ mysql -u root -p
Enter password: <enter your password here>

mysql> CREATE DATABASE `designate` CHARACTER SET utf8 COLLATE utf8_general_ci;
mysql> exit;
```

Install additional packages

```
$ sudo apt install libmysqlclient-dev
$ pip install pymysql
```

Installing BIND9

Install the DNS server, BIND9

```
$ sudo apt install bind9
```

Update the BIND9 Configuration

```
$ sudo editor /etc/bind/named.conf.options
```

Change the corresponding lines in the config file:

```
options {
  directory "/var/cache/bind";
  dnssec-validation auto;
  auth-nxdomain no; # conform to RFC1035
  listen-on-v6 { any; };
  allow-new-zones yes;
  request-ixfr no;
  recursion no;
};
```

Disable AppArmor for BIND9

```
$ sudo touch /etc/apparmor.d/disable/usr.sbin.named
$ sudo systemctl reload apparmor
```

Restart BIND9:

```
$ sudo systemctl restart bind9
```

Create and Import pools.yaml File

Create the pools.yaml file

```
$ editor pools.yaml
```

Copy or mirror the configuration from this sample file here:

```
- name: default
  # The name is immutable. There will be no option to change the name after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default BIND9 Pool

  attributes: {}

  # List out the NS records for zones hosted within this pool
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1

  # List out the nameservers for this pool. These are the actual BIND servers.
  # We use these to verify changes have propagated to all nameservers.
  nameservers:
    - host: 127.0.0.1
      port: 53

  # List out the targets for this pool. For BIND, most often, there will be␣
↪one
  # entry for each BIND server.
```

(continues on next page)

(continued from previous page)

```
targets:
  - type: bind9
    description: BIND9 Server 1

    # List out the designate-mdns servers from which BIND servers should
    # request zone transfers (AXFRs) from.
    masters:
      - host: 127.0.0.1
        port: 5354

    # BIND Configuration options
    options:
      host: 127.0.0.1
      port: 53
      rndc_host: 127.0.0.1
      rndc_port: 953
      rndc_key_file: /etc/bind/rndc.key

# Optional list of additional IP/Port's for which designate-mdns will send
# DNS NOTIFY packets to
# also_notifies:
#  - host: 192.0.2.4
#    port: 53
```

Initialize the Database

Sync the Designate database.

```
$ designate-manage database sync
```

Start the Central Service

Start the central service.

```
$ designate-central
```

Youll now be seeing the log from the central service.

Initialize Pools Information

Import the pools.yaml file into Designate. It is important that `designate-central` is started before invoking this command

```
$ designate-manage pool update --file pools.yaml
```

Start the other Services

Open up some new ssh windows and log in to your server (or open some new screen/tmux sessions).

```
$ cd openstack/designate
$ . .venv/bin/activate
```

Start the other services

```
$ designate-api
$ designate-mdns
$ designate-worker
$ designate-producer
```

Youll now be seeing the logs from the other services.

Exercising the API

Note

If you have a firewall enabled, make sure to open port 53, as well as Designates default port (9001).

Using a web browser, curl statement, or a REST client, calls can be made to the Designate API. You can find the various API calls on the api-ref document.

For example:

```
$ curl 127.0.0.1:9001/v2/zones -H 'Content-Type: application/json' --data '
  {
    "name": "example.com.",
    "email": "example@example.com"
  }'

{"status": "PENDING",.....
$ curl 127.0.0.1:9001/v2/zones
{"zones": [{"status": "ACTIVE",.....
```

The `ACTIVE` status shows that the zone propagated. So you should be able to perform a DNS query and see it:

```
$ dig @127.0.0.1 example.com SOA +short
ns1-1.example.org. example.example.com. 1487884120 3531 600 86400 3600
```

You can find the IP Address of your server by running

```
ip addr show eth0 | grep "inet\b" | awk '{print $2}' | cut -d/ -f1
```

If you have Keystone set up, you can use it by configuring the `[keystone_authtoken]` section and changing the `auth_strategy = keystone` in the `service:api` section. This will make it easier to use clients like the `openstack` CLI that expect Keystone.

1.3.6 OpenStack Integrations

This page overviews integrations with other services like Neutron and others to make use of Designate more convenient.

Reverse - FloatingIP

The FloatingIP PTR feature of Designate relies on information of the FloatingIP which is in a different service than Designate itself. It can be in any service as long as there is a plugin for it that can be loaded via the configuration setting called network_api.

- Controller, views and schemas in the V2 API
- RPC Client towards Central used by the API and Sink
- Logic in Central to make it convenient for setting, unsetting, listing and getting FloatingIP PTR records compared to the Records themselves which would be more work. (This is outlined in code docstrings for the specific methods.)
- Sink handlers for the various backend to help us be more consistent.

Record invalidation

Happens mainly happens via comparing a Tenants FloatingIPs towards the list we have of Records which are of a certain plugin type and with the use of a Sink handler that listens for incoming events from the various services.

Configuring Neutron

Configuring the FloatingIP feature is really simple:

```
[network_api:neutron]
endpoints = RegionOne|http://localhost:9696
endpoint_type = publicURL
timeout = 30
insecure = False
ca_certificates_file = /etc/path/to/ca.pem
```

Note that using admin_user, admin_password and admin_tenant_name is optional, if not present well piggyback on the context.auth_token passed in by the API.

Note

If endpoints is not configured and theres no service catalog is present in the context passed by the API to Central the request will fail in a NoEndpoint exception.

Neutron Designate direct integration

Neutron supports creating DNS Recordsets as neutron ports are created, and pushing that information into designate.

The configuration for this is in the Networking Guide

Designate Sink

Designate Sink is a component of designate that can listen to the event stream of other openstack services and perform actions based on them.

1.3.7 Other modules

1.4 User guide

In this section, you will find documentation relevant for using Designate.

Contents:

1.4.1 Managing Zones

Managing Zones

In the Domain Name System, *zones* are used to break up the namespace into more easily managed pieces. For example, within the root zone `.` there are zones for each of the top level domains such as `.org.` and `.com.` and responsiblity for each of those zones could lie with a different organisation. Within those zones, there are then delegations to other zones, such as `example.org.` or `example.com.` which might again be managed by a different organisation and/or set of nameservers. This forms a hierarchy of responsibility, with the higher levels being mainly composed of delegations to lower levels.

Zones in Designate

Zones in Designate model the ownership concept from DNS itself, where any given zone can only be owned by a single tenant. However, while DNS is able to support a hierarchy of zones, there is no support for delegating subzones to another tenant, and one tenant cannot create zones that lie within the zone of another tenant.

The creation of a zone in Designate also creates two recordsets automatically: an SOA record and an NS record. By default these records cannot be modified without the admin role.

Zones vs Top Level Domains

While top level domains are considered zones from a DNS perspective, in Designate they are often not managed as a zone, and instead have their own TLD type that allows any tenant to create zones within that TLD and restricts tenants from creating zones that arent within a managed TLD. If no TLDs are being managed within Designate, tenants can create any zone aside from the root zone and top level domains.

Creating a zone

Creating a zone requires only the name of the zone and an email address of the party responsible for the zone.

```
$ openstack zone create --email dnsmaster@example.com example.com.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| attributes     | {}                                   |
| created_at     | 2016-07-13T14:54:16.000000           |
```

(continues on next page)

(continued from previous page)

```
| description    | None                                 |
| email          | dnsmaster@example.com                |
| id             | 14093115-0f0f-497a-ac69-42235e46c26f |
| masters        |                                      |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 656bc359067844fba6005d400f19df76     |
| serial         | 1468421656                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+
```

Note that the state is PENDING. Designate has received the request to create the zone, but may not have completed it yet. After a short time, verify successful creation of the DNS Zone:

```
$ openstack zone list
+--------------------------------------+--------------+---------+------------
↪+--------+--------+
| id                                   | name         | type    |     serial␣
↪| status | action |
+--------------------------------------+--------------+---------+------------
↪+--------+--------+
| 14093115-0f0f-497a-ac69-42235e46c26f | example.com. | PRIMARY | 1468421656␣
↪| ACTIVE | NONE   |
+--------------------------------------+--------------+---------+------------
↪+--------+--------+
```

There will now be two recordsets visible in the zone:

```
$ openstack recordset list example.com.
+--------------------------------------+--------------+------+----------------
↪-----------------------------------------------------+--------+--------+
| id                                   | name         | type | records       ␣
↪                                                     | status | action |
+--------------------------------------+--------------+------+----------------
↪-----------------------------------------------------+--------+--------+
| 269cf8d2-c498-49a8-aef9-01e81d078313 | example.com. | SOA  | ns1.devstack.
↪org. admin.example.com. 1618291836 3509 600 86400 3600 | ACTIVE | NONE   |
| 31b50023-88b2-4011-b31b-474fa25a8e39 | example.com. | NS   | ns1.devstack.
↪org.                                                 | ACTIVE | NONE   |
+--------------------------------------+--------------+------+----------------
↪-----------------------------------------------------+--------+--------+
```

The values for refresh, retry, minimum and expire on the SOA record are set by the Designate operator. The TTL, however, can be modified by users via the zone:

```
$ openstack zone set example.com. --ttl 3000
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | UPDATE                               |
| attributes     |                                      |
| created_at     | 2021-04-13T05:30:36.000000           |
| description    | None                                 |
| email          | admin@example.com                    |
| id             | b9861a55-0e50-4896-8ab9-25d8c4494f64 |
| masters        |                                      |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 9d69e3a004aa40c581f00d7bb7763e0a     |
| serial         | 1618545015                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3000                                 |
| type           | PRIMARY                              |
| updated_at     | 2021-04-16T03:50:15.000000           |
| version        | 11                                   |
+----------------+--------------------------------------+
```

The `dig` tool can be used to query one of the backend nameservers to confirm the result. In this example, there is a DNS server at `192.168.122.186` managed by designate as part of the default pool.

```
$ dig @192.168.122.186 example.com.

; <<>> DiG 9.11.20-RedHat-9.11.20-5.el8_3.1 <<>> @192.168.122.186 example.com.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63663
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 970f584e4cb93505eaf46f526079097ac959da76062f1d0a (good)
;; QUESTION SECTION:
;example.com.                   IN      A

;; AUTHORITY SECTION:
example.com.            3000    IN      SOA     ns1.devstack.org. admin.example.
↪com. 1618545015 3509 600 86400 3600

;; Query time: 0 msec
;; SERVER: 192.168.122.186#53(192.168.122.186)
;; WHEN: Fri Apr 16 03:50:18 UTC 2021
;; MSG SIZE  rcvd: 126
```

In the AUTHORITY section, the numeric value between the name and *IN* is the TTL, which has updated to the new value of 3000.

Deleting a zone

A zone can be deleted using either its name or ID:

```
$ openstack zone delete example.com.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | DELETE                               |
| attributes     |                                      |
| created_at     | 2021-04-13T05:30:36.000000           |
| description    | None                                 |
| email          | admin@example.com                    |
| id             | b9861a55-0e50-4896-8ab9-25d8c4494f64 |
| masters        |                                      |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 9d69e3a004aa40c581f00d7bb7763e0a     |
| serial         | 1618545024                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3000                                 |
| type           | PRIMARY                              |
| updated_at     | 2021-04-16T10:18:05.000000           |
| version        | 15                                   |
+----------------+--------------------------------------+
```

Any records present in the zone are also deleted and will no longer resolve.

Note

Zones that have shares cannot be deleted without removing the shares or using the *delete-shares* modifier.

Associating a Zone with a Pool

When your administrator has configured designate to use multiple DNS server pools, it might be necessary for you to indicate a specific pool attribute or ID when you create a zone. Your administrator will provide you with the necessary pool information to create a zone.

In this example, the pool attribute that indicates one of several service tiers, must be specified when creating a zone:

```
$ openstack zone create --email dnsmaster@example.com example.com. --
↪attributes service_tier:silver
 +----------------+--------------------------------------+
 | Field          | Value                                |
 +----------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| action          | CREATE                                |
| attributes      | service_tier:silver                   |
|                 |                                       |
| created_at      | 2023-04-04T18:30:45.000000            |
| description     | None                                  |
| email           | dnsmaster@example.com                 |
| id              | d106e7b0-9973-41a1-b3db-0fb34b6d952c  |
| masters         |                                       |
| name            | example.com.                          |
| pool_id         | 10cec123-43f0-4b60-98a8-1204dd826c67  |
| project_id      | 5160768b59524fd283a4fa82d7327644      |
| serial          | 1674585045                            |
| status          | PENDING                               |
| transferred_at  | None                                  |
| ttl             | 3600                                  |
| type            | PRIMARY                               |
| updated_at      | None                                  |
| version         | 1                                     |
+-----------------+---------------------------------------+
```

Note

Remember that

[service:central] scheduler_filters = attribute

configuration setting is required to associate a newly created zone with an existing pool.

In this example, a specific pool ID, `7a2cde6b-d321-fa11-f99e-ccc378fe3dd1`, must be specified when creating a zone:

```
$ openstack zone create --email dnsmaster@example.com example.com. --
↪attributes pool_id:7a2cde6b-d321-fa11-f99e-ccc378fe3dd1
+-----------------+-----------------------------------------------+
| Field           | Value                                         |
+-----------------+-----------------------------------------------+
| action          | CREATE                                        |
| attributes      | pool_id:7a2cde6b-d321-fa11-f99e-ccc378fe3dd1  |
|                 |                                               |
| created_at      | 2023-04-04T18:39:12.000000                    |
| description     | None                                          |
| email           | dnsmaster@example.com                         |
| id              | 54f2bcaa-65ef-8274-5fde-987234508afe          |
| masters         |                                               |
| name            | example.com.                                  |
| pool_id         | 7a2cde6b-d321-fa11-f99e-ccc378fe3dd1          |
| project_id      | 5160768b59524fd283a4fa82d7327644              |
| serial          | 2385822109                                    |
| status          | PENDING                                       |
```

(continues on next page)

(continued from previous page)

```
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+
```

Note

Remember that

[service:central] scheduler_filters = pool_id_attribute

configuration setting is required to associate a newly created zone with an existing pool.

Verify that the zone has been created:

```
$ openstack zone list
+--------------------------------------+---------------+---------+------------
↪+--------+--------+
| id                                   | name          | type    |     serial␣
↪| status | action |
+--------------------------------------+---------------+---------+------------
↪+--------+--------+
| 54f2bcaa-65ef-8274-5fde-987234508afe | example.com.  | PRIMARY | 2385822109␣
↪| ACTIVE | NONE   |
+--------------------------------------+---------------+---------+------------
↪+--------+--------+
```

Zone Import and Export

Overview

Zones can be imported into and serialised out of Designate using the zone import and export APIs. Using the zone file format along with these APIs you can both create zones and recordsets in batches and export zone data from Designate easily.

Exporting Zones

You can export a zone file from Designate using the *zone export create* subcommand on an existing zone, and subsequently access the exported zone file using *zone export showfile*.

For example, use *openstack recordset list* to view the records for a zone youd like to export:

```
$ openstack recordset list example.org.
+--------------------------------------+-----------------+------+------------
↪-----------------------------------------------------+--------+--------+
| id                                   | name            | type | records   ␣
↪                                                     | status | action |
+--------------------------------------+-----------------+------+------------
```

(continues on next page)

(continued from previous page)

```
↪-------------------------------------------------------+--------+--------+
| b4dfeb36-c4ae-4399-9493-6e6997099356 | example.org.     | NS   | ns1.
↪example.org.                                                | ACTIVE |␣
↪NONE   |
| e9e3b31f-8aef-465f-9380-e3380191f8bd | example.org.     | SOA  | ns1.
↪example.org. admin.example.org. 1624414033 3583 600 86400 3600  | ACTIVE |␣
↪NONE   |
| 09407eaa-1fac-4257-b9e1-11d693bc1eae | www.example.org. | A    | 192.0.2.2 ␣
↪                                                      | ACTIVE | NONE   |
|                                      |                  |      | 192.0.2.1 ␣
↪                                                      |        |        |
+--------------------------------------+------------------+------+------------
↪-------------------------------------------------------+--------+--------+
```

Using the *openstack zone export create* command, export *example.org.*:

```
$ openstack zone export create example.org.
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| created_at | 2021-06-23T02:01:30.000000           |
| id         | e75aef2c-b562-4cd9-a426-4a73f6cb82be |
| location   | None                                 |
| message    | None                                 |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| status     | PENDING                              |
| updated_at | None                                 |
| version    | 1                                    |
| zone_id    | d8f81db6-937b-4388-bfb3-ba620e6c09fb |
+------------+--------------------------------------+
```

You can access the contents of the zone file using *zone export showfile*. Using the *-f value* parameter will print the contents of the zone file without any tabulation, which can be useful if you want to modify the exported zone file locally and then import it back into Designate to update the zone.

```
$ openstack zone export showfile e75aef2c-b562-4cd9-a426-4a73f6cb82be -f value
$ORIGIN example.org.
$TTL 3600

example.org.  IN NS ns1.example.org.
example.org.  IN SOA ns1.example.org. admin.example.org. 1624414033 3583 600␣
↪86400 3600

www.example.org.  IN A 192.0.2.2
www.example.org.  IN A 192.0.2.1
```

By default, the zone export file will be created on demand as it is accessed and as a result the contents of the zone export file will be updated as you add new recordsets to the zone:

```
$ openstack recordset create example.org. test --type A --record 192.0.2.100
```

(continues on next page)

(continued from previous page)

```
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2021-06-23T02:35:06.000000           |
| description | None                                 |
| id          | aa27ccd8-77b1-41df-a3ed-2129259b334a |
| name        | test.example.org.                    |
| project_id  | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| records     | 192.0.2.100                          |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | d8f81db6-937b-4388-bfb3-ba620e6c09fb |
| zone_name   | example.org.                         |
+-------------+--------------------------------------+
$ openstack zone export showfile e75aef2c-b562-4cd9-a426-4a73f6cb82be -f value
$ORIGIN example.org.
$TTL 3600

example.org.  IN NS ns1.example.org.
example.org.  IN SOA ns1.example.org. admin.example.org. 1624415706 3583 600␣
↪86400 3600
www.example.org.  IN A 192.0.2.2
www.example.org.  IN A 192.0.2.1
test.example.org.  IN A 192.0.2.100
```

Zone Export Internals

The zone export resource created does not contain the zone file data, instead it holds the location of that data as Designate can be configured by the operator to store zone exports in external services. By default, the location of the zone export file is internal to Designate and uses the Designate protocol *designate://*. In this case, zone file data will be generated on demand when *zone export showfile* is used. You can view the location URI of the zone file data using *zone export show*:

```
$ openstack zone export show e75aef2c-b562-4cd9-a426-4a73f6cb82be
+------------+-------------------------------------------------------------------
↪----------------+
| Field      | Value                                                             ␣
↪                |
+------------+-------------------------------------------------------------------
↪----------------+
| created_at | 2021-06-23T02:01:30.000000                                        ␣
↪                |
| id         | e75aef2c-b562-4cd9-a426-4a73f6cb82be                              ␣
↪                |
| location   | designate://v2/zones/tasks/exports/e75aef2c-b562-4cd9-a426-
```

(continues on next page)

(continued from previous page)

```
↪4a73f6cb82be/export |
| message    | None                                                              ␣
↪                 |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7                                  ␣
↪                 |
| status     | COMPLETE                                                          ␣
↪                 |
| updated_at | 2021-06-23T02:01:30.000000                                        ␣
↪                 |
| version    | 2                                                                 ␣
↪                 |
| zone_id    | d8f81db6-937b-4388-bfb3-ba620e6c09fb                              ␣
↪                 |
+------------+-------------------------------------------------------------------
↪----------------+
```

Zone Import

You can import a zone and all of its recordsets by putting them all into a file that uses the zone file format and calling *openstack zone import create*:

```
$ cat zone_file
$ORIGIN example.org.
$TTL 3600

example.org.  IN NS ns1.example.org.
example.org.  IN SOA ns1.example.org. admin.example.org. 1624415706 3583 600␣
↪86400 3600
www.example.org.  IN A 192.0.2.2
www.example.org.  IN A 192.0.2.1
test.example.org.  IN A 192.0.2.100

$ openstack zone import create zone_file
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| created_at | 2021-06-24T03:39:58.000000           |
| id         | 6140580d-c72a-4f07-82ab-908da979a9a3 |
| message    | None                                 |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| status     | PENDING                              |
| updated_at | None                                 |
| version    | 1                                    |
| zone_id    | None                                 |
+------------+--------------------------------------+
```

You can now view the zone in Designate:

```
$ openstack recordset list example.org.
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| id                                   | name             | type | records  ␣
↪                                                           | status | action␣
↪|
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| 3d9e96c2-da27-4c5b-9b2b-c1b44a58c1e5 | www.example.org. | A    | 192.0.2.2␣
↪                                                           | ACTIVE | NONE  ␣
↪|
|                                      |                  |      | 192.0.2.1␣
↪                                                          |        |       ␣
↪|
| 541bac15-18da-411f-a8e5-8ccecb65ae1f | example.org.     | SOA  | ns1.
↪example.org. admin.example.org. 1624415706 3541 600 86400 3600  | ACTIVE |␣
↪NONE   |
| a643b088-6052-49c0-81f7-6ade6682d9a3 | example.org.     | NS   | ns1.
↪example.org.                                                   | ACTIVE |␣
↪NONE   |
| f97274f1-e062-4f59-8ec0-11bccd830547 | test.example.org. | A    | 192.0.2.
↪100                                                          | ACTIVE | NONE␣
↪  |
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
```

You cannot use zone imports to update a zone or create records in a zone that already exists. Importing a zone that already exists will result in an error and no records will be created or modified.

```
$ echo "new.example.org. IN A 192.0.2.101" >> zone_file
$ openstack zone import create zone_file
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| created_at | 2021-06-24T03:40:28.000000           |
| id         | 50516762-23ec-4bf3-a065-530171c5d0fb |
| message    | None                                 |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| status     | PENDING                              |
| updated_at | None                                 |
| version    | 1                                    |
| zone_id    | None                                 |
+------------+--------------------------------------+
$ openstack zone import show 50516762-23ec-4bf3-a065-530171c5d0fb
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| created_at | 2021-06-24T03:40:28.000000           |
```

(continues on next page)

(continued from previous page)

```
| id         | 50516762-23ec-4bf3-a065-530171c5d0fb |
| message    | An undefined error occurred.         |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| status     | ERROR                                |
| updated_at | 2021-06-24T03:40:28.000000           |
| version    | 2                                    |
| zone_id    | None                                 |
+------------+--------------------------------------+
$ openstack recordset list example.org.
+--------------------------------------+--------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| id                                   | name               | type | records  ␣
↪                                                            | status | action␣
↪|
+--------------------------------------+--------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| 3d9e96c2-da27-4c5b-9b2b-c1b44a58c1e5 | www.example.org.   | A    | 192.0.2.2␣
↪                                                            | ACTIVE | NONE  ␣
↪|
|                                      |                    |      | 192.0.2.1␣
↪                                                            |        |       ␣
↪|
| 541bac15-18da-411f-a8e5-8ccecb65ae1f | example.org.       | SOA  | ns1.
↪example.org. admin.example.org. 1624415706 3541 600 86400 3600  | ACTIVE |␣
↪NONE   |
| a643b088-6052-49c0-81f7-6ade6682d9a3 | example.org.       | NS   | ns1.
↪example.org.                                                    | ACTIVE |␣
↪NONE   |
| f97274f1-e062-4f59-8ec0-11bccd830547 | test.example.org.  | A    | 192.0.2.
↪100                                                           | ACTIVE | NONE␣
↪  |
+--------------------------------------+--------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
```

You must set the zone TTL using a TTL statement in the zone tile. The SOA record created for the zone will not always match the values in the zone file as some values are dependent on Designate configuration options:

- The *MNAME* is set using the zones assigned pool information.
- The refresh value is set randomly between the `default_soa_refresh_min` and `default_soa_refresh_max` configuration values.
- The minimum value is set to the `soa_default_minimum` configuration value.

The NS record for the zone is generated based on the pool the zone has been assigned. Other NS records are imported without modification.

For example, the following zone file uses *test.example.org.* as its namserver, and provides its own values for the zone TTL, refresh, minimum and expire. The refresh and minimum values will be discarded on

import and the nameserver changed to the pools nameserver at *ns1.example.org.*:

```
$ cat zone_file
$ORIGIN example.org.
$TTL 3000

example.org.  IN NS test.example.org.
example.org.  IN SOA test.example.org. admin.example.org. 1624415706 9000 500␣
↪86000 5000
www.example.org.  IN A 192.0.2.2
test.example.org.  IN NS test.example.org.
$ openstack zone import create zone_file
+------------+--------------------------------------+
| Field      | Value                                |
+------------+--------------------------------------+
| created_at | 2021-06-25T07:07:41.000000           |
| id         | ccd0af00-aa5f-43e0-a57d-67cfa2f3738e |
| message    | None                                 |
| project_id | cf5a8f5cc5834d2dacd1d54cd0a354b7     |
| status     | PENDING                              |
| updated_at | None                                 |
| version    | 1                                    |
| zone_id    | None                                 |
+------------+--------------------------------------+
$ openstack recordset list example.org.
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| id                                   | name             | type | records  ␣
↪                                                           | status | action␣
↪|
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
| 35143297-5268-4bc9-80bb-9d2d12c609e0 | example.org.     | SOA  | ns1.
↪example.org. admin.example.org. 1624415706 3582 500 86000 3600  | ACTIVE |␣
↪NONE   |
| 3532dee3-effc-4aac-b5c4-90b6e2ad20e0 | test.example.org. | NS   | test.
↪example.org.                                                    | ACTIVE |␣
↪NONE   |
| bef04729-f49e-4920-83b6-2ef9b620fa9d | example.org.     | NS   | ns1.
↪example.org.                                                    | ACTIVE |␣
↪NONE   |
| c290d79a-6583-4666-a6f7-d4b967f67d79 | www.example.org.  | A    | 192.0.2.2␣
↪                                                           | ACTIVE | NONE  ␣
↪|
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------
↪+
```

Zone Ownership Transfers

Designate allows you to transfer ownership of zones between projects. For example, the engineering team project may want to transfer the ownership of the wow.example.com. zone from the engineering project to the marketing teams project.

This can be accomplished without cloud administrator intervention using the zone transfer features in Designate. Both the sending and receiving project must agree to the transfer by using the zone transfer process.

Zone Transfer Requests

Creating a Zone Transfer Request

To create a zone transfer offer we create a zone transfer request in Designate. You can optionally provide a target project ID in the request to lock the transfer to a specific project. When using a target project ID, no other project will be allowed to accept the zone transfer. If you do not provide a target project ID, any project that has the transfer request ID and key can receive the zone transfer.

Note

The target project ID must be provided as the `ID` and not the project name.

To transfer the zone wow.example.com. to project 1d12e87fad0d437286c2873b36a12316 you would run:

```
$ openstack zone transfer request create --target-project-id␣
↪1d12e87fad0d437286c2873b36a12316 wow.example.com.

+------------------+--------------------------------------------------------+
| Field            | Value                                                  |
+------------------+--------------------------------------------------------+
| created_at       | 2022-05-26T22:06:39.000000                             |
| description      | None                                                   |
| id               | 63cab5e5-65fa-4480-b26c-c16c267c44b2                   |
| key              | BIFJIQWH                                               |
| links            | {'self': 'http://127.0.0.1:60053/v2/zones/tasks/tra |
|                  | nsfer_requests/63cab5e5-65fa-4480-b26c-c16c267c44b2 |
|                  | '}                                                     |
| project_id       | 6265985fc493465db6a978b318a01996                       |
| status           | ACTIVE                                                 |
| target_project_id | 1d12e87fad0d437286c2873b36a12316                       |
| updated_at       | None                                                   |
| zone_id          | 962f08b4-b671-4096-bf24-8908c9d4af0c                   |
| zone_name        | wow.example.com.                                       |
+------------------+--------------------------------------------------------+
```

You will then provide the ID and key to a member of the receiving project.

Displaying a Zone Transfer Request

To display the zone transfer request we created in the previous section you would run:

```
$ openstack zone transfer request show 63cab5e5-65fa-4480-b26c-c16c267c44b2

+------------------+--------------------------------------------------------+
| Field            | Value                                                  |
+------------------+--------------------------------------------------------+
| created_at       | 2022-05-26T22:06:39.000000                             |
| description      | None                                                   |
| id               | 63cab5e5-65fa-4480-b26c-c16c267c44b2                   |
| key              | BIFJIQWH                                               |
| links            | {'self': 'http://127.0.0.1:60053/v2/zones/tasks/tra |
|                  | nsfer_requests/63cab5e5-65fa-4480-b26c-c16c267c44b2 |
|                  | '}                                                     |
| project_id       | 6265985fc493465db6a978b318a01996                       |
| status           | ACTIVE                                                 |
| target_project_id | 1d12e87fad0d437286c2873b36a12316                       |
| updated_at       | None                                                   |
| zone_id          | 962f08b4-b671-4096-bf24-8908c9d4af0c                   |
| zone_name        | wow.example.com.                                       |
+------------------+--------------------------------------------------------+
```

Listing Zone Transfer Requests

You can list all of the existing zone transfer requests by using the *openstack zone transfer request list* command:

```
$ openstack zone transfer request list

+----------+----------+----------+-----------+------------------+--------+-
↪---------+
| id       | zone_id  | zone_name | project_id | target_project_id | status |␣
↪key      |
+----------+----------+----------+-----------+------------------+--------+-
↪---------+
| 63cab5e5 | 962f08b4 | wow.examp | 6265985fc4 | 1d12e87fad0d43728 | ACTIVE |␣
↪BIFJIQWH |
| -65fa-44 | -b671-40 | le.com.   | 93465db6a9 | 6c2873b36a12316   |        |␣
↪         |
| 80-b26c- | 96-bf24- |           | 78b318a019 |                   |        |␣
↪         |
| c16c267c | 8908c9d4 |           | 96         |                   |        |␣
↪         |
| 44b2     | af0c     |           |            |                   |        |␣
↪         |
+----------+----------+----------+-----------+------------------+--------+-
↪---------+
```

Updating a Zone Transfer Request

Designate allows you to update a limited set of fields on zone transfer requests, such as the description and target project ID.

To add a description the zone transfer request we created above, you would run the following command:

```
$ openstack zone transfer request set --description "wow zone transfer"␣
↪63cab5e5-65fa-4480-b26c-c16c267c44b2

+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| created_at       | 2022-05-26T22:06:39.000000                           |
| description      | wow zone transfer                                    |
| id               | 63cab5e5-65fa-4480-b26c-c16c267c44b2                 |
| key              | BIFJIQWH                                             |
| links            | {'self': 'http://127.0.0.1:60053/v2/zones/tasks/tra  |
|                  | nsfer_requests/63cab5e5-65fa-4480-b26c-c16c267c44b2  |
|                  | '}                                                   |
| project_id       | 6265985fc493465db6a978b318a01996                     |
| status           | ACTIVE                                               |
| target_project_id | 1d12e87fad0d437286c2873b36a12316                     |
| updated_at       | 2022-05-27T20:52:08.000000                           |
| zone_id          | 962f08b4-b671-4096-bf24-8908c9d4af0c                 |
| zone_name        | wow.example.com.                                     |
+------------------+------------------------------------------------------+
```

Deleting a Zone Transfer Request

If you would like to cancel a zone transfer you can delete the zone transfer request using the *openstack zone transfer request delete* command:

```
$ openstack zone transfer request delete 63cab5e5-65fa-4480-b26c-c16c267c44b2
```

There is no output from the zone transfer request delete command.

Zone Transfer Accepts

Accepting a Zone Transfer Request

Once you have the zone transfer request ID and key, you can create a *zone transfer accept* to finish the zone transfer.

An example of accepting the zone transfer we created in the *Zone Transfer Requests* section:

```
$ openstack zone transfer accept request --transfer-id 63cab5e5-65fa-4480-
↪b26c-c16c267c44b2 --key BIFJIQWH

+-------------------------+------------------------------------------------+
| Field                   | Value                                          |
+-------------------------+------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| created_at               | 2022-05-27T21:37:43.000000                    |
| id                       | a4c4f872-c98c-411b-a787-58ed0e2dce11          |
| key                      | BIFJIQWH                                      |
| links                    | {'self': 'http://127.0.0.1:60053/v2/zones/ta  |
|                          | sks/transfer_accepts/a4c4f872-c98c-411b-a787  |
|                          | -58ed0e2dce11', 'zone': 'http://127.0.0.1:60  |
|                          | 053/v2/zones/962f08b4-b671-4096-bf24-8908c9d  |
|                          | 4af0c'}                                       |
| project_id               | 1d12e87fad0d437286c2873b36a12316              |
| status                   | COMPLETE                                      |
| updated_at               | 2022-05-27T21:37:43.000000                    |
| zone_id                  | 962f08b4-b671-4096-bf24-8908c9d4af0c          |
| zone_transfer_request_id | 63cab5e5-65fa-4480-b26c-c16c267c44b2          |
+--------------------------+-----------------------------------------------+
```

Displaying a Zone Transfer Accept

To check the status of your zone transfer accept, you can use the *openstack zone transfer accept* command:

```
$ openstack zone transfer accept show a4c4f872-c98c-411b-a787-58ed0e2dce11

+--------------------------+-----------------------------------------------+
| Field                    | Value                                         |
+--------------------------+-----------------------------------------------+
| created_at               | 2022-05-27T21:37:43.000000                    |
| id                       | a4c4f872-c98c-411b-a787-58ed0e2dce11          |
| key                      | None                                          |
| links                    | {'self': 'http://127.0.0.1:60053/v2/zones/ta  |
|                          | sks/transfer_accepts/a4c4f872-c98c-411b-a787  |
|                          | -58ed0e2dce11', 'zone': 'http://127.0.0.1:60  |
|                          | 053/v2/zones/962f08b4-b671-4096-bf24-8908c9d  |
|                          | 4af0c'}                                       |
| project_id               | 1d12e87fad0d437286c2873b36a12316              |
| status                   | COMPLETE                                      |
| updated_at               | 2022-05-27T21:37:43.000000                    |
| zone_id                  | 962f08b4-b671-4096-bf24-8908c9d4af0c          |
| zone_transfer_request_id | 63cab5e5-65fa-4480-b26c-c16c267c44b2          |
+--------------------------+-----------------------------------------------+
```

Listing Zone Transfer Accepts

Designate can provide a list of existing zone transfer accept records using the *openstack zone transfer accept list* command:

Note

By default, only users with the admin role can list zone transfer accept records.

```
$ openstack zone transfer accept list

+------------+------------+------------+--------------------------+--------
↪--+-----+
| id         | zone_id    | project_id | zone_transfer_request_id | status␣
↪  | key |
+------------+------------+------------+--------------------------+--------
↪--+-----+
| a4c4f872-c9 | 962f08b4-b6 | 1d12e87fad0 | 63cab5e5-65fa-4480-b26c- |␣
↪COMPLETE |     |
| 8c-411b-a78 | 71-4096-bf2 | d437286c287 | c16c267c44b2             |      ␣
↪  |     |
| 7-58ed0e2dc | 4-8908c9d4a | 3b36a12316  |                          |      ␣
↪  |     |
| e11         | f0c         |             |                          |      ␣
↪  |     |
+------------+------------+------------+--------------------------+--------
↪--+-----+
```

Secondary Zones

The Designate v2 API introduced functionality that allows Designate to act as a DNS slave, rather than a master for a zone. This is accomplished by completing a zone transfer (AXFR) from a DNS server managed outside of Designate.

RecordSets / Records

Changes to secondary zones are managed outside of Designate. Users must make the changes they wish, and prompt a fresh zone transfer (AXFR) into Designate to make those changes live on any DNS servers Designate manages.

Setup

To add a secondary zone to Designate, there must be a DNS master for the zone, to which Designate can act as a slave. For this guide, we assume that you have already set this up.

The remaining Designate set up will be similar to a non-secondary zone setup. Youll need a primary DNS server for Designate to manage and transfer secondary zones to.

In our examples well use the following values:

Name - example.com.

Masters - 192.168.27.100

Setup - example NSD4

Skip this section if you have a master already to use.

Note

For this it is assumed that you are running on Ubuntu.

Install

For some reason theres a bug with the nsd package so it doesnt create the user that it needs for the installation. So well create that before installing the package.

```
$ sudo apt-get install nsd
```

Configure

```
$ăsudo zcat /usr/share/doc/nsd/examples/nsd.conf.sample.gz >/tmp/nsd.conf
$ăsudo mv /tmp/nsd.conf /etc/nsd/nsd.conf
```

Add the following to /etc/nsd/nsd.conf

Note

If youre wondering why we set notify to *192.168.27.100*:*5354* its because MDNS runs on 5354 by default.

```
$ăsudo vi /etc/nsd/nsd.conf
```

Add the contents:

```
pattern:
    name: "mdns"
    zonefile: "%s.zone"
    notify: 192.168.27.100@5354 NOKEY
    provide-xfr: 192.168.27.100 NOKEY
    allow-axfr-fallback: yes
```

Add a zone file

Create a new *Zone* in NSD called *example.com.*

/etc/nsd/example.com.zone

```
$ăsudo vi /etc/nsd/example.com.zone
```

And add the contents:

```
$TTL 1800 ;minimum ttl
example.com.         IN      SOA     ns1.example.com. admin.example.net. (
                        2014111301      ;serial
                        3600            ;refresh
                        600             ;retry
                        180000          ;expire
                        600             ;negative ttl
                        )

                TXT             "v=spf1 +a +mx ~all"
```

(continues on next page)

(continued from previous page)

```
                SPF             "v=spf1 +a +mx ~all"

                NS              ns1.example.com.
                NS              ns2.example.com.
                NS              ns3.example.com.

                MX      0       mail1.example.com.
                MX      5       mail2.example.com.
                MX      10      mail3.example.com.

                A               10.0.0.1
                A               10.0.0.2
                A               10.0.0.3


ns1             A               172.16.28.100
ns2             A               172.16.28.101
ns3             A               172.16.28.103

mail1             A               10.0.10.1
mail2             A               10.0.10.2
mail3             A               10.0.10.3

google          CNAME           google.com.
```

Restart NSD

```
$ sudo service nsd restart
```

Check that its working

```
$ sudo nsd-control status
```

Activate the zone in NSD

```
$ sudo nsd-control addzone example.com mdns
```

Creating the Zone

When you create a domain in Designate there are two possible initial actions:

- Domain is created but transfer fails if its not available yet in master, then typically the initial transfer will be done once the master sends first NOTIFY.
- Domain is created and transfers straight away.

In both cases the interaction between your master and Designate is handled by the MDNS instance at the Designate side.

Definition of values:

- *email* set to the value of the *managed_resource_email* option in the *central* section of the Designate configuration.
- *transferred_at* is **null** and *version* is **1** since the zone has not transferred yet.

```
$ openstack zone create --type secondary --masters 192.168.27.100 example.com.
```

Shared Zones

Shared zones allow sharing a particular zone across tenants. This is useful in cases when records for one zone should be managed by multiple projects. For example when a Designate zone is assigned to a shared network in Neutron.

Zone shares have the following properties:

- Quotas will be enforced against the zone owner.
- Projects that a zone is shared with can only manage recordsets created or owned by the project.
- Zone owners can see, modify, and remove recordsets created by another project.
- Projects that a zone is shared with cannot see or modify the attributes of the zone.
- Zones that have shares cannot be deleted without removing the shares or using the *delete-shares* modifier.
- Projects that a zone is shared with cannot create sub-zones.

How to Share a Zone With Another Project

Create a zone to share:

```
$ openstack zone create example.com. --email admin@example.com
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| email          | admin@example.com                    |
| id             | 92b2214f-8a57-4ed3-95f0-a64099f3b516 |
| name           | example.com.                         |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | 804806ad94364aecb0f9ae86ad653055     |
| serial         | 1596186919                           |
| status         | PENDING                              |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
+----------------+--------------------------------------+
```

Share the zone using the *openstack zone share create* command (in this example, the ID of the project we want to share with is *356df8e6c7564b5bb107f5de26cdb8ea*):

```
$ openstack zone share create example.com. 356df8e6c7564b5bb107f5de26cdb8ea
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| created_at        | 2023-01-30T23:17:44.000000           |
| id                | 77e4d5b9-2057-4be7-8cf0-9f84ef0efec1 |
| project_id        | 804806ad94364aecb0f9ae86ad653055     |
| target_project_id | 356df8e6c7564b5bb107f5de26cdb8ea     |
| updated_at        | None                                 |
| zone_id           | 92b2214f-8a57-4ed3-95f0-a64099f3b516 |
+-------------------+--------------------------------------+
```

Project *356df8e6c7564b5bb107f5de26cdb8ea* now has access to zone *92b2214f-8a57-4ed3-95f0-a64099f3b516* and can manage recordsets in the zone.

Using credentials for project *356df8e6c7564b5bb107f5de26cdb8ea*, we can create a recordset for *www.example.com.*:

```
$ openstack recordset create --type A --record 192.0.2.1 example.com. www
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2023-01-30T23:28:05.000000           |
| description | None                                 |
| id          | aff3e00a-9e5c-4cfa-9650-65196f73418b |
| name        | www.example.com.                     |
| project_id  | 356df8e6c7564b5bb107f5de26cdb8ea     |
| records     | 192.0.2.1                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 92b2214f-8a57-4ed3-95f0-a64099f3b516 |
| zone_name   | example.com.                         |
+-------------+--------------------------------------+
```

How to List All of the Projects Sharing a Zone

You can list all of the zone shares for a zone with the *openstack zone share list* command:

```
$ openstack zone share list example.com.
+----------------------+----------------------+--------------------------+
| id                   | zone_id              | target_project_id        |
+----------------------+----------------------+--------------------------+
| 77e4d5b9-2057-4be7-  | 92b2214f-8a57-4ed3-  | 356df8e6c7564b5bb107f5d |
| 8cf0-9f84ef0efec1    | 95f0-a64099f3b516    | e26cdb8ea                |
+----------------------+----------------------+--------------------------+
```

How To Remove a Zone Share

To stop sharing a zone with a project, you can use the *openstack zone share delete* command:

```
$ openstack zone share delete example.com. 77e4d5b9-2057-4be7-8cf0-
↪9f84ef0efec1
```

A zone cannot be unshared in the following cases:

- Zone has recordsets in other projects.

1.4.2 Working with Recordsets

Managing Records

While zones are used to break up the DNS namespace into a hierarchy, resource records, or simply records, are used to store data within the namespace. Each record has a

- **Name:** the string that indicates its location in the DNS namespace.
- **Type:** the set of letter codes that identify the records usage. For example `A` for an address record or `CNAME` for a canonical name record.
- **Class:** the set of letter codes that specify the namespace for the record. Typically, this is `IN` for internet, though other namespaces do exist.
- **TTL:** the duration in seconds that the record remains valid.
- **Rdata:** the data for the record, such as an IP address for an `A` type record or another record name for a `CNAME` type record.

Recordsets in Designate

DNS records in Designate are managed using `Recordsets`, which represent one or more DNS records with the same *Name* and *Type*, but potentially different data. For example, a recordset named `www.example.com`, with a type of `A`, that contains the data `192.0.2.1` and `192.0.2.2` might reflect two web servers hosting `www.example.com` located at those two IP addresses.

You must create Recordsets within a zone. If you delete a zone that contains recordsets, those recordsets within the zone are also deleted.

Creating a recordset

By default, any user can create Recordsets in zones that their project owns. In this example, a user has created a zone named `example.org.`.

Recordsets are created using the `openstack recordset create` command and require a zone, a name, a type, and data for the record. To recreate the earlier example using the OpenStack client with the Designate plugin, the user would run:

```
$ openstack recordset create --type A --record 192.0.2.1 example.org. www
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2021-05-03T03:13:46.000000           |
```

(continues on next page)

(continued from previous page)

```
| description | None                                 |
| id          | 549c3e83-443f-474b-b467-6bcd7cb9f37d |
| name        | www.example.org.                     |
| project_id  | c85fdba96041438fa0cad2dc7909d3f5     |
| records     | 192.0.2.1                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 077460ef-34db-486a-8d59-c9564dc3a3a9 |
| zone_name   | example.org.                         |
+-------------+--------------------------------------+
```

As the final argument `www` is not a fully qualified domain name (FQDN) it is prepended to the zone name. You can achieve the same result using the FQDN, `www.example.org.`. Note that the trailing `.` is required when using the FQDN. Omitting it results in the name, "`www.example.org.example.org.`".

You can supply the `--record` argument multiple times to create multiple records within the recordset. A typical use for this is Round-robin DNS.

```
$ openstack recordset create --type A --record 192.0.2.1 --record 192.0.2.␣
↪example.org. web
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2021-05-03T03:26:43.000000           |
| description | None                                 |
| id          | 9e0fba43-ca67-44ed-b9d9-fc1242920319 |
| name        | web.example.org.                     |
| project_id  | c85fdba96041438fa0cad2dc7909d3f5     |
| records     | 192.0.2.1                            |
|             | 192.0.2.2                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 077460ef-34db-486a-8d59-c9564dc3a3a9 |
| zone_name   | example.org.                         |
+-------------+--------------------------------------+
```

You can view the recordsets for a zone using the `openstack recordset list` command:

```
$ openstack recordset list example.org.
+--------------------------------------+------------------+------+------------
↪-----------------------------------------------------------+--------+--------+
| id                                   | name             | type | records   ␣
↪                                                            | status | action |
```

(continues on next page)

(continued from previous page)

```
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------+
| 3bebbd03-07d7-4274-a784-39c32a2be8c6 | example.org.     | SOA  | ns1.
↪example.net. admin.example.org. 1620012616 3599 600 86400 3600  | ACTIVE |␣
↪NONE   |
| 7d34e4d3-a2f1-4af0-831c-ba52a8312c6a | example.org.     | NS   | ns1.
↪example.net.                                                    | ACTIVE |␣
↪NONE   |
| 9e0fba43-ca67-44ed-b9d9-fc1242920319 | web.example.org. | A    | 192.0.2.1 ␣
↪                                                          | ACTIVE | NONE   |
|                                      |                  |      | 192.0.2.2 ␣
↪                                                         |        |        |
| 549c3e83-443f-474b-b467-6bcd7cb9f37d | www.example.org. | A    | 192.0.2.1 ␣
↪                                                          | ACTIVE | NONE   |
+--------------------------------------+------------------+------+-----------
↪-----------------------------------------------------------+--------+--------+
```

The `SOA` and `NS` records for the zone are also visible here, but cannot be modified.

The authoritative nameserver for the zone is listed as the record data for the `NS` type record of the zone, which in this example is `ns1.example.net.`. To verify this you can query the nameserver using `dig` for the `NS` type:

```
$ dig @ns1.example.net example.org. -t NS +short
ns1.devstack.org.
```

You can also verify the `A` recordsets. You dont need the `-t` option because it is the default:

```
$ dig @ns1.example.net web.example.org. +short
192.0.2.2
192.0.2.1
$ dig @ns1.example.net www.example.org. +short
192.0.2.1
```

If you want to construct a `TXT` record that exceeds the 255-octet maximum length of a character-string, it has to be split into multiple strings as defined in RFC7208 section 3.3. For example, `"v=DKIM1; .... firstsecond string..."` can become `"v=DKIM1; .... first" "second string..."`. If you provide a record data with less than 255 characters, it will be treated as a single character-string and validated for empty spaces outside quotes and unescaped double quotation marks as in RFC1035 section 5.1.

For example, to create a `TXT` record made of one string of 410 characters you can split it into 2 to like this:

```
$ openstack recordset create --type TXT --record '"210 characters string"
↪"200 characters string"' example.org. _domainkey
```

Updating a recordset

You can modify a recordset by using the `openstack recordset set` command. When updating a recordset by name, you must use the FQDN. As with most OpenStack commands, you can also use recordset ID. For example, to update the recordset `www.example.org.` to contain two records, you could use the following:

```
$ openstack recordset set example.org. www.example.org. --record 192.0.2.1 --
↪record 192.0.2.2
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | UPDATE                               |
| created_at  | 2021-05-03T03:30:16.000000           |
| description | None                                 |
| id          | 549c3e83-443f-474b-b467-6bcd7cb9f37d |
| name        | www.example.org.                     |
| project_id  | c85fdba96041438fa0cad2dc7909d3f5     |
| records     | 192.0.2.2                            |
|             | 192.0.2.1                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | 2021-05-03T03:44:16.000000           |
| version     | 5                                    |
| zone_id     | 077460ef-34db-486a-8d59-c9564dc3a3a9 |
| zone_name   | example.org.                         |
+-------------+--------------------------------------+
```

Deleting a recordset

You can use the `openstack recordset delete` command to remove recordsets using the zone and either the FQDN or the recordset ID.

```
$ openstack recordset delete example.org. web.example.org.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | DELETE                               |
| created_at  | 2021-05-03T03:47:00.000000           |
| description | None                                 |
| id          | 5ab3418f-5377-47eb-b967-9e9ff7f3c26b |
| name        | web.example.org.                     |
| project_id  | c85fdba96041438fa0cad2dc7909d3f5     |
| records     | 192.0.2.1                            |
|             | 192.0.2.2                            |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | A                                    |
| updated_at  | 2021-05-03T03:47:13.000000           |
| version     | 2                                    |
```

(continues on next page)

(continued from previous page)

```
| zone_id     | 077460ef-34db-486a-8d59-c9564dc3a3a9 |
| zone_name   | example.org.                         |
+-------------+--------------------------------------+
```

How To Manage PTR Records

PTR Record Basics

PTR records provide a reverse mapping from a single IP or set of IP addresses to a fully qualified domain name (FQDN). For example,

```
$ dig -x 192.0.2.12 +short
example.org.
```

The way this works in the DNS system is through the *in-addr.arpa.* zone. For example

```
$ dig example.org +short
192.0.2.12
$ dig -x 192.0.2.12
; <<>> DiG 9.9.5-3ubuntu0.1-Ubuntu <<>> -x 192.0.2.12
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 3431
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;12.2.0.192.in-addr.arpa.   IN      PTR      example.org.

;; AUTHORITY SECTION:
12.2.0.192.in-addr.arpa. 3600 IN     NS      ns1.example.org.

;; Query time: 40 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Feb 20 19:05:44 UTC 2015
;; MSG SIZE  rcvd: 119
```

In the question section we see the address being requested from the DNS system as *12.2.0.192.in-addr.arpa.*. As you can see, the IP address has been reversed in order to function similarly to a domain name where the more specific elements come first. The reversed IP address is then added to the *in-addr.arpa.* domain, at which point the DNS system can perform a simple look up to find any *PTR* records that describe what domain name, if any, maps to that IP.

Create a PTR Record in Designate

To create a *PTR* record in Designate we need a *in-addr.arpa.* zone that will receive the actual *PTR* record

Using the V2 API and the OpenStack CLI

To begin lets create a zone that we want to return when we do our reverse lookup.

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "example.org.",
  "email": "admin@example.org",
  "ttl": 3600,
  "description": "A great example zone"
}
```

Here is the JSON response describing the new zone.

```
HTTP/1.1 202 Accepted
Location: http://127.0.0.1:9001/v2/zones/251fbde4-6eb8-44e6-bc48-e095f1763a1f
Content-Length: 476
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-bfcd0723-624c-4ec2-bbd5-99e985efe8db
Date: Tue, 02 Jun 2020 17:24:10 GMT
Connection: keep-alive

{
  "id": "251fbde4-6eb8-44e6-bc48-e095f1763a1f",
  "pool_id": "794ccc2c-d751-44fe-b57f-8894c9f5c842",
  "project_id": "123d51544df443e790b8e95cce52c285",
  "name": "example.org.",
  "email": "admin@example.org",
  "description": "A great example zone",
  "ttl": 3600,
  "serial": 1591118650,
  "status": "PENDING",
  "action": "CREATE",
  "version": 1,
  "attributes": {},
  "type": "PRIMARY",
  "masters": [],
  "created_at": "2020-06-02T17:24:10.000000",
  "updated_at": null,
  "transferred_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/251fbde4-6eb8-44e6-bc48-
↪e095f1763a1f"
  }
}
```

Using the CLI:

```
$ openstack zone create --email admin@example.org \
    --description "A great example zone" --ttl 3600 example.org.
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| action        | CREATE                               |
| attributes    |                                      |
| created_at    | 2020-06-02T17:24:10.000000           |
| description   | A great example zone                 |
| email         | admin@example.org                    |
| id            | 251fbde4-6eb8-44e6-bc48-e095f1763a1f |
| masters       |                                      |
| name          | example.org.                         |
| pool_id       | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id    | 123d51544df443e790b8e95cce52c285     |
| serial        | 1591118650                           |
| status        | PENDING                              |
| transferred_at | None                                |
| ttl           | 3600                                 |
| type          | PRIMARY                              |
| updated_at    | None                                 |
| version       | 1                                    |
+---------------+--------------------------------------+
```

Note

The *status* is *PENDING*. If we make a *GET* request to the *self* field in the zone, it will most likely have been processed and updated to *ACTIVE*.

Now that we have a zone we would like to use for our reverse DNS lookup, we need to add an *in-addr.arpa.* zone that includes the IP address we want to look up.

Lets configure *192.0.2.11* to return our *example.org.* domain name when we do a reverse look up.

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "11.2.0.192.in-addr.arpa.",
  "email": "admin@example.org",
  "ttl": 3600,
  "description": "A in-addr.arpa. zone for reverse lookups"
}
```

As you can see, in the *name* field weve reversed our IP address and used that as a subdomain in the *in-addr.arpa.* zone.

Here is the response.

```
HTTP/1.1 202 Accepted
Location: http://127.0.0.1:9001/v2/zones/f5546034-b27e-4326-bf9d-c53ed879f7fa
Content-Length: 512
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-4e691123-045e-4f8e-ae50-b5eabb5af3fa
Date: Tue, 02 Jun 2020 17:32:46
Connection: keep-alive

{
  "id": "f5546034-b27e-4326-bf9d-c53ed879f7fa",
  "pool_id": "794ccc2c-d751-44fe-b57f-8894c9f5c842",
  "project_id": "123d51544df443e790b8e95cce52c285",
  "name": "11.2.0.192.in-addr.arpa.",
  "email": "admin@example.org",
  "description": "A in-addr.arpa. zone for reverse lookups",
  "ttl": 3600,
  "serial": 1591119166,
  "status": "PENDING",
  "action": "CREATE",
  "version": 1,
  "attributes": {},
  "type": "PRIMARY",
  "masters": [],
  "created_at": "2020-06-02T17:32:47.000000",
  "updated_at": null,
  "transferred_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/f5546034-b27e-4326-bf9d-
↪c53ed879f7fa"
  }
}
```

Using the CLI:

```
$ openstack zone create --email admin@example.org \
    --ttl 3600 --description "A in-addr.arpa. zone for reverse lookups" \
    11.2.0.192.in-addr.arpa.
+---------------+----------------------------------------+
| Field         | Value                                  |
+---------------+----------------------------------------+
| action        | CREATE                                 |
| attributes    |                                        |
| created_at    | 2020-06-02T17:32:47.000000             |
| description   | A in-addr.arpa. zone for reverse lookups |
| email         | admin@example.org                      |
| id            | f5546034-b27e-4326-bf9d-c53ed879f7fa   |
| masters       |                                        |
| name          | 11.2.0.192.in-addr.arpa.               |
| pool_id       | 794ccc2c-d751-44fe-b57f-8894c9f5c842   |
| project_id    | 123d51544df443e790b8e95cce52c285       |
```

(continues on next page)

(continued from previous page)

```
| serial          | 1591119166                           |
| status          | PENDING                              |
| transferred_at  | None                                 |
| ttl             | 3600                                 |
| type            | PRIMARY                              |
| updated_at      | None                                 |
| version         | 1                                    |
+-----------------+--------------------------------------+
```

Now that we have our *in-addr.arpa.* zone, we add a new *PTR* record to the zone.

```
POST /v2/zones/f5546034-b27e-4326-bf9d-c53ed879f7fa/recordsets HTTP/1.1
Content-Type: application/json
Accept: application/json

{
  "name": "11.2.0.192.in-addr.arpa.",
  "type": "PTR",
  "records": [
    "example.org."
  ],
  "ttl": 3600,
  "description": "A PTR recordset"
}
```

Here is the response.

```
HTTP/1.1 202 Accepted
Location: http://127.0.0.1:9001/v2/zones/f5546034-b27e-4326-bf9d-c53ed879f7fa/
↪recordsets/ca604f72-83e6-421f-bf1c-bb4dc1df994a
Content-Length: 573
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-5b7044d0-591a-445a-839f-1403b1455824
Date: Tue, 02 Jun 2020 19:55:50 GMT
Connection: keep-alive

{
  "id": "ca604f72-83e6-421f-bf1c-bb4dc1df994a",
  "zone_id": "f5546034-b27e-4326-bf9d-c53ed879f7fa",
  "project_id": "123d51544df443e790b8e95cce52c285",
  "name": "11.2.0.192.in-addr.arpa.",
  "zone_name": "11.2.0.192.in-addr.arpa.",
  "type": "PTR",
  "records": [
    "example.org."
  ],
  "description": "A PTR recordset",
  "ttl": 3600,
  "status": "PENDING",
  "action": "CREATE",
```

(continues on next page)

(continued from previous page)

```
  "version": 1,
  "created_at": "2020-06-02T19:55:50.000000",
  "updated_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/f5546034-b27e-4326-bf9d-
↪c53ed879f7fa/recordsets/ca604f72-83e6-421f-bf1c-bb4dc1df994a"
  }
}
```

With the CLI:

```
$ openstack recordset create --record example.org. --type PTR \
    --ttl 3600 --description "A PTR recordset" \
    11.2.0.192.in-addr.arpa. 11.2.0.192.in-addr.arpa.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2020-06-02T19:55:50.000000           |
| description | A PTR recordset                      |
| id          | ca604f72-83e6-421f-bf1c-bb4dc1df994a |
| name        | 11.2.0.192.in-addr.arpa.             |
| project_id  | 123d51544df443e790b8e95cce52c285     |
| records     | example.org.                         |
| status      | PENDING                              |
| ttl         | 3600                                 |
| type        | PTR                                  |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | f5546034-b27e-4326-bf9d-c53ed879f7fa |
| zone_name   | 11.2.0.192.in-addr.arpa.             |
+-------------+--------------------------------------+
```

We should now have a correct *PTR* record assigned in our nameserver that we can test.

Lets test it out!

```
$ dig @localhost -x 192.0.2.11

; <<>> DiG 9.9.5-3ubuntu0.1-Ubuntu <<>> @localhost -x 192.0.2.11
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32832
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
```

(continues on next page)

(continued from previous page)

```
;11.2.0.192.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
11.2.0.192.in-addr.arpa. 3600 IN     PTR     example.org.

;; AUTHORITY SECTION:
11.2.0.192.in-addr.arpa. 3600 IN     NS      ns1.example.org.

;; Query time: 3 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Feb 20 21:45:53 UTC 2015
;; MSG SIZE  rcvd: 98
```

As you can see from the answer section everything worked as expected.

IPv6

Following the previous example we will configure *fd00::2:11* to return our *example.org.* domain name. As reverse DNS lookups for *IPv6* addresses use the special domain *ip6.arpa*, we need to create

```
$ openstack zone create --email admin@example.org \
    --ttl 3600 --description "A ip6.arpa zone for IPv6 reverse lookups" \
    1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.
+----------------+------------------------------------------------------------
↪---------------+
| Field          | Value                                                      ␣
↪               |
+----------------+------------------------------------------------------------
↪---------------+
| action         | CREATE                                                     ␣
↪               |
| attributes     |                                                            ␣
↪               |
| created_at     | 2020-06-04T13:07:36.000000                                 ␣
↪               |
| description    | IPv6 reverse lookup zone                                   ␣
↪               |
| email          | admin@example.org                                          ␣
↪               |
| id             | 9c8f30a1-6d9d-4f40-9fac-ab8abfb24fba                       ␣
↪               |
| masters        |                                                            ␣
↪               |
| name           | 1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.
↪0.d.f.ip6.arpa. |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842                       ␣
↪               |
| project_id     | 123d51544df443e790b8e95cce52c285                           ␣
↪               |
| serial         | 1591276055                                                 ␣
```

(continues on next page)

(continued from previous page)

```
↪                |
| status         | PENDING                                                    ␣
↪                |
| transferred_at | None                                                       ␣
↪                |
| ttl            | 3600                                                       ␣
↪                |
| type           | PRIMARY                                                    ␣
↪                |
| updated_at     | None                                                       ␣
↪                |
| version        | 1                                                          ␣
↪                |
+----------------+------------------------------------------------------------
↪---------------+
```

And add the *PTR* record

```
$ openstack recordset create --record example.org. --type PTR \
    --ttl 3600 --description "A PTR recordset" \
    1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.␣
↪\
    1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.
+-------------+------------------------------------------------------------------
↪-------------+
| Field       | Value                                                            ␣
↪             |
+-------------+------------------------------------------------------------------
↪-------------+
| action      | CREATE                                                           ␣
↪             |
| created_at  | 2020-06-04T13:10:30.000000                                       ␣
↪             |
| description | A PTR recordset                                                  ␣
↪             |
| id          | 246c5cbb-315d-437d-a52f-bf0a0cfa91a0                             ␣
↪             |
| name        | 1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.
↪f.ip6.arpa. |
| project_id  | 123d51544df443e790b8e95cce52c285                                 ␣
↪             |
| records     | example.org.                                                     ␣
↪             |
| status      | PENDING                                                          ␣
↪             |
| ttl         | 3600                                                             ␣
↪             |
| type        | PTR                                                              ␣
↪             |
| updated_at  | None                                                             ␣
```

(continues on next page)

(continued from previous page)

```
↪             |
| version     | 1                                                                ␣
↪             |
| zone_id     | 9c8f30a1-6d9d-4f40-9fac-ab8abfb24fba                             ␣
↪             |
| zone_name   | 1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.
↪f.ip6.arpa. |
+-------------+------------------------------------------------------------------
↪-------------+
```

Now we can do a reverse lookup with

```
$ dig @localhost -x fd00::2:11

; <<>> DiG 9.11.3-1ubuntu1.12-Ubuntu <<>> @10.5.0.32 -x fd00::2:11
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50892
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 812dd247d36b98504b6d12485ed8f44bd7ae0a902343c348 (good)
;; QUESTION SECTION:
;1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa. IN␣
↪PTR

;; ANSWER SECTION:
1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.␣
↪3600 IN PTR example.org.

;; AUTHORITY SECTION:
1.1.0.0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa.␣
↪3600 IN NS ns1.example.org.

;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Jun 04 13:16:59 UTC 2020
;; MSG SIZE  rcvd: 197
```

Advanced Usage

You can add many *PTR* records to a larger subnet by using a more broadly defined *in-addr.arpa.* zone. For example, if we wanted to ensure *any* IP in a subnet resolves to a specific domain we would add a wildcard DNS record to this zone.

```
POST /v2/zones HTTP/1.1
Accept: application/json
```

(continues on next page)

(continued from previous page)

```
Content-Type: application/json

{
  "name": "2.0.192.in-addr.arpa.",
  "type": "PRIMARY",
  "email": "admin@example.org",
  "ttl": 3600,
  "description": "A more broadly defined in-addr.arpa. zone for reverse␣
↪lookups"
}
```

With the CLI:

```
$ openstack zone create --email admin@example.org --ttl 3600 \
    --description "A more broadly defined in-addr.arpa. zone for reverse␣
↪lookups" \
    2.0.192.in-addr.arpa.
+---------------+--------------------------------------------------------------
↪---+
| Field         | Value                                                        ␣
↪   |
+---------------+--------------------------------------------------------------
↪---+
| action        | CREATE                                                       ␣
↪   |
| attributes    |                                                              ␣
↪   |
| created_at    | 2020-06-02T20:07:11.000000                                   ␣
↪   |
| description   | A more broadly defined in-addr.arpa. zone for reverse␣
↪lookups |
| email         | admin@example.org                                            ␣
↪   |
| id            | e9fd0ced-1d3e-43fa-b9aa-6d4b7a73988d                         ␣
↪   |
| masters       |                                                              ␣
↪   |
| name          | 2.0.192.in-addr.arpa.                                        ␣
↪   |
| pool_id       | 794ccc2c-d751-44fe-b57f-8894c9f5c842                         ␣
↪   |
| project_id    | 123d51544df443e790b8e95cce52c285                             ␣
↪   |
| serial        | 1591128431                                                   ␣
↪   |
| status        | PENDING                                                      ␣
↪   |
| transferred_at | None                                                        ␣
↪   |
| ttl           | 3600                                                         ␣
```

(continues on next page)

(continued from previous page)

```
↪    |
| type            | PRIMARY                                                    ␣
↪    |
| updated_at      | None                                                       ␣
↪    |
| version         | 1                                                          ␣
↪    |
+-----------------+------------------------------------------------------------
↪---+
```

We then could use the corresponding domain to create a *PTR* record for a specific IP.

```
POST /v2/zones/e9fd0ced-1d3e-43fa-b9aa-6d4b7a73988d/recordsets HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "3.2.0.192.in-addr.arpa.",
  "type": "PTR"
  "ttl": 3600,
  "records": [
    "cats.example.com."
  ]
}
```

With the CLI:

```
$ openstack recordset create --record cats.example.org. --type PTR \
    --ttl 3600 2.0.192.in-addr.arpa. 3.2.0.192.in-addr.arpa.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2020-06-02T20:10:54.000000           |
| description | None                                 |
| id          | c843729b-7aaf-4f99-a40a-d9bf70edf271 |
| name        | 3.2.0.192.in-addr.arpa.              |
| project_id  | 123d51544df443e790b8e95cce52c285     |
| records     | cats.example.org.                    |
| status      | PENDING                              |
| ttl         | 3600                                 |
| type        | PTR                                  |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | e9fd0ced-1d3e-43fa-b9aa-6d4b7a73988d |
| zone_name   | 2.0.192.in-addr.arpa.                |
+-------------+--------------------------------------+
```

Or with a wildcard DNS record:

```
$ openstack recordset create --record example.org. --type PTR \
    --ttl 3600 2.0.192.in-addr.arpa. *.2.0.192.in-addr.arpa.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2020-06-04T12:22:45.000000           |
| description | None                                 |
| id          | 4fa96619-a1f8-4409-ba5f-fa904db4c97c |
| name        | *.2.0.192.in-addr.arpa.              |
| project_id  | 123d51544df443e790b8e95cce52c285     |
| records     | example.org.                         |
| status      | PENDING                              |
| ttl         | 3600                                 |
| type        | PTR                                  |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | e9fd0ced-1d3e-43fa-b9aa-6d4b7a73988d |
| zone_name   | 2.0.192.in-addr.arpa.                |
+-------------+--------------------------------------+
```

When we do our reverse look, we should see *cats.example.com.*

```
$ dig @localhost -x 192.0.2.3 +short
cats.example.com.
```

When we query any other IP address in *192.0.2.0/24* we get

```
$ dig @10.5.0.32 -x 192.0.2.10 +short
example.org.
```

Success!

Note

In BIND9, when creating a new *PTR* we could skip the zone name. For example, if the zone is *2.0.192.in-addr.arpa.*, using *12* for the record name ends up as *12.2.0.192.in-addr.arpa.*. In Designate, the name of a record MUST be a complete host name.

Classless IN-Addr.ARPA Delegation

You may want to delegate blocks of IP addresses to projects that do not align to subnet boundries. For example, if you wanted to give project A three IP addresses. To allow project A to manage DNS records for those three addresses, without delegating a whole subnet zone to project A, you can use classless IN-ADDR.ARPA delegation as described in RFC 2317.

Note

As discussed in section 4 of RFC 2317, the examples in the RFC use / in the delegated zones but - is recommended. Designate will not allow you to use / in zone names. You will need to use the

recommended - instead.

In this example, we will delegate a PTR zone for three IP addresses, from the 192.0.2.0/24 subnet, to the *Demo* project 9284a20339184a9bb299386c380211c7.

Note

Unless noted in the examples, the commands are using a credential with an admin role. This is not necessary, but is a typical use case.

First, the full subnet in-addr.arpa zone will need to be created:

```
$ openstack zone create --email me@example.com 2.0.192.in-addr.arpa.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| attributes     |                                      |
| created_at     | 2022-09-09T20:05:41.000000           |
| description    | None                                 |
| email          | me@example.com                       |
| id             | bbdf0e8f-8d73-4659-ae62-f59e95a31cd7 |
| masters        |                                      |
| name           | 2.0.192.in-addr.arpa.                |
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | cc5ab848dbe7462e9c7603d54a9af90f     |
| serial         | 1662753940                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+
```

Next we will create the delegated zone:

```
$ openstack zone create --email me@example.com 1-3.2.0.192.in-addr.arpa.
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| action         | CREATE                               |
| attributes     |                                      |
| created_at     | 2022-09-09T20:06:59.000000           |
| description    | None                                 |
| email          | me@example.com                       |
| id             | 2d353ed7-cb7f-4ff7-9c1e-54481304f4cb |
| masters        |                                      |
| name           | 1-3.2.0.192.in-addr.arpa.            |
```

(continues on next page)

(continued from previous page)

```
| pool_id        | 794ccc2c-d751-44fe-b57f-8894c9f5c842 |
| project_id     | cc5ab848dbe7462e9c7603d54a9af90f     |
| serial         | 1662754018                           |
| status         | PENDING                              |
| transferred_at | None                                 |
| ttl            | 3600                                 |
| type           | PRIMARY                              |
| updated_at     | None                                 |
| version        | 1                                    |
+----------------+--------------------------------------+
```

Now we can share the delegated zone with the *Demo* project:

```
$ openstack zone share create 1-3.2.0.192.in-addr.arpa.␣
↪9284a20339184a9bb299386c380211c7
+------------------+--------------------------------------+
| Field            | Value                                |
+------------------+--------------------------------------+
| created_at       | 2022-09-09T20:07:20.000000           |
| id               | 7859ca43-bcee-4fd1-aa2d-efda17b75198 |
| project_id       | cc5ab848dbe7462e9c7603d54a9af90f     |
| target_project_id | 9284a20339184a9bb299386c380211c7     |
| updated_at       | None                                 |
| zone_id          | 2d353ed7-cb7f-4ff7-9c1e-54481304f4cb |
+------------------+--------------------------------------+
```

Once we have the zones created and shared, we can now add the CNAME records to the full subnet zone that point to the delegated zone records. This will need to be repeated for each IP address being delegated. This example creates the first CNAME record for the 192.0.2.1 IP address.

```
$ openstack recordset create --record 1.1-3.2.0.192.in-addr.arpa. --type␣
↪CNAME 2.0.192.in-addr.arpa. 1.2.0.192.in-addr.arpa.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2022-09-09T20:09:16.000000           |
| description | None                                 |
| id          | 482bd367-9815-4d86-a93d-734bbc92499a |
| name        | 1.2.0.192.in-addr.arpa.              |
| project_id  | cc5ab848dbe7462e9c7603d54a9af90f     |
| records     | 1.1-3.2.0.192.in-addr.arpa.          |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | CNAME                                |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | bbdf0e8f-8d73-4659-ae62-f59e95a31cd7 |
| zone_name   | 2.0.192.in-addr.arpa.                |
+-------------+--------------------------------------+
```

Finally, members of the *Demo* project can now create the PTR records for the delegates IP addresses. In this example the administrator will create the first record on behalf of the *Demo* project.

```
$ openstack recordset create --sudo-project-id␣
↪9284a20339184a9bb299386c380211c7 --record www.example.com. --type PTR 1-3.2.
↪0.192.in-addr.arpa. 1.1-3.2.0.192.in-addr.arpa.
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| action      | CREATE                               |
| created_at  | 2022-09-09T20:08:17.000000           |
| description | None                                 |
| id          | cea3f3ce-687b-422c-a378-bdcfe382a159 |
| name        | 1.1-3.2.0.192.in-addr.arpa.          |
| project_id  | 9284a20339184a9bb299386c380211c7     |
| records     | www.example.com.                     |
| status      | PENDING                              |
| ttl         | None                                 |
| type        | PTR                                  |
| updated_at  | None                                 |
| version     | 1                                    |
| zone_id     | 2d353ed7-cb7f-4ff7-9c1e-54481304f4cb |
| zone_name   | 1-3.2.0.192.in-addr.arpa.            |
+-------------+--------------------------------------+
```

We can now use dig to query a recursive resolver to verify the delegation:

```
$ dig -x 192.0.2.1 @198.51.100.5

; <<>> DiG 9.16.32-RH <<>> -x 192.0.2.1 @198.51.100.5
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16209
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: a415d9b43dcef11c01000000631ba068973cbfbf5b765032 (good)
;; QUESTION SECTION:
;1.2.0.192.in-addr.arpa.            IN      PTR

;; ANSWER SECTION:
1.2.0.192.in-addr.arpa.     3600    IN      CNAME   1.1-3.2.0.192.in-addr.
↪arpa.
1.1-3.2.0.192.in-addr.arpa. 3600 IN PTR     www.example.com.

;; Query time: 0 msec
;; SERVER: 198.51.100.5#53(198.51.100.5)
;; WHEN: Fri Sep 09 13:22:00 PDT 2022
;; MSG SIZE  rcvd: 149
```

Note

Your resolver or DNS server settings (such as allow recursion and/or minimal responses) may cause dig to only display the CNAME and not resolve the PTR record in the same request.

Using DNS with Neutron & Nova

Neutron can be integrated with Designate to provide automatic *recordset* creation for ports and, by proxy, Nova server instances. This section will describe how you can use this integration to have Designate DNS *recordsets* created for Neutron ports and Nova instances at creation time.

Neutron DNS Extensions

DNS integration in Neutron is optional and an extension must be enabled in the Neutron configuration file, by a cloud administrator, for DNS names to be assigned automatically to Neutron and Nova resources. You can check if a DNS integration extension is enabled by querying the Neutron extensions API:

```
$ openstack extension list --network -f value -c Alias | grep dns-integration
dns-integration
```

One of these extensions must be enabled to allow Neutron and, via Neutron, Nova to automatically create DNS *recordsets* in Designate:

- dns-integration
- dns-domain-ports (includes dns-integration)
- subnet-dns-publish-fixed-ip (includes dns-integration and dns-domain-ports)
- dns-integration-domain-keywords (includes all others)

dns-integration

When the *dns-integration* extension is enabled the following DNS attributes will be available via Neutron:

Resource	dns_name	dns_domain
Ports	Yes	No
Networks	No	Yes
Floating IPs	Yes	Yes

dns-domain-ports

In addition, if the *dns-domain-ports* extension is enabled in Neutron, ports can be created with a dns_domain specified. This dns_domain will take precedence over the dns_domain setting for the network. You can check if the *dns-domain-ports* extension is enabled by querying the Neutron extensions API:

```
$ openstack extension list --network -f value -c Alias | grep dns-domain-ports
dns-domain-ports
```

With the *dns-domain-ports* extension is enabled the following DNS settings will be available via Neutron:

Resource	dns_name	dns_domain
Ports	Yes	Yes
Networks	No	Yes
Floating IPs	Yes	Yes

Both of these extensions impose a set of criteria for when DNS *recordsets* will be created in Designate.

- A *dns_domain* must be specified either on the network, port, or floating IP. If both the network and the port or floating IP specify a *dns_domain*, the *dns_domain* specified on the port or floating IP will take precedent over the *dns_domain* provided on the network.
- The network must not have the *router:external* field set to True.
- The network type must be one of: FLAT, VLAN, GRE, VXLAN, or GENEVE.
- For VLAN, GRE, VXLAN, or GENEVE networks, the segmentation ID must be outside the ranges configured in the Neutron ml2_confg file. For example, with VXLAN networks, the range setting is [ml2_type_vxlan] vni_ranges.
- The *zone* for the *dns_domain* must already exist in Designate and the project ID creating the Nova instance, port, or floating IP must have permission to create *recordsets* in the *zone*.

These restrictions typically mean that a special network will need to be created by an administrator that will allow *recordsets* to be created in Designate.

If these criteria are not all met, Neutron will create a DNS assignment in the Neutron internal resolvers using the default *dns_domain* specified in the Neutron configuration file. The current default domain is openstacklocal..

Warning

If the user creating the Nova instance, port, or floating IP does not have permission to create *recordsets* in the *zone* or the *zone* does not exist in Designate, Neutron will create the port with the *dns_assignment* field populated using the *dns_domain* provided, but no *recordset* will be created in Designate. Neutron will log the error Error publishing port data in external DNS service..

subnet-dns-publish-fixed-ip

A third Neutron extension is available called *subnet-dns-publish-fixed-ip*. This extension includes the capabilities of the *dns-domain-ports* extension, but removes the restrictions if the subnet *dns_publish_fixed_ip* property is set to True.

dns-integration-domain-keywords

The fourth Neutron extension, including the capabilities of the *subnet-dns-publish-fixed-ip* extension, is called *dns-integration-domain-keywords*. It allows the use of keywords in the *dns_domain* that will be replaced when a port is created. Valid keywords are: <project_id>, <project_name>, <user_id>, and <user_name>.

Note

For more information on enabling DNS integration in Neutron, see the Neutron Networking Guide.

DNS for Nova Server Instances

DNS integration with Neutron allows you to automatically create a DNS *recordset* for Nova instances. When Nova requests the Neutron port to be created for the new instance, Neutron will attempt to create a DNS *recordset* for the port in Designate.

As an example, we will create a new Nova instance with the DNS name of server.example.org registered in Designate.

Note

This example is for user created networks. DNS records can be automatically created for Nova server instances on networks created by a cloud administrator if they meet the Neutron criteria.

Steps:

1. Check that the *subnet-dns-publish-fixed-ip* Neutron extension is enabled.
2. Create the *zone* example.org. in Designate.
3. Create a network, providing the *dns_domain* of example.org., that we will use for the Nova instance.
4. Create a subnet on the network with *dns_publish_fixed_ip* set to True.
5. Create the Nova instance, with name server and a NIC on the network.
6. Verify the DNS *recordset* was created in the Designate *zone*.

Note

The DNS domain must always be a *Fully Qualified Domain Name* (FQDN), meaning it will always end with a period.

CLI Commands:

```
$ openstack extension list --network -f value -c Alias | grep subnet-dns-
↪publish-fixed-ip
$ openstack zone create --email example@example.org example.org.
$ openstack network create --dns-domain example.org. example-net
$ openstack subnet create --allocation-pool start=192.0.2.10,end=192.0.2.200 -
↪-network example-net --subnet-range 192.0.2.0/24 --dns-publish-fixed-ip␣
↪example-subnet
$ openstack server create --image cirros-0.5.2-x86_64-disk --flavor 1 --nic␣
↪net-id=example-net server
$ openstack recordset list --type A example.org.

+--------------+--------------------+------+------------+--------+--------+
| id           | name               | type | records    | status | action |
+--------------+--------------------+------+------------+--------+--------+
| 7b8d1be6-1b23 | server.example.org. | A    | 192.0.2.44 | ACTIVE | NONE   |
| -478a-94d5-60 |                    |      |            |        |        |
| b876dca2c8    |                    |      |            |        |        |
+--------------+--------------------+------+------------+--------+--------+
```

DNS for Neutron Ports

DNS integration with Neutron allows you to automatically create a DNS *recordset* for Neutron ports.

As an example, we will create a new Neutron port with the DNS name of example-port.example.org registered in Designate.

Note

This example is for user created networks. DNS records can be automatically created for Neutron ports on networks created by a cloud administrator if they meet the Neutron criteria.

Steps:

1. Check that the *subnet-dns-publish-fixed-ip* Neutron extension is enabled.
2. Create the *zone* example.org. in Designate.
3. Create a network, providing the *dns_domain* of example.org., that we will use for the Neutron port.
4. Create a subnet on the network with *dns_publish_fixed_ip* set to True.
5. Create the Neutron port specifying the *dns_name* of example-port for the port.
6. Verify the DNS *recordset* was created in the Designate *zone*.

Note

The DNS domain must always be a *Fully Qualified Domain Name* (FQDN), meaning it will always end with a period.

CLI Commands:

```
$ openstack extension list --network -f value -c Alias | grep subnet-dns-
↪publish-fixed-ip
$ openstack zone create --email example@example.org example.org.
$ openstack network create --dns-domain example.org. example-net
$ openstack subnet create --allocation-pool start=192.0.2.10,end=192.0.2.200 -
↪-network example-net --subnet-range 192.0.2.0/24 --dns-publish-fixed-ip␣
↪example-subnet
$ openstack port create --network example-net --dns-name example-port my-
↪example-port
$ openstack recordset list --type A example.org.

+--------------+--------------------------+------+-------------+--------+---
↪-----+
| id           | name                     | type | records     | status |␣
↪action |
+--------------+--------------------------+------+-------------+--------+---
↪-----+
| 9ebbe94f-2442 | example-port.example.org. | A    | 192.0.2.149 | ACTIVE |␣
↪NONE   |
| -4bb8-9cfa-6d |                          |      |             |        |  ␣
```

(continues on next page)

(continued from previous page)

```
↪    |
| ca1daba73f    |                           |      |            |        | ␣
↪    |
+---------------+---------------------------+------+------------+--------+----
↪----+
```

DNS for Floating IPs

DNS integration with Neutron allows you to automatically create a DNS *recordset* for Neutron floating IP addresses.

As an example, we will create a new Neutron floating IP with the DNS name of example-fip.example.org registered in Designate.

Steps:

1. Create the Neutron floating IP specifying the *dns_name* of example-fip and the *dns_domain* as example.org..
2. Verify the DNS *recordset* was created in the Designate *zone*.

Note

The DNS domain must always be a *Fully Qualified Domain Name* (FQDN), meaning it will always end with a period.

CLI Commands:

```
$ openstack floating ip create --dns-name example-fip --dns-domain example.
↪org. example-net
$ openstack recordset list --type A example.org.

+--------------+-------------------------+------+------------+--------+----
↪----+
| id           | name                    | type | records    | status |␣
↪action |
+--------------+-------------------------+------+------------+--------+----
↪----+
| e1eca823-169d | example-fip.example.org. | A    | 192.0.2.106 | ACTIVE |␣
↪NONE  |
| -4d0a-975e-91 |                          |      |             |        |  ␣
↪   |
| a9907ec0c1    |                          |      |             |        |  ␣
↪   |
+--------------+-------------------------+------+------------+--------+-----
↪---+
```

1.5 Administration guide

In this section, you will find documentation relevant for administering and operating Designate.

Contents:

1.5.1 Managing Top Level Domain Names

System Administrators can use top level domains (TLDs) to restrict the domains under which users can create zones. While in the Domain Name System the term TLD refers specifically to the set of domains that lie directly below the root, such as `.org`, in Designate a TLD can be any domain.

For example, if you want to require that users create zones ending in `.org.`, this can be achieved by creating a single `.org` TLD:

```
$ openstack tld create --name org
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2021-06-10T05:20:16.000000           |
| description | None                                 |
| id          | 9fd0a12d-511e-4024-bf76-6ec2e3e71edd |
| name        | org                                  |
| updated_at  | None                                 |
+-------------+--------------------------------------+
```

Note

When using the *openstack tld* command, ensure that the FQDN that you enter has no trailing dot (*example.net.*).

If you now attempt to create a zone that does not lie within the `.org` TLD, it will fail:

```
$ openstack zone create --email admin@test.net test.net.
Invalid TLD
```

TLDs are much like an allowlist: if there are many TLDs then the zone must exist within one of the TLDs. If no TLDs have been created in Designate, then users can create any zone. Unlike the blacklists feature, TLDs do not have a policy that allows priviliged users to create zones outside the allowed TLDs.

You can modify the values for a TLD using the *set* command. You can use either the name or the ID to specify which TLD to set:

```
$ openstack tld set org --name example.net
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2021-06-10T05:20:16.000000           |
| description |                                      |
| id          | 9fd0a12d-511e-4024-bf76-6ec2e3e71edd |
| name        | example.net                          |
```

(continues on next page)

(continued from previous page)

```
| updated_at  | 2021-06-10T07:09:45.000000           |
+-------------+--------------------------------------+
```

You can delete a TLD by providing either the ID or the current name:

```
$ openstack tld delete org
```

This command has no output when completed successfully.

1.5.2 DNS Server Plugin Documentation

Contents:

Akamai v2 Backend

This page documents using the Akamai v2 backend. The backend uses the FastDNS V2 API to create and delete zones remotely.

Designate Configuration

Example configuration required: One section for each pool target

```
- name: default-akamai-v2
  # The name is immutable. There will be no option to change the␣
↪name after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Akamai v2

  attributes: {}

  # List out the NS records for zones hosted within this pool
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1

  # List out the nameservers for this pool. These are the actual␣
↪Akamai servers.
  # We use these to verify changes have propagated to all␣
↪nameservers.
  nameservers:
    - host: 192.0.2.2
      port: 53

  # List out the targets for this pool. For Akamai, most often,␣
↪there will be
  # one entry for each Akamai server.
  targets:
    - type: akamai_v2
      description: Akamai v2 server
```

(continues on next page)

(continued from previous page)

```
    # List out the designate-mdns servers from which Akamai␣
↪servers should
    # request zone transfers (AXFRs) from.
    masters:
      - host: 192.0.2.1
        port: 5354

    options:
      host: 192.0.2.2
      port: 53
      akamai_host: 192.0.2.2
      akamai_client_token: client_token_string
      akamai_access_token: access_token_string
      akamai_client_secret: client_secret_string
      akamai_contract_id: contract_id
      akamai_gid: group_id
```

Then update the pools in designate - see *designate-manage pool* for further details on the `designate-manage pool` command

```
$ designate-manage pool update
```

Bind9 Backend

This page documents using the Bind 9 backend. The backend uses the rndc utility to create and delete zones remotely.

The traffic between rndc and Bind is authenticated with a key.

Designate Configuration

Example configuration required for Bind9 operation. One section for each pool target

```
 targets:
   - type: bind9
     description: BIND9 Server 1

     # List out the designate-mdns servers from which BIND servers␣
↪should
     # request zone transfers (AXFRs) from.
     masters:
       - host: 192.0.2.1
         port: 5354

     # BIND Configuration options
     options:
       host: 192.0.2.2
       port: 53
       rndc_host: 192.0.2.2
       rndc_port: 953
```

(continues on next page)

(continued from previous page)

```
        rndc_key_file: /etc/designate/rndc.key
        clean_zonefile: false
```

The key and config files are relative to the host running Designate (and can be different from the hosts running Bind)

Then update the pools in designate - see *designate-manage pool* for further details on the `designate-manage pool` command

```
$ designate-manage pool update
```

Bind9 Configuration

Ensure Bind can access the /etc/bind/rndc.conf and /etc/bind/rndc.key files and receive rndc traffic from Designate.

Enable rndc addzone/delzone functionality by editing named.conf.options or named.conf and add this line under options

```
allow-new-zones yes;
```

Example configuration of /etc/bind/rndc.key

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "<b64-encoded string>";
};
```

Infoblox Backend

Provides an integration between Designate and Infoblox grids.

Features

The Infoblox Designate backend allows an Infoblox grid to be used for serving zones controlled by OpenStack Designate.

The Infoblox backend may be setup to map a specific Designate pool to a single DNS view, or it may be setup to map individual tenants to per-tenant DNS views.

Infoblox Configuration

- Create a user for use by Designate.
- Set up one or more nameserver groups to be used to serve Designate zones.
  - Set the Designate mDNS servers as external primaries
  - Add a grid member as a grid secondary; select the Lead Secondary option for this member
  - Add additional grid secondaries as desired

Designate Backend Configuration

- Designate may be configured to talk to any number of grid API service points (GM or Cloud appliance).
  - Setup a pool for each combination of DNS view and nameserver group you wish to manage.
  - Setup a pool target for each API service point that Designate should talk to.
    * A single Designate pool should point to only one API service point in any single grid. That is, do not point a pool at more than one API service point in the same grid.
    * It is OK to point a pool at multiple grids, just not to multiple service points on the same grid.
    * You may specify the DNS view and nameserver group on a per-target basis.
- The `[infoblox:backend]` stanza in the designate configuration file can be used to set default values for the grid connectivity and other information.
- These values can be overridden on a per-target basis with the options element of the target configuration.
- Set the mDNS port to 53 in the `[service:mdns]` stanza.
- Designate always puts any servers associated with the pool as NS records for the domain. So, if you wish for any Infoblox nameservers to be listed in NS records, they must be added via Designate.

Multi-tenant Configuration

When configured with `multi_tenant = True` in the designate.conf file, the DNS view will be chosen as follows:

- A search will be made for a network view with the EA TenantID, with the value of the OpenStack tenant_id.
- If found, then DNS view used will be <dns_view>.<network_view>, where <dns_view> is the value specified in designate.conf, and <network_view> is the name of the view found in the search.
- If no such network view is found, then a network view will be created with the name <network_view>.<tenant_id>, where <network_view> is the value specified in designate.conf. This network view will be tagged with the TenantID EA.
- If the DNS view does not exist (in either case above), then it will be created.

NS1 Backend

NS1 Configuration

1. Configure the NS1 Backend using this sample target snippet

```
targets:
  - type: ns1
    description: NS1 DNS Server

    # List out the designate-mdns servers from which NS1 servers should
    # request zone transfers (AXFRs) from.
    masters:
```

(continues on next page)

(continued from previous page)

```
        - host: 192.0.2.1
          port: 5354

      # NS1 Configuration options
      options:
        #NS1 XFR container ip and port
        host: 192.0.2.2
        port: 5302
        #NS1 API enpoint IP address or name (Core container). Enter only base␣
↪address or name.
        #Plugin will generate full api address, e.g. https://192.0.2.2/v1/
↪zones/<zone name>
        api_endpoint: 192.0.2.2
        #NS1 API key
        api_token: changeme
        # If a tsigkey is needed, uncomment the line below and insert the key␣
↪name, algorithm and value
        # NOTE: TSIG key has to be set manually
        #tsigkey_name: testkey
        #tsigkey_hash: hmac-sha512
        #tsigkey_value: 4EJz00m4ZWe005HjLiXRedJbSnCUx5Dt+4wVYsBweG5HKAV6cqSVJ/
↪oem/6mLgDNFAlLP3Jg0npbg1SkP7RMDg==
```

2. Then update the pools in designate

```
$ designate-manage pool update
```

See *designate-manage pool* for further details on the `designate-manage pool` command, and *DNS Server Pools* for information about the yaml file syntax

TSIG Key Configuration

In some cases a deployer may need to use tsig keys to sign AXFR (zone transfer) requests. As NS1 does not support a per host key setup, this needs to be set on a per zone basis, on creation.

To do this, generate a tsigkey using any of available utilities (e.g. tsig-keygen):

```
$ tsig-keygen -a hmac-sha512 testkey
key "testkey" {
    algorithm hmac-sha512;
    secret
↪"vQbMI3u5QGUyRu6FWRm16eL0F0dfOOmVJjWKCTg4mIMNnba0g2PLrV+0G92WcTfJrgqZ20a4hv3RWDICKCcJhw==
↪";
};
```

Then insert it into Designate. Make sure the pool id is correct (the `--resource-id` below.)

```
openstack tsigkey create --name testkey --algorithm hmac-sha512 --secret␣
↪4EJz00m4ZWe005HjLiXRedJbSnCUx5Dt+4wVYsBweG5HKAV6cqSVJ/oem/
↪6mLgDNFAlLP3Jg0npbg1SkP7RMDg== --scope POOL --resource-id 794ccc2c-d751-
↪44fe-b57f-8894c9f5c842
```

Then add it to the `pools.yaml` file as shown in the example.

PDNS4 Backend

PDNS4 Configuration

The version PowerDNS in Ubuntu Xenial is pdns4. This has a different DB schema, and is incompatible with the legacy PowerDNS driver. In PDNS 4 the API was marked stable, and this is what we will use.

You will need to configure PowerDNS, and its database before performing these steps.

You will need to use a database backend for PowerDNSs API to function.

See PowerDNS Docs for details.

1. Enable the API in the `pdns.conf` file.

```
webserver=yes
api=yes
api-key=changeme
```

2. Configure the PowerDNS Backend using this sample target snippet

```
targets:
  - type: pdns4
    description: PowerDNS4 DNS Server

    # List out the designate-mdns servers from which PowerDNS servers should
    # request zone transfers (AXFRs) from.
    masters:
      - host: 192.0.2.1
        port: 5354

    # PowerDNS Configuration options
    options:
      host: 192.0.2.1
      port: 53
      api_endpoint: http://127.0.0.1:8081
      api_token: changeme
      api_ca_cert: /etc/ssl/certs/ca-certificates.crt
      # If a tsigkey is needed, uncomment the line below and insert the name
      # tsigkey_name: <keyname>
```

3. Then update the pools in designate

```
$ designate-manage pool update
```

See *designate-manage pool* for further details on the `designate-manage pool` command, and *DNS Server Pools* for information about the yaml file syntax

TSIG Key Configuration

> **Note**
>
> This is only available in PowerDNS 4.2 or newer

In some cases a deployer may need to use tsig keys to sign AXFR (zone transfer) requests. As pdns does not support a per host key setup, this needs to be set on a per zone basis, on creation.

To do this, generate a tsigkey on the PowerDNS Server:

```
$ pdnsutil generate-tsig-key <keyname> hmac-sha512
Create new TSIG key keyname hmac-sha512␣
↪4EJz00m4ZWe005HjLiXRedJbSnCUx5Dt+4wVYsBweG5HKAV6cqSVJ/oem/
↪6mLgDNFAlLP3Jg0npbg1SkP7RMDg==
```

Then insert it into Designate. Make sure the pool id is correct (the `--resource-id` below.)

```
openstack tsigkey create --name <keyname> --algorithm hmac-sha512 --secret␣
↪4EJz00m4ZWe005HjLiXRedJbSnCUx5Dt+4wVYsBweG5HKAV6cqSVJ/oem/
↪6mLgDNFAlLP3Jg0npbg1SkP7RMDg== --scope POOL --resource-id 794ccc2c-d751-
↪44fe-b57f-8894c9f5c842
```

Then add it to the `pools.yaml` file as shown in the example. The ID used is the name of the key in the PowerDNS server.

For a list of drivers and the status of each drivers testing please go to *DNS Server Driver Support Matrix*

1.5.3 High Availability Guide

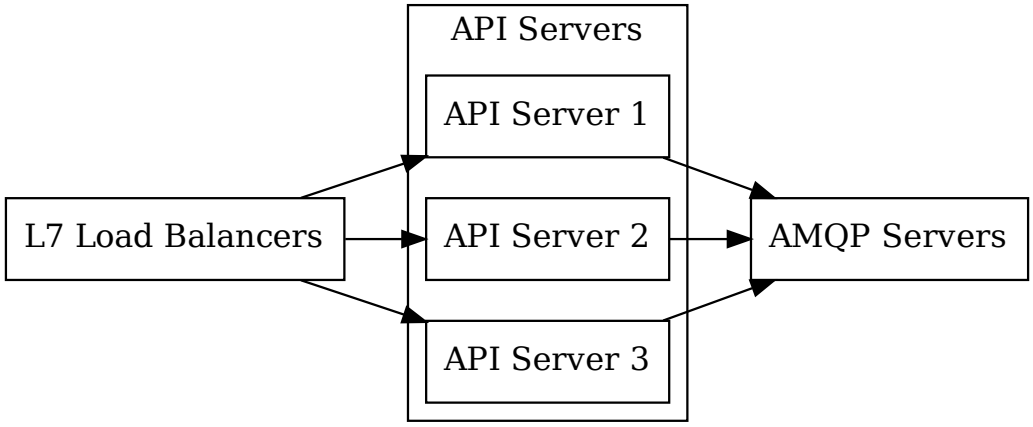
Designate supports running all of its components services in active-active HA modes.

Some services require some extra setup to ensure that they can work in active-active, and the services are listed below.

designate-api

Needs Access to:

- AMQP



Notes

To run multiple *designate-api* services, you should run the services behind a load balancer.

When behind the load balancer, you may need to set the following:

```
[service:api]
api_base_uri = http://<load balancer URI>/
enable_host_header = True
```

Or the following:

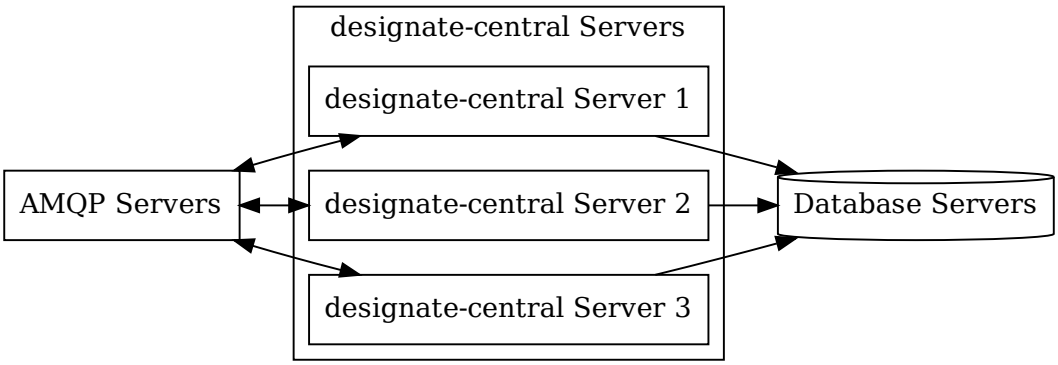
```
[oslo_middleware]
enable_proxy_headers_parsing = true
```

And then the load balancer to set appropriate headers (e.g. enable *mod_proxy* in apache.)

designate-central

Needs Access to:

- AMQP
- Database



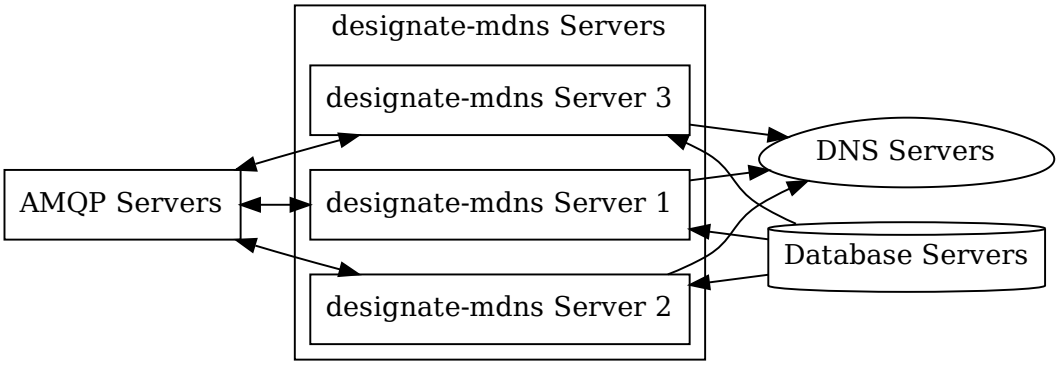
Notes

You can run as many *designate-central* services as needed, as long as they all have access to the AMQP server(s), work will be distributed across all of them.

designate-mdns

Needs Access to:

- AMQP
- Database
- DNS Servers



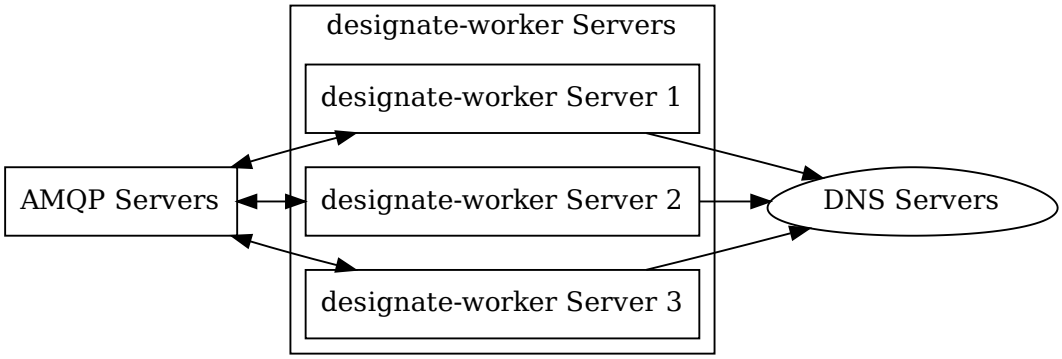
Notes

You can run as many *designate-mdns* services as needed, as long as they all have access to the AMQP server(s), work will be distributed across all of them.

designate-worker

Needs Access to:

- AMQP
- DNS Servers



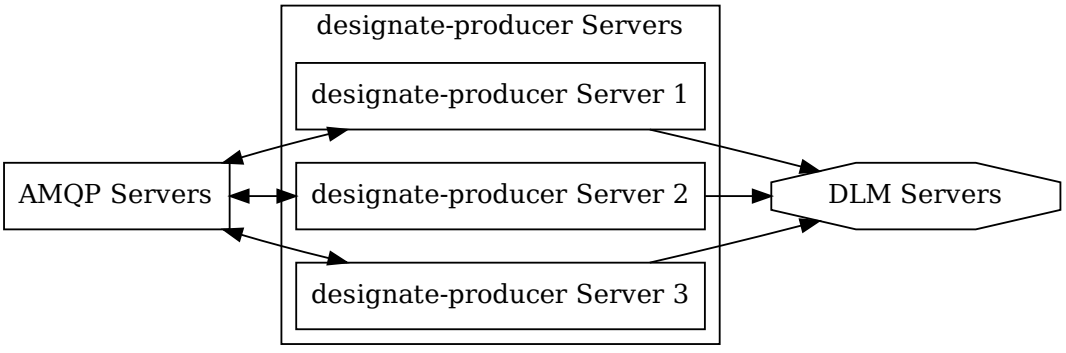
Notes

You can run as many *designate-worker* services as needed, as long as they all have access to the AMQP server(s), work will be distributed across all of them.

designate-producer

Needs Access to:

- AMQP
- DLM



Notes

You can run as many *designate-producer* services as needed, as long as they all have access to the AMQP server(s), and a distributed lock manager, work will be sharded across all the services.

You will need to set a coordination *backend_url*. This needs to be a DLM that is supported by tooz, that supports group membership. See tooz driver list for available drivers

Warning

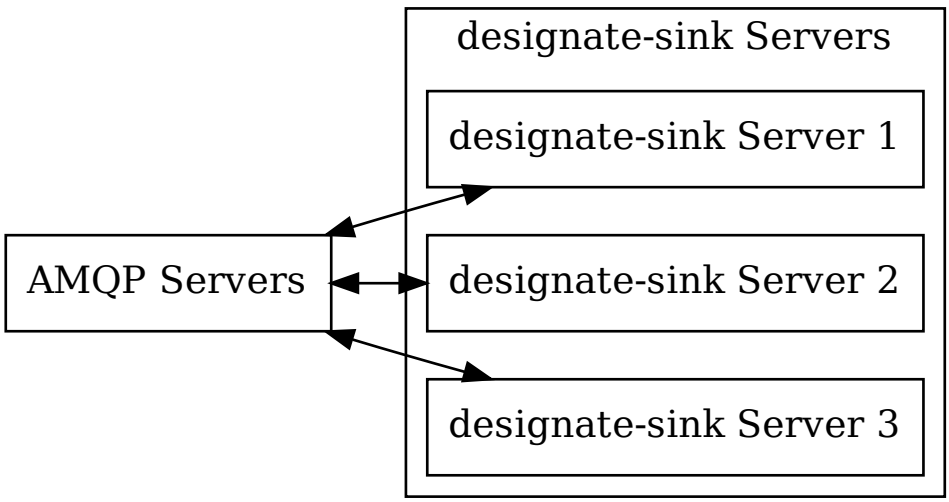
Failure to set a *backend_url* can cause unexpected consequences, and may result in some periodic tasks being ran more than once.

```
[coordination]
backend_url = kazoo://<zookeeper url>:<zookeeper port>
```

designate-sink

Needs Access to:

- AMQP



Notes

You can run as many *designate-sink* services as needed, as long as they all have access to the AMQP server(s), work will be distributed across all of them.

1.5.4 DNS Server Pools

About Pools

Administrators can group DNS namespace servers into multiple pools to help them manage their DNS environments. You can provide users with tiers of service, by configuring pools to offer more capacity and better geographical proximity. Administrators of private clouds can leverage multiple pools to separate internal and external facing zones.

By default, Designate contains only one DNS server pool called `default`. When users create a zone in a multi-pool environment, the pool scheduler must select a pool to host the new zone. Administrators control pool selection through the use of filters, that the scheduler uses to select a pool for the new zone.

The filter interface consists of pool attributes that are key-value pairs that the scheduler attaches to the zone during creation. Administrators can update pool attributes later, but none of the updates will trigger zones to move to another pool. Zones can be moved manually to a different pool, as mentioned in the API Reference.

Designate provides a set of filters that reflect some common use cases and also a simple interface that administrators can use to create custom filters.

Process Overview for Configuring Multiple Pools

The process of configuring multiple DNS server pools in Designate, consists of the following steps:

1. Define the new pool in the pool definition file.
2. Load the new pool definition into the Designate database by running the `designate-manage` command.
3. Configure the pool scheduler to use one or more filters to match any new zones that users create with the appropriate pool. You can choose filters that are provided with Designate, or create new filters.
4. Supply the required pool information to users to specify when they create zones.

Pool Definition File

A pool definition file is required when you create a DNS server pool in Designate. The required key value pairs in YAML format are documented here:

```
---

- name: default
  # The name is immutable. There will be no option to change the name after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default PowerDNS Pool


  # Attributes are Key:Value pairs that describe the pool. for example the␣
↪level
  # of service (i.e. service_tier:GOLD), capabilities (i.e. anycast: true) or
  # other metadata. Users can use this information to point their zones to the
  # correct pool
  attributes: {}
```

(continues on next page)

(continued from previous page)

```
  # List out the NS records for zones hosted within this pool
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1
    - hostname: ns1-2.example.org.
      priority: 2

  # List out the nameservers for this pool. These are the actual PowerDNS
  # servers. We use these to verify changes have propagated to all␣
↪nameservers.
  nameservers:
    - host: 192.0.2.2
      port: 53

  # List out the targets for this pool. For PowerDNS, this is the database
  # (or databases, if you deploy a separate DB for each PowerDNS server)
  targets:
    - type: powerdns
      description: PowerDNS Database Cluster

      # List out the designate-mdns servers from which PowerDNS servers should
      # request zone transfers (AXFRs) from.
      masters:
        - host: 192.0.2.1
          port: 5354

      # PowerDNS Configuration options
      options:
        host: 192.0.2.2
        port: 53
        connection: 'mysql+pymysql://designate:password@127.0.0.1/designate_
↪pdns?charset=utf8'

  # Optional list of additional IP/Port's for which designate-mdns will send
  # DNS NOTIFY packets to
  also_notifies:
   - host: 192.0.2.4
     port: 53

  # Optional configuration to provide a catalog zone for the pool's zones.
  # If configured, catalog_zone_fqdn is required and all other keys are
  # optional.
  catalog_zone:
      catalog_zone_fqdn: cat.example.org.
      catalog_zone_refresh: 60
      # TSIG secret and algorithm to use for securing AXFRs for catalog zones.
      catalog_zone_tsig_key: SomeSecretKey
      catalog_zone_tsig_algorithm: hmac-sha512
```

Pools.yaml attributes

NS records

The ns_records section is the list of name servers Designate will advertise in the zones as available for query. Nameservers listed in the ns_records section are expected to be advertised from external networks.

Nameservers

The nameservers section is the list of name servers Designate will query to confirm an update has completed on all of the Designate managed nameservers. Nameservers listed in the nameservers section are not expected to be advertised from external networks unless they are also listed in the ns_records section.

NS Records vs. Nameservers: Understanding the Differences

There isnt always a direct relationship between the ns_records and the nameservers sections, as they serve different use cases. Here are a few use cases:

- **ns_records list is equal to the nameservers list:**
  The end user may want to query all of the advertised nameservers to confirm that their updates have successfully propagated.
- **ns_records list is smaller than the nameservers list:**
  The end user may have some private, or stealth, nameservers that they do not want to advertise publicly as ns_records. Because they are private, these stealth nameservers are not expected be necessarily reachable from external networks.
- **ns_records list is larger than the nameservers list:**
  The end user may not want to query all of the nameservers they manage after a zone update as some of those nameservers might be backup, or maybe the list of nameservers is just too big to query all of them.

Targets

The targets section defines how Designate should communicate with the backend nameservers and how those backend nameservers should communicate with MiniDNS. For the BIND driver, the options section defines the address and port the NOTIFY messages should go to and the IP:port the driver should use to make rndc calls to BIND, as can be seen in *this example.* PowerDNS require using the connection keyword, as can be seen above.

Catalog zones

Catalog zones provide easy provisioning capabilities of zones to secondary nameservers, transferred via AXFR from a special zone, the *catalog zone*.

In Designate, catalog zones are configured per pool. A catalog zone will include all zones from the pool (except the catalog zone itself), called *member zones*. That means all zones from that pool are automatically synced to secondary name servers upon zone creation, update or deletion. For more details about catalog zones, see RFC 9432.

Catalog zones can be configured in `pools.yaml` via the *catalog_zone* key (see the sample above). This example instructs a PowerDNS server listening at `192.0.2.2:53` to pull zones via AXFR from Designates `mini-DNS` at `192.0.2.1:5354`. Note that the secondary nameserver also needs to be properly configured to consume the catalog zone. Please refer to the secondary nameservers documentation for

details. Once this is set up and applied using `designate-manage pool update`, Designate will handle the catalog zone creation as well as synchronization of member zones.

As secondary nameservers configure their zones based on zone transfers (AXFR) from the catalog zone, it is highly recommended to use transaction signatures (TSIG) for secure and authenticated zone transfers. See the above sample for details on how to use catalog zones with TSIG.

Warning

Even though not mandatory, it is highly recommended to secure transfers of catalog zones with TSIG.

designate-manage pool Command Reference

You manage pools in Designate with the `designate-manage pool` commands.

Note

Control plane does not need to be restarted after designate-manage pool command changes.

Pool update

You manage can modify the current deployed pools with the `designate-manage pool update` command.

```
designate-manage pool update [options]
```

Pool update options

`--file <file>`
: Input file. When a file is not specified, `/etc/designate/pools.yaml` is used by default.

`--dry-run`
: Simulates what happens when you run this command.

`--delete`
: Removes all pools not listed in the config file.

Warning

Using `--delete` can be **extremely** dangerous, because `designate-manage` removes any pools that are not in the supplied YAML file, **including the default one** and any zones that are in those pools. Before using `--delete`, use `--delete --dry-run` to view the outcome.

Generating a Copy of the Pool Configuration

```
designate-manage pool generate_file [options]
```

Options

--file <file>
The YAML file where `designate-manage` writes its output. When a file is not specified, `designate-manage` writes to `/etc/designate/pools.yaml`.

Showing the current Pools Configuration

```
designate-manage pool show_config [options]
```

Options

--pool_id <pool_id>
ID of the pool to be examined.

--all_pools
Show the config of all the pools.

1.5.5 Pool Scheduler Filters

About Filters

When a user creates a zone, the pool scheduler uses filters to assign the zone to a particular DNS server pool. As the administrator, you choose an ordered list of filters that runs on each `zone create` API request. You configure the scheduler to use filters that are provided with Designate or create your own.

Filters Provided with Designate

Designate provides several filters that represent common use cases.

Base Class - Filter

class designate.scheduler.filters.base.**Filter**(*storage*)

This is the base class used for filtering Pools.

This class should implement a single public function *filter()* which accepts a *designate.objects.pool.PoolList* and returns a *designate.objects.pool.PoolList*

abstract filter(*context*, *pools*, *zone*)

Filter list of supplied pools based on attributes in the request

Parameters

- **context** designate.context.DesignateContext - Context Object from request
- **pools** *designate.objects.pool.PoolList* - List of pools to choose from
- **zone** *designate.objects.zone.Zone* - Zone to be created

Returns
designate.objects.pool.PoolList - Filtered list of Pools

Attribute Filter

class designate.scheduler.filters.attribute_filter.**AttributeFilter**(*storage*)

Bases: *Filter*

This allows users to choose the pool by supplying hints to this filter. These are provided as attributes as part of the zone object provided at zone create time.

```
{
    "attributes": {
        "pool_level": "gold",
        "fast_ttl": "true",
        "pops": "global",
    },
    "email": "user@example.com",
    "name": "example.com."
}
```

The zone attributes are matched against the potential pool candidates, and any pools that do not match **all** hints are removed.

Warning

This should be uses in conjunction with the `designate.scheduler.impl_filter.filters.random_filter.RandomFilter` in case of multiple Pools matching the filters, as without it, we will raise an error to the user.

name = 'attribute'

Name to enable in the `[designate:central:scheduler].filters` option list

Pool ID Attribute Filter

class designate.scheduler.filters.pool_id_attribute_filter.**PoolIDAttributeFilter**(*storage*)

Bases: *Filter*

This allows users with the correct role to specify the exact pool_id to schedule the supplied zone to.

This is supplied as an attribute on the zone

```
{
    "attributes": {
        "pool_id": "794ccc2c-d751-44fe-b57f-8894c9f5c842"
    },
    "email": "user@example.com",
    "name": "example.com."
}
```

The pool is loaded to ensure it exists, and then a policy check is performed to ensure the user has the correct role.

Warning

This should only be enabled if required, as it will raise a 403 Forbidden if a user without the correct role uses it.

`filter`(*context*, *pools*, *zone*)

Attempt to load and set the pool to the one provided in the Zone attributes.

Parameters

- **context** `designate.context.DesignateContext` - Context Object from request
- **pools** *designate.objects.pool.PoolList* - List of pools to choose from
- **zone** *designate.objects.zone.Zone* - Zone to be created

Returns

designate.objects.pool.PoolList A PoolList with containing a single pool.

Raises

Forbidden, PoolNotFound

`name = 'pool_id_attribute'`

Name to enable in the `[designate:central:scheduler].filters` option list

Random Filter

`class designate.scheduler.filters.random_filter.RandomFilter`(*storage*)

Bases: *Filter*

Randomly chooses one of the input pools if there are multiple ones supplied.

Note

This should be used as one of the last filters, as it reduces the supplied pool list to one.

`name = 'random'`

Name to enable in the `[designate:central:scheduler].filters` option list

Fallback Filter

`class designate.scheduler.filters.fallback_filter.FallbackFilter`(*storage*)

Bases: *Filter*

If there is no zones available to schedule to, this filter will insert the default_pool_id.

> **Note**
>
> This should be used as one of the last filters, if you want to preserve behavior from before the scheduler existed.

name = 'fallback'

Name to enable in the `[designate:central:scheduler].filters` option list

Default Pool Filter

class designate.scheduler.filters.default_pool_filter.**DefaultPoolFilter**(*storage*)

Bases: *Filter*

This filter will always return the default pool specified in the designate config file

> **Warning**
>
> This should be used as the only filter, as it will always return the same thing - a *designate.objects.pool.PoolList* with a single *designate.objects.pool.Pool*

name = 'default_pool'

Name to enable in the `[designate:central:scheduler].filters` option list

In Doubt Default Pool Filter

class designate.scheduler.filters.in_doubt_default_pool_filter.**InDoubtDefaultPoolFilter**(*stor*

Bases: *Filter*

If the previous filter(s) didnt make a clear selection of one pool and if the default pool is in the set of multiple pools, this filter will select the default pool.

This filter will pass through the pool list, if there are one or less pools available to schedule to, or if the default pool is not in the set of multiple pools.

> **Note**
>
> This should be used as one of the last filters.

name = 'in_doubt_default_pool'

Name to enable in the `[designate:central:scheduler].filters` option list

Creating Custom Filters

You can create your own filters by extending *designate.scheduler.filters.base.Filter* and registering a new entry point in the `designate.scheduler.filters` namespace in `designate.conf`:

```
[entry_points]
designate.scheduler.filters =
my_custom_filter = my_extension.filters.my_custom_filter:MyCustomFilter
```

Configuring Filters in the Scheduler

After you have decided whether to use the filters provided with Designate or create custom filters you must configure the filters in the pool scheduler.

Inside the `designate.conf` file under the `[service:central]` section, add the filters that you want the scheduler to use to the `scheduler_filters` parameter:

```
[service:central]
scheduler_filters = attribute, pool_id_attribute, fallback, random, my_custom_
↪filter
```

Important

The scheduler runs the filters list from left to right.

1.5.6 Configuring Multiple Pools

Administrators can group DNS namespace servers into multiple pools to help them manage their DNS environments. See *DNS Server Pools* to learn more about what pools are and how you can use them.

Configuring Designate to use multiple pools consists of:

1. Defining new pools and loading their definitions into the database.
2. Configuring the pool scheduler with filters that you have created or with filters provided by Designate.
3. Supplying the required pool information to users to specify when they create zones.

Defining New Pools

In Designate, you define a new pool in a pool definition file, and then load the definition into the Designate database by running the `designate-manage` command.

1. Add the pool to the pool definition file, by following the required key value pairs in YAML format that are documented under Pool Definition File in *DNS Server Pools*.

   Here is an example of a pool definition file, `pools.yaml`, that configures two different pools. Each pool supports a different usage level, *gold* and *standard*, and each contains zones that reflect their respective usage levels.

   The *gold* level provides 6 nameservers that users have access to. The *standard* level provides only 2 nameservers. Both pools have one target that is written to.

```
---

- name: golden_pool
  description: The golden pool!

  attributes:
    service_tier: gold

  ns_records:
    - hostname: ns1-gold.example.org
```

(continues on next page)

(continued from previous page)

```
    priority: 1

  - hostname: ns2-gold.example.org
    priority: 2

  - hostname: ns3-gold.example.net
    priority: 3

  - hostname: ns4-gold.example.net
    priority: 4

  - hostname: ns5-gold.example.net
    priority: 5

  - hostname: ns6-gold.example.net
    priority: 6

nameservers:
  - host: ns1-gold.example.net
    port: 53

  - host: ns2-gold.example.net
    port: 53

  - host: ns3-gold.example.net
    port: 53

  - host: ns4-gold.example.net
    port: 53

  - host: ns5-gold.example.net
    port: 53

  - host: ns6-gold.example.net
    port: 53

targets:
  - type: bind9
    description: bind9 golden master

    masters:
      - host: mdns.designate.example.com
        port: 5354

    options:
      host: ns-master-gold.example.org
      port: 53
      rndc_host: ns-master-gold.example.org
      rndc_port: 953
```

(continues on next page)

(continued from previous page)

```
      rndc_key_file: /etc/designate.rndc.key


- name: standard_pool
  description: The standard pool

  attributes:
    service_tier: standard

  ns_records:
    - hostname: ns1-std.example.org
      priority: 1

    - hostname: ns2-std.example.org
      priority: 2

  nameservers:
    - host: ns1-std.example.net
      port: 53

    - host: ns2-std.example.net
      port: 53

  targets:
    - type: bind9
      description: bind9 golden master

      masters:
        - host: mdns.designate.example.com
          port: 5354

      options:
        host: ns-master-std.example.org
        port: 53
        rndc_host: ns-master-std.example.org
        rndc_port: 953
        rndc_key_file: /etc/designate.rndc.key
```

2. Load the definitions into the Designate database using the `designate-manage pool update` command:

```
# Do a dry run
$ designate-manage pool update --file pools.yaml --dry-run
$ designate-manage pool update --file pools.yaml
```

Designate now has two pools to work with. The next step is to configure the pool scheduler to use the attributesprovided through filterswhen choosing the pool to store the zone on.

Showing the configured pools

In Designate, you can show the current configured default pool by running the `designate-manage pool show_config` command. You can either see a different pool by adding pool_id <POOL_ID>, or you can see all the configured pools by adding `--all_pools` or just `--all`.

Configuring the Pool Scheduler

When a user creates a zone, the pool scheduler uses filters to assign the zone to a particular DNS server pool. As the administrator, you choose an ordered list of filters that runs on each `zone create` API request. You configure the scheduler to use filters that are provided with Designate or create your own.

1. Do one of the following:
   - Write one or more custom filters.

     See *Pool Scheduler Filters*.
   - Choose one or more of the filters that Designate provides:
     - `attribute`assigns the zone to the pool whose attribute is specified.
     - `pool_id_attribute`if the user is a member of the specified role assigns the zone to the pool whose ID is specified.
     - `default_pool`assigns the zone to the default pool specified in the Designate configuration file.
     - `fallback`if there are no pools available, assigns the zone to the default pool.
     - `random`if multiple pools have been specified, randomly assigns the zone to a pool.
     - `in_doubt_default_pool`if none of the specified pools are available, and the default pool has not been specified, assigns the zone to the default pool.
2. Add the filters that you want the scheduler to use in the `service:central` section of the `designate.conf` file. See *Pool Scheduler Filters* for more information.

Schedule by Pool ID Example

For example, to allow a user to select a pool by specifying an ID or fallback to using a default, you could use the following configuration:

```
[service:central]
default_pool_id = 794ccc2c-d751-44fe-b57f-8894c9f5c842
scheduler_filters = pool_id_attribute, fallback
```

The pool scheduler applies filters from left to right. If the zone body doesnt contain an *attributes* object with a *pool_id* set to a valid pool ID, the fallback filter is then called, returning the default pool as the scheduled pool for that zone.

Schedule by Tier Example

In this tier example, the *attribute* filter is used to select the correct pool.

```
[service:central]
default_pool_id = 794ccc2c-d751-44fe-b57f-8894c9f5c842  # the std pool
scheduler_filters = attribute, fallback
```

When a user wants to assign a zone to the *gold* pool, the user must provide the appropriate attribute in the zone.

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
    "attributes": {
        "service_tier": "gold"
    },
    "email": "user@example.com",
    "name": "example.net."
}
```

In this example, the user defines which pool is scheduled. If the zone should be scheduled based on the tenant, a custom filter could be written that looks up the appropriate group and adds the appropriate pool.

1.5.7 Blacklisting Domain Names

Note

The blacklist feature will be renamed and moved to denylist in the near future.

You can prevent users from creating zones with names that match a particular regular expression using blacklists. For example, you might use a blacklist to prevent users from:

- creating a specific zone.
- creating zones that contain a certain string,
- creating subzones of a certain zone.

Managing Blacklists

You can create blacklists using the `zone blacklist create` command with System Administrator privileges. For example, to blacklist `example.com.` and all of its subdomains:

```
$ openstack zone blacklist create --pattern ".*example.com."
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2021-05-27T04:06:42.000000           |
| description | None                                 |
| id          | 7622e241-8c3d-4c03-a692-8747e3cf2658 |
| pattern     | .*example.com.                       |
| updated_at  | None                                 |
+-------------+--------------------------------------+
```

If a Domain or Project Persona attempts to create `foo.example.com.`, or `example.com.`, they encounter an error:

```
$ openstack zone create --email admin@example.com example.com.
Blacklisted zone name
$ openstack zone create --email admin@example.com foo.example.com.
Blacklisted zone name
```

Note

Users who satisfy the `use_blacklisted_zone` policy can create zones with names that are on a blacklist. By default, the only users who have this override are System Administrators.

You can update a blacklist using `zone blacklist set` to modify its pattern or description;

```
$ openstack zone blacklist set 81fbfe02-6bf9-4812-a40e-1522ab6862ca --pattern
↪".*web.example.com"
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2021-05-27T04:14:14.000000           |
| description | None                                 |
| id          | 81fbfe02-6bf9-4812-a40e-1522ab6862ca |
| pattern     | .*web.example.com                    |
| updated_at  | 2021-05-27T04:14:48.000000           |
+-------------+--------------------------------------+
```

You can delete a blacklist using *zone blacklist delete*:

```
$ openstack zone blacklist delete 7622e241-8c3d-4c03-a692-8747e3cf2658
```

There is no output when this command is successful.

Using the REST API

The regular expressions used for blacklists are similar to Python regular expressions, but you must escape certain characters when making HTTP calls.

For examples, this refex restricts using `example.com.` and its ASCII subdomains:

`^([A-Za-z0-9_\\-]+\.)*example\.com\.$`

However, you must insert the escape character (backslash,) before the instances of dot (.) and .com:

`^([A-Za-z0-9_\\-]+\\.)*example\\.com\\.$`

Here is the API call and the regex with the HTTP characters escaped:

```
POST /v2/blacklists/ HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "pattern" : "^([A-Za-z0-9_\\-]+\\.)*example\\.com\\.$",
  "description" : "This blacklists *.example.com."
}
```

Regular Expressions

Regular Expressions can be difficult to work with. The Python Regex Documentation may serve as a useful introduction, and online regular expression tools can assist when building and testing regexes for use with the blacklist API.

1.5.8 View and Manage Quotas

Quotas exist in Designate for various resources. You can configure quotas globally or on a per-project basis.

Viewing Quotas

The Designate plugin for the OpenStack Client allows users to query their current quota using the `dns quota list` command.

```
$ openstack dns quota list
+-------------------+-------+
| Field             | Value |
+-------------------+-------+
| api_export_size   | 1000  |
| recordset_records | 20    |
| zone_records      | 500   |
| zone_recordsets   | 500   |
| zones             | 10    |
+-------------------+-------+
```

Users can also view their quotas with a simple View Current Projects Quotas Designate API call:

```
GET /v2/quotas/ HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-bfcd0723-624c-4ec2-bbd5-99e985efe8db

{
  "api_export_size": 1000,
  "recordset_records": 20,
  "zone_records": 500,
  "zone_recordsets": 500,
  "zones": 10
}
```

Administrators with a cross-project read role can query the quotas for other projects using the `--project-id` option to the `dns quota list` command or by specifying a project_id when making the View Quotas Designate API call.

```
$ openstack dns quota list --project-id ecd4341280d645e5959d32a4b7659da1
+-------------------+-------+
```

(continues on next page)

(continued from previous page)

```
| Field              | Value |
+--------------------+-------+
| api_export_size    | 1000  |
| recordset_records  | 20    |
| zone_records       | 500   |
| zone_recordsets    | 500   |
| zones              | 20    |
+--------------------+-------+
```

```
GET /v2/quotas/ecd4341280d645e5959d32a4b7659da1 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Modifying Quotas

You can edit Designate quotas on a per-project basis. An administrator can edit quotas for any project, but they must have an *all_tenants* role or use a system scoped admin token.

Administrators can set a custom quota for a project using the OpenStack Client `dns quota set` command.

```
$ openstack dns quota set --project-id ecd4341280d645e5959d32a4b7659da1 --
↪zones 30
+--------------------+-------+
| Field              | Value |
+--------------------+-------+
| api_export_size    | 1000  |
| recordset_records  | 20    |
| zone_records       | 500   |
| zone_recordsets    | 500   |
| zones              | 30    |
+--------------------+-------+
```

Below is an example of setting a quota using the Set Quotas Designate API.

```
PATCH /v2/quotas/ecd4341280d645e5959d32a4b7659da1 HTTP/1.1
Accept: application/json
Content-Type: application/json
X-Auth-All-Projects: True

{
  "zones": 30
}
```

The response would be:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-ee264c7d-d9f3-4de8-92ec-7de4dc93a255
```

(continues on next page)

(continued from previous page)

```
{
  "api_export_size": 1000,
  "recordset_records": 20,
  "zone_records": 500,
  "zone_recordsets": 500,
  "zones": 30
}
```

Resetting Quotas

You can reset custom quotas for a project to their default values by using the `dns quota reset` command. Administrators can reset quotas for any project, but they must have an *all_tenants* role or use a system scoped admin token.

```
$ openstack dns quota reset --project-id ecd4341280d645e5959d32a4b7659da1
```

Note

There is no output from a successful `dns quota reset` command.

Below is an example of resetting a projects quota via the Reset Quota Designate API.

```
DELETE /v2/quotas/ecd4341280d645e5959d32a4b7659da1 HTTP/1.1
Accept: application/json
Content-Type: application/json
X-Auth-All-Projects: True
```

The response would be:

```
HTTP/1.1 204 No Content
X-Openstack-Request-Id: req-82b85853-145d-4253-be86-b9aa3116b975
```

Available Quotas

The quotas available in Designate are listed below with a short description and the default values.

Zones

Quota	Description	Default
zones	The number of zone allowed per project	10

Recordsets/Records

Quota	Description	Default
zone_recordsets	Number of recordsets allowed per zone	500
zone_records	Number of records allowed per zone	500
recordset_records	Number of records allowed per recordset	20

Zone Exports

Quota	Description	Default
api_export_size	Number of recordsets allowed in a zone export	1000

Default Quotas

You can set a default value for each quota that applies to all users by editing the `[DEFAULT]` configuration section of the `designate.conf` file, for example:

```
[DEFAULT]
########################
## General Configuration
########################
quota_zones = 10
quota_zone_recordsets = 500
quota_zone_records = 500
quota_recordset_records = 20
quota_api_export_size = 1000
```

Project ID Verification

Although Designate API can accept arbitrary strings as the Project ID to set the quota for, actual enforcement of quota will be performed only when the project ID of the quota matches the `project-id` in the request that attempts to create a resource.

To prevent mistakes when specifying the `project-id` for a quota, you can turn on project ID verification in the Designate configuration file:

```
[service:api]
quotas_verify_project_id = True
```

You must also specify how Designate connects to Keystone and locates the appropriate Keystone endpoint with which to perform requests. In the `[keystone]` section, ensure that the Session- and Adapter-related options are set.

Here is an example:

```
[keystone]
cafile = /path/to/ca/bundle
valid_interfaces = internal,public
region_name = RegionWest
```

See keystoneauth documentation for more details.

With project ID verification enabled, Designate will use the credentials provided with the request to attempt to verify that the project ID is valid in Keystone.

As a result of this verification, the request might return additional errors in these cases:

- when the Keystone V3 endpoint could not be found in the service catalog (as specified in `[keystone]` section) - `504` error is returned

- when the authentication with incoming token was successful but the project id was not actually found - `400` is returned

For project ID validation to be successful, the user setting quotas should have permission to list projects in Keystone. If the user does not have permission to list projects in Keystone, the validation will be skipped.

1.5.9 Designate Policies

Warning

JSON formatted policy file is deprecated since Designate 12.0.0 (Wallaby). This oslopolicy-convert-json-to-yaml tool will migrate your existing JSON-formatted policy file to YAML in a backward-compatible way.

Designate, like most OpenStack services, supports Role Based Access Control (RBAC) using oslo policy to define default RBAC policies in the Designate code. These default policies can be overridden by operators using a yaml policy file. For a sample policy file, refer to *policy.yaml*.

Currently Designate defaults to the OpenStack legacy admin or owner scheme, but Designate also supports a newer RBAC model using Keystone Default Roles and Keystone Scoped Tokens via configuration settings.

Enabling Keystone Default Roles and Scoped Tokens

Starting with the Xena release of Designate, Keystone token scopes and default roles can be enforced. By default, in the Xena release, oslo policy will not be enforcing these new roles and scopes. However, at some point in the future they may become the default. You may want to enable them now to be ready for the later transition. This section will describe those settings.

The Oslo Policy project defines two configuration settings, among others, that can be set in the Designate configuration file to influence how policies are handled by Designate. Those two settings are enforce_scope and enforce_new_defaults.

When you enable Keystone Default Roles and Keystone Scoped Tokens the Designate policy honors the following roles:

- Admin
- Project scoped - Reader
- Project scoped - Member

[oslo_policy] enforce_scope

Keystone has introduced the concept of token scopes. To ensure backward compatibility, Oslo Policy does not enforce scope validation of tokens by default.

In the Xena release, Designate supports enforcing Keystone token scopes. To enable Keystone token scoping, add the following to your Designate configuration file:

```
[oslo_policy]
enforce_scope = True
```

The primary effect of this setting is to allow only project scoped calls to the Designate API. The system scope token will return 403.

[oslo_policy] enforce_new_defaults

The Designate Xena release added support for Keystone Default Roles in the default policies. To be backward compatible, Oslo Policy currently uses deprecated policies that do not require the new Keystone Default Roles by default.

Designate supports requiring these new Keystone Default Roles as of the Xena release. To start requiring these roles in Designate, enable the new policies by adding the following setting to your Designate configuration file:

```
[oslo_policy]
enforce_new_defaults = True
```

Oslo Tools For Policy Management

This section describes how to use Oslo Policy tools to managing Designate policies.

Sample File Generation

To generate a sample policy.yaml file from the Designate defaults, run the oslo policy generation script:

```
oslopolicy-sample-generator
--config-file etc/designate/designate-policy-generator.conf
--output-file policy.yaml.sample
```

Merged File Generation

To generate a policy file which shows the effective policy in use by the project, including all registered policy defaults and the policy overrides included in a policy.yaml file, run this command:

```
oslopolicy-policy-generator
--config-file etc/designate/designate-policy-generator.conf
```

This tool uses the output_file path from the config-file.

List Redundant Configurations

To generate a list of matches for policy rules that are defined in a configuration file where the rule does not differ from a registered default rule, run this command:

```
oslopolicy-list-redundant
--config-file etc/designate/designate-policy-generator.conf
```

These are rules that can be removed from the policy file with no change in effective policy.

Designate Default Policy Overview

The following is an overview of all available policies in Designate. For a sample configuration file, refer to *policy.yaml*.

designate

admin

Default
`role:admin or is_admin:True`

(no description provided)

owner

Default
`project_id:%(tenant_id)s`

(no description provided)

admin_or_owner

Default
`rule:admin or rule:owner`

(no description provided)

default

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

(no description provided)

create_blacklist

Default
`role:admin`

Operations

- **POST** `/v2/blacklists`

Scope Types

- **project**

Create blacklist.

find_blacklists

Default
`role:admin`

Operations

- **GET** `/v2/blacklists`

Scope Types

- **project**

Find blacklists.

`get_blacklist`

Default
`role:admin`

Operations

- **GET** `/v2/blacklists/{blacklist_id}`

Scope Types

- **project**

Get blacklist.

`update_blacklist`

Default
`role:admin`

Operations

- **PATCH** `/v2/blacklists/{blacklist_id}`

Scope Types

- **project**

Update blacklist.

`delete_blacklist`

Default
`role:admin`

Operations

- **DELETE** `/v2/blacklists/{blacklist_id}`

Scope Types

- **project**

Delete blacklist.

`use_blacklisted_zone`

Default
`role:admin`

Operations

- **POST** `/v2/zones`

Scope Types

- **project**

Allowed bypass the blacklist.

`all_tenants`

Default
`role:admin`

Scope Types

- **project**

Action on all tenants.

`edit_managed_records`

Default
`role:admin`

Scope Types

- **project**

Edit managed records.

`use_low_ttl`

Default
`role:admin`

Scope Types

- **project**

Use low TTL.

`use_sudo`

Default
`role:admin`

Scope Types

- **project**

Accept sudo from user to tenant.

`hard_delete`

Default
`role:admin`

Scope Types

- **project**

Clean backend resources associated with zone

`create_pool`

Default
`role:admin`

Scope Types

- **project**

Create pool.

`find_pools`

Default
`role:admin`

Operations

- **GET** /v2/pools

Scope Types

- **project**

Find pool.

`find_pool`

Default
role:admin

Operations

- **GET** /v2/pools

Scope Types

- **project**

Find pools.

`get_pool`

Default
role:admin

Operations

- **GET** /v2/pools/{pool_id}

Scope Types

- **project**

Get pool.

`update_pool`

Default
role:admin

Scope Types

- **project**

Update pool.

`delete_pool`

Default
role:admin

Scope Types

- **project**

Delete pool.

`zone_create_forced_pool`

Default
role:admin

Operations

- **POST** /v2/zones

Scope Types

- **project**

load and set the pool to the one provided in the Zone attributes.

`get_quotas`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s) or
(True:%(all_tenants)s and role:reader)
```

Operations

- **GET** /v2/quotas

Scope Types

- **project**

View Current Projects Quotas.

`set_quota`

Default

```
role:admin
```

Operations

- **PATCH** /v2/quotas/{project_id}

Scope Types

- **project**

Set Quotas.

`reset_quotas`

Default

```
role:admin
```

Operations

- **DELETE** /v2/quotas/{project_id}

Scope Types

- **project**

Reset Quotas.

`find_records`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/reverse/floatingips/{region}:{floatingip_id}
- **GET** /v2/reverse/floatingips

Scope Types

- **project**

Find records.

`count_records`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Scope Types

- **project**

(no description provided)

`create_recordset`

Default

```
(role:member and project_id:%(project_id)s) and
('PRIMARY':%(zone_type)s) or (role:admin) and
('PRIMARY':%(zone_type)s) or (role:admin) and
('SECONDARY':%(zone_type)s) or ('True':%(zone_shared)s) and
('PRIMARY':%(zone_type)s)
```

Operations

- **POST** `/v2/zones/{zone_id}/recordsets`

Scope Types

- **project**

Create Recordset

`get_recordsets`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Scope Types

- **project**

(no description provided)

`get_recordset`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s) or
('True':%(zone_shared)s)
```

Operations

- **GET** `/v2/zones/{zone_id}/recordsets/{recordset_id}`

Scope Types

- **project**

Get recordset

`find_recordset`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Scope Types

- **project**

List a Recordset in a Zone

`find_recordsets`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/zones/{zone_id}/recordsets

Scope Types

- **project**

List Recordsets in a Zone

`update_recordset`

Default

```
(role:member and project_id:%(project_id)s)
and ('PRIMARY':%(zone_type)s) or (role:admin)
and ('PRIMARY':%(zone_type)s) or (role:admin)
and ('SECONDARY':%(zone_type)s) or role:member
and (project_id:%(recordset_project_id)s) and
('PRIMARY':%(zone_type)s)
```

Operations

- **PUT** /v2/zones/{zone_id}/recordsets/{recordset_id}

Scope Types

- **project**

Update recordset

`delete_recordset`

Default

```
(role:member and project_id:%(project_id)s)
and ('PRIMARY':%(zone_type)s) or (role:admin)
and ('PRIMARY':%(zone_type)s) or (role:admin)
and ('SECONDARY':%(zone_type)s) or role:member
and (project_id:%(recordset_project_id)s) and
('PRIMARY':%(zone_type)s)
```

Operations

- **DELETE** /v2/zones/{zone_id}/recordsets/{recordset_id}

Scope Types

- **project**

Delete RecordSet

`count_recordset`

Default
`(role:admin) or (role:reader and project_id:%(project_id)s)`

Scope Types

- **project**

Count recordsets

`find_service_status`

Default
`role:admin`

Operations

- **GET** `/v2/service_status/{service_id}`

Scope Types

- **project**

Find a single Service Status

`find_service_statuses`

Default
`role:admin`

Operations

- **GET** `/v2/service_status`

Scope Types

- **project**

List service statuses.

`update_service_status`

Default
`role:admin`

Scope Types

- **project**

(no description provided)

`get_zone_share`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **GET** `/v2/zones/{zone_id}/shares/{zone_share_id}`

Scope Types

- **project**

Get a Zone Share

`share_zone`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **POST** `/v2/zones/{zone_id}/shares`

Scope Types

- **project**

Share a Zone

`find_zone_shares`

Default
`@`

Operations

- **GET** `/v2/zones/{zone_id}/shares`

List Shared Zones

`find_project_zone_share`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Scope Types

- **project**

Check the can query for a specific projects shares.

`unshare_zone`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **DELETE** `/v2/zones/{zone_id}/shares/{shared_zone_id}`

Scope Types

- **project**

Unshare Zone

`find_tenants`

Default
`role:admin`

Scope Types

- **project**

Find all Tenants.

`get_tenant`

Default
`role:admin`

Scope Types

- **project**

Get all Tenants.

`count_tenants`

Default
`role:admin`

Scope Types

- **project**

Count tenants

`create_tld`

Default
`role:admin`

Operations

- **POST** `/v2/tlds`

Scope Types

- **project**

Create Tld

`find_tlds`

Default
`role:admin`

Operations

- **GET** `/v2/tlds`

Scope Types

- **project**

List Tlds

`get_tld`

Default
`role:admin`

Operations

- **GET** `/v2/tlds/{tld_id}`

Scope Types

- **project**

Show Tld

`update_tld`

Default
`role:admin`

Operations

- **PATCH** /v2/tlds/{tld_id}

Scope Types

- **project**

Update Tld

`delete_tld`

Default
role:admin

Operations

- **DELETE** /v2/tlds/{tld_id}

Scope Types

- **project**

Delete Tld

`create_tsigkey`

Default
role:admin

Operations

- **POST** /v2/tsigkeys

Scope Types

- **project**

Create Tsigkey

`find_tsigkeys`

Default
role:admin

Operations

- **GET** /v2/tsigkeys

Scope Types

- **project**

List Tsigkeys

`get_tsigkey`

Default
role:admin

Operations

- **GET** /v2/tsigkeys/{tsigkey_id}

Scope Types

- **project**

Show a Tsigkey

`update_tsigkey`

Default
`role:admin`

Operations

- **PATCH** `/v2/tsigkeys/{tsigkey_id}`

Scope Types

- **project**

Update Tsigkey

`delete_tsigkey`

Default
`role:admin`

Operations

- **DELETE** `/v2/tsigkeys/{tsigkey_id}`

Scope Types

- **project**

Delete a Tsigkey

`create_zone`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **POST** `/v2/zones`

Scope Types

- **project**

Create Zone

`get_zones`

Default
`(role:admin) or (role:reader and project_id:%(project_id)s)`

Scope Types

- **project**

(no description provided)

`get_zone`

Default
`(role:admin) or (role:reader and project_id:%(project_id)s) or ('True':%(zone_shared)s)`

Operations

- **GET** /v2/zones/{zone_id}

Scope Types

- **project**

Get Zone

`get_zone_servers`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Scope Types

- **project**

(no description provided)

`get_zone_ns_records`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/zones/{zone_id}/nameservers

Scope Types

- **project**

Get the Name Servers for a Zone

`find_zones`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/zones

Scope Types

- **project**

List existing zones

`update_zone`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **PATCH** /v2/zones/{zone_id}

Scope Types

- **project**

Update Zone

`delete_zone`

Default

`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **DELETE** `/v2/zones/{zone_id}`

Scope Types

- **project**

Delete Zone

`xfr_zone`

Default

`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/xfr`

Scope Types

- **project**

Manually Trigger an Update of a Secondary Zone

`abandon_zone`

Default

`role:admin`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/abandon`

Scope Types

- **project**

Abandon Zone

`count_zones`

Default

`(role:admin) or (role:reader and project_id:%(project_id)s)`

Scope Types

- **project**

(no description provided)

`count_zones_pending_notify`

Default

`(role:admin) or (role:reader and project_id:%(project_id)s)`

Scope Types

- **project**

(no description provided)

`purge_zones`

Default
`role:admin`

Scope Types

- **project**

(no description provided)

`pool_move_zone`

Default
`role:admin`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/pool_move`

Scope Types

- **project**

Pool Move Zone

`zone_export`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **GET** `/v2/zones/tasks/exports/{zone_export_id}/export`

Scope Types

- **project**

Retrive a Zone Export from the Designate Datastore

`create_zone_export`

Default
`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/export`

Scope Types

- **project**

Create Zone Export

`find_zone_exports`

Default
`(role:admin) or (role:reader and project_id:%(project_id)s)`

Operations

- **GET** `/v2/zones/tasks/exports`

Scope Types

- **project**

List Zone Exports

`get_zone_export`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** `/v2/zones/tasks/exports/{zone_export_id}`

Scope Types

- **project**

Get Zone Exports

`update_zone_export`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **POST** `/v2/zones/{zone_id}/tasks/export`

Scope Types

- **project**

Update Zone Exports

`delete_zone_export`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **DELETE** `/v2/zones/tasks/exports/{zone_export_id}`

Scope Types

- **project**

Delete a zone export

`create_zone_import`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **POST** `/v2/zones/tasks/imports`

Scope Types

- **project**

Create Zone Import

`find_zone_imports`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/zones/tasks/imports

Scope Types

- **project**

List all Zone Imports

`get_zone_import`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** /v2/zones/tasks/imports/{zone_import_id}

Scope Types

- **project**

Get Zone Imports

`update_zone_import`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **POST** /v2/zones/tasks/imports

Scope Types

- **project**

Update Zone Imports

`delete_zone_import`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **DELETE** /v2/zones/tasks/imports/{zone_import_id}

Scope Types

- **project**

Delete a Zone Import

`create_zone_transfer_accept`

Default

```
((role:admin) or (role:member and project_id:%(project_id)s))
or project_id:%(target_project_id)s or
None:%(target_project_id)s
```

Operations

- **POST** /v2/zones/tasks/transfer_accepts

Scope Types

- **project**

Create Zone Transfer Accept

`get_zone_transfer_accept`

Default

```
(role:admin) or (role:reader and project_id:%(project_id)s)
```

Operations

- **GET** `/v2/zones/tasks/transfer_requests/{zone_transfer_accept_id}`

Scope Types

- **project**

Get Zone Transfer Accept

`find_zone_transfer_accepts`

Default

```
role:admin
```

Operations

- **GET** `/v2/zones/tasks/transfer_accepts`

Scope Types

- **project**

List Zone Transfer Accepts

`create_zone_transfer_request`

Default

```
(role:admin) or (role:member and project_id:%(project_id)s)
```

Operations

- **POST** `/v2/zones/{zone_id}/tasks/transfer_requests`

Scope Types

- **project**

Create Zone Transfer Accept

`get_zone_transfer_request`

Default

```
((role:admin) or (role:member and project_id:%(project_id)s))
or project_id:%(target_project_id)s or
None:%(target_project_id)s
```

Operations

- **GET** `/v2/zones/tasks/transfer_requests/{zone_transfer_request_id}`

Scope Types

- **project**

Show a Zone Transfer Request

`get_zone_transfer_request_detailed`

Default

`(role:admin) or (role:reader and project_id:%(project_id)s)`

Scope Types

- **project**

(no description provided)

`find_zone_transfer_requests`

Default

`@`

Operations

- **GET** `/v2/zones/tasks/transfer_requests`

List Zone Transfer Requests

`update_zone_transfer_request`

Default

`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **PATCH** `/v2/zones/tasks/transfer_requests/{zone_transfer_request_id}`

Scope Types

- **project**

Update a Zone Transfer Request

`delete_zone_transfer_request`

Default

`(role:admin) or (role:member and project_id:%(project_id)s)`

Operations

- **DELETE** `/v2/zones/tasks/transfer_requests/{zone_transfer_request_id}`

Scope Types

- **project**

Delete a Zone Transfer Request

1.5.10 Config Documentation

The following is an overview of all available configuration in Designate. For a sample configuration file, refer to *designate.conf* .

DEFAULT

host

Type
string

Default
`current_hostname`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Name of this node

pybasedir

Type
string

Default
`<Path>`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Directory where the designate python module is installed

state_path

Type
string

Default
`/var/lib/designate`

Top-level directory for maintaining designates state

allowed_remote_exmods

Type
list

Default
`[]`

Additional modules that contains allowed RPC exceptions.

Table 1: Deprecated Variations

Group	Name
DEFAULT	allowed_rpc_exception_modules

default_ttl

Type
integer

Default
`3600`

TTL Value

default_soa_refresh_min

Type
integer

Default
3500

SOA refresh-min value

Table 2: Deprecated Variations

Group	Name
DEFAULT	default_soa_refresh

default_soa_refresh_max

Type
integer

Default
3600

SOA max value

default_soa_retry

Type
integer

Default
600

SOA retry

default_soa_expire

Type
integer

Default
86400

SOA expire

default_soa_minimum

Type
integer

Default
3600

SOA minimum value

`supported_record_type`

Type
list

Default
`['A', 'AAAA', 'CNAME', 'MX', 'SRV', 'TXT', 'SPF', 'NS', 'PTR', 'SSHFP', 'SOA', 'NAPTR', 'CAA', 'CERT']`

Supported record types

`backlog`

Type
integer

Default
`4096`

Number of backlog requests to configure the socket with

`root_helper`

Type
string

Default
`sudo designate-rootwrap /etc/designate/rootwrap.conf`

designate-rootwrap configuration

`network_api`

Type
string

Default
`neutron`

Which API to use.

`notify_api_faults`

Type
boolean

Default
`False`

Send notifications if theres a failure in the API.

`notification_plugin`

Type
string

Default
`default`

The notification plugin to use

`quota_driver`

Type
string

Default
`storage`

Quota driver to use

`quota_zones`

Type
integer

Default
`10`

Number of zones allowed per tenant

`quota_zone_recordsets`

Type
integer

Default
`500`

Number of recordsets allowed per zone

`quota_zone_records`

Type
integer

Default
`500`

Number of records allowed per zone

`quota_recordset_records`

Type
integer

Default
`20`

Number of records allowed per recordset

`quota_api_export_size`

Type
integer

Default
`1000`

Number of recordsets allowed in a zone export

`run_external_periodic_tasks`

Type
boolean

Default
`True`

Some periodic tasks can be run in a separate process. Should we run them here?

`backdoor_port`

Type
string

Default
<None>

Enable eventlet backdoor. Acceptable values are 0, <port>, and <start>:<end>, where 0 results in listening on a random tcp port number; <port> results in listening on the specified port number (and not enabling backdoor if that port is in use); and <start>:<end> results in listening on the smallest unused port number within the specified range of port numbers. The chosen port is displayed in the services log file.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The backdoor_port option is deprecated and will be removed in a future release.

`backdoor_socket`

Type
string

Default
<None>

Enable eventlet backdoor, using the provided path as a unix socket that can receive connections. This option is mutually exclusive with backdoor_port in that only one should be provided. If both are provided then the existence of this option overrides the usage of that option. Inside the path {pid} will be replaced with the PID of the current process.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The backdoor_socket option is deprecated and will be removed in a future release.

`log_options`

Type
boolean

Default
`True`

Enables or disables logging values of all registered options when starting a service (at DEBUG level).

`graceful_shutdown_timeout`

Type
integer

Default
`60`

Specify a timeout after which a gracefully shutdown server will exit. Zero value means endless wait.

`api_paste_config`

Type
string

Default
`api-paste.ini`

File name for the paste.deploy config for api service

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The api_paste_config option is deprecated and will be removed in a future release.

`wsgi_log_format`

Type
string

Default
`%(client_ip)s "%(request_line)s" status: %(status_code)s len: %(body_length)s time: %(wall_seconds).7f`

A python format string that is used as the template to generate log lines. The following values can beformatted into it: client_ip, date_time, request_line, status_code, body_length, wall_seconds.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The wsgi_log_format option is deprecated and will be removed in a future release.

`tcp_keepidle`

Type
integer

Default
`600`

Sets the value of TCP_KEEPIDLE in seconds for each server socket. Not supported on OS X.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The tcp_keepidle option is deprecated and will be removed in a future release.

`wsgi_default_pool_size`

Type
integer

Default
`100`

Size of the pool of greenthreads used by wsgi

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The wsgi_default_pool_size option is deprecated and will be removed in a future release.

`max_header_line`

Type
integer

Default
`16384`

Maximum line size of message headers to be accepted. max_header_line may need to be increased when using large tokens (typically those generated when keystone is configured to use PKI tokens with big service catalogs).

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The max_header_line option is deprecated and will be removed in a future release.

`wsgi_keep_alive`

Type
boolean

Default
`True`

If False, closes the client socket connection explicitly.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The wsgi_keep_alive option is deprecated and will be removed in a future release.

`client_socket_timeout`

Type
integer

Default
`900`

Timeout for client connections socket operations. If an incoming connection is idle for this number of seconds it will be closed. A value of 0 means wait forever.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The client_socket_timeout option is deprecated and will be removed in a future release.

`wsgi_server_debug`

Type
boolean

Default
`False`

True if the server should send exception tracebacks to the clients on 500 errors. If False, the server will respond with empty bodies.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The wsgi_server_debug option is deprecated and will be removed in a future release.

`executor_thread_pool_size`

Type
integer

Default
`64`

Size of executor thread pool when executor is threading or eventlet.

Table 3: Deprecated Variations

Group	Name
DEFAULT	rpc_thread_pool_size

`rpc_response_timeout`

Type
integer

Default
`60`

Seconds to wait for a response from a call.

`transport_url`

Type
string

Default
`rabbit://`

The network address and optional user credentials for connecting to the messaging backend, in URL format. The expected format is:

driver://[user:pass@]host:port[,[userN:passN@]hostN:portN]/virtual_host?query

Example: rabbit://rabbitmq:password@127.0.0.1:5672//

For full details on the fields in the URL see the documentation of oslo_messaging.TransportURL at https://docs.openstack.org/oslo.messaging/latest/reference/transport.html

`control_exchange`

Type
string

Default
`designate`

The default exchange under which topics are scoped. May be overridden by an exchange name specified in the transport_url option.

`rpc_ping_enabled`

Type
boolean

Default
`False`

Add an endpoint to answer to ping calls. Endpoint is named oslo_rpc_server_ping

`debug`

Type
boolean

Default
`False`

Mutable
This option can be changed without restarting.

If set to true, the logging level will be set to DEBUG instead of the default INFO level.

`log_config_append`

Type
string

Default
<None>

Mutable
This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 4: Deprecated Variations

Group	Name
DEFAULT	log-config
DEFAULT	log_config

`log_date_format`

Type
string

Default
`%Y-%m-%d %H:%M:%S`

Defines the format string for %(asctime)s in log records. Default: the value above . This option is ignored if log_config_append is set.

`log_file`

Type
string

Default
<None>

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

Table 5: Deprecated Variations

Group	Name
DEFAULT	logfile

`log_dir`

Type
string

Default
<None>

(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

Table 6: Deprecated Variations

Group	Name
DEFAULT	logdir

`watch_log_file`

Type
boolean

Default
`False`

Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if log_file option is specified and Linux platform is used. This option is ignored if log_config_append is set.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
This function is known to have bene broken for long time, and depends on the unmaintained library

`use_syslog`

Type
boolean

Default
`False`

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

`use_journal`

Type
boolean

Default
`False`

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages.This option is ignored if log_config_append is set.

`syslog_log_facility`

Type
string

Default
`LOG_USER`

Syslog facility to receive log lines. This option is ignored if log_config_append is set.

`use_json`

Type
boolean

Default
`False`

Use JSON formatting for logging. This option is ignored if log_config_append is set.

`use_stderr`

Type
boolean

Default
`False`

Log output to standard error. This option is ignored if log_config_append is set.

`log_color`

Type
boolean

Default
`False`

(Optional) Set the color key according to log levels. This option takes effect only when logging to stderr or stdout is used. This option is ignored if log_config_append is set.

`log_rotate_interval`

Type
integer

Default
1

The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is set to interval.

`log_rotate_interval_type`

Type
string

Default
`days`

Valid Values
Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

`max_logfile_count`

Type
integer

Default
`30`

Maximum number of rotated log files.

`max_logfile_size_mb`

Type
integer

Default
`200`

Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

`log_rotation_type`

Type
string

Default
`none`

Valid Values
interval, size, none

Log rotation type.

Possible values

interval
Rotate logs at predefined time intervals.

size
Rotate logs once they reach a predefined size.

none
Do not rotate log files.

`logging_context_format_string`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s
[%(global_request_id)s %(request_id)s %(user_identity)s]
%(instance)s%(message)s
```

Format string to use for log messages with context. Used by oslo_log.formatters.ContextFormatter

`logging_default_format_string`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-]
%(instance)s%(message)s
```

Format string to use for log messages when context is undefined. Used by oslo_log.formatters.ContextFormatter

`logging_debug_format_suffix`

Type
string

Default
```
%(funcName)s %(pathname)s:%(lineno)d
```

Additional data to append to log message when logging level for the message is DEBUG. Used by oslo_log.formatters.ContextFormatter

`logging_exception_prefix`

Type
string

Default
```
%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s
%(instance)s
```

Prefix each line of exception output with this format. Used by oslo_log.formatters.ContextFormatter

`logging_user_identity_format`

Type
string

Default
```
%(user)s %(project)s %(domain)s %(system_scope)s
%(user_domain)s %(project_domain)s
```

Defines the format string for %(user_identity)s that is used in logging_context_format_string. Used by oslo_log.formatters.ContextFormatter

`default_log_levels`

Type
list

Default
```
['amqp=WARN', 'amqplib=WARN', 'boto=WARN', 'qpid=WARN',
'sqlalchemy=WARN', 'suds=INFO', 'oslo.messaging=INFO',
'oslo_messaging=INFO', 'iso8601=WARN', 'requests.packages.
urllib3.connectionpool=WARN', 'urllib3.connectionpool=WARN',
'websocket=WARN', 'requests.packages.urllib3.util.retry=WARN',
'urllib3.util.retry=WARN', 'keystonemiddleware=WARN',
'routes.middleware=WARN', 'stevedore=WARN', 'taskflow=WARN',
'keystoneauth=WARN', 'oslo.cache=INFO', 'oslo_policy=INFO',
'dogpile.core.dogpile=INFO', 'kazoo.client=WARN',
'keystone=INFO', 'oslo_service.loopingcall=WARN']
```

List of package logging levels in logger=LEVEL pairs. This option is ignored if log_config_append is set.

`publish_errors`

Type
boolean

Default
`False`

Enables or disables publication of error events.

`instance_format`

Type
string

Default
`"[instance: %(uuid)s] "`

The format for an instance that is passed with the log message.

`instance_uuid_format`

Type
string

Default
`"[instance: %(uuid)s] "`

The format for an instance UUID that is passed with the log message.

`rate_limit_interval`

Type
integer

Default
`0`

Interval, number of seconds, of log rate limiting.

`rate_limit_burst`

Type
integer

Default
`0`

Maximum number of logged messages per rate_limit_interval.

`rate_limit_except_level`

Type
string

Default
`CRITICAL`

Valid Values
CRITICAL, ERROR, INFO, WARNING, DEBUG,

Log level name used by rate limiting. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

`fatal_deprecations`

Type
boolean

Default
`False`

Enables or disables fatal status of deprecations.

backend:dynect

`job_timeout`

Type
integer

Default
`30`

Timeout in seconds for pulling a job in DynECT.

`timeout`

Type
integer

Default
`10`

Timeout in seconds for API Requests.

`timings`

Type
boolean

Default
`False`

Measure requests timings.

coordination

backend_url

Type
string

Default
<None>

The backend URL to use for distributed coordination. If unset services that need coordination will function as a standalone service. This is a *tooz* url - see https://docs.openstack.org/tooz/latest/user/compatibility.html

heartbeat_interval

Type
floating point

Default
`5.0`

Number of seconds between heartbeats for distributed coordination.

run_watchers_interval

Type
floating point

Default
`10.0`

Number of seconds between checks to see if group membership has changed

cors

allowed_origin

Type
list

Default
<None>

Indicate whether this resource may be shared with the domain received in the requests origin header. Format: <protocol>://<host>[:<port>], no trailing slash. Example: https://horizon.example.com

allow_credentials

Type
boolean

Default
`True`

Indicate that the actual request can include user credentials

`expose_headers`

Type
list

Default
`['X-OpenStack-Request-ID', 'Host']`

Indicate which headers are safe to expose to the API. Defaults to HTTP Simple Headers.

`max_age`

Type
integer

Default
`3600`

Maximum cache age of CORS preflight requests.

`allow_methods`

Type
list

Default
`['GET', 'PUT', 'POST', 'DELETE', 'PATCH', 'HEAD']`

Indicate which methods can be used during the actual request.

`allow_headers`

Type
list

Default
```
['X-Auth-Token', 'X-Auth-Sudo-Tenant-ID',
'X-Auth-Sudo-Project-ID', 'X-Auth-All-Projects',
'X-Designate-Edit-Managed-Records', 'X-Designate-Hard-Delete',
'OpenStack-DNS-Hide-Counts']
```

Indicate which header field names may be used during the actual request.

database

`sqlite_synchronous`

Type
boolean

Default
`True`

If True, SQLite uses synchronous mode.

`backend`

Type
string

Default
`sqlalchemy`

The back end to use for the database.

`connection`

Type
string

Default
<None>

The SQLAlchemy connection string to use to connect to the database.

`slave_connection`

Type
string

Default
<None>

The SQLAlchemy connection string to use to connect to the slave database.

`asyncio_connection`

Type
string

Default
<None>

The SQLAlchemy asyncio connection string to use to connect to the database.

`asyncio_slave_connection`

Type
string

Default
<None>

The SQLAlchemy asyncio connection string to use to connect to the slave database.

`mysql_sql_mode`

Type
string

Default
`TRADITIONAL`

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: mysql_sql_mode=

`mysql_wsrep_sync_wait`

Type
integer

Default
<None>

For Galera only, configure wsrep_sync_wait causality checks on new connections. Default is None, meaning dont configure any setting.

`connection_recycle_time`

Type
integer

Default
3600

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

`max_pool_size`

Type
integer

Default
5

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

`max_retries`

Type
integer

Default
10

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

`retry_interval`

Type
integer

Default
10

Interval between retries of opening a SQL connection.

`max_overflow`

Type
integer

Default
50

If set, use this value for max_overflow with SQLAlchemy.

`connection_debug`

Type
integer

Default
`0`

Minimum Value
0

Maximum Value
100

Verbosity of SQL debugging information: 0=None, 100=Everything.

`connection_trace`

Type
boolean

Default
`False`

Add Python stack traces to SQL as comment strings.

`pool_timeout`

Type
integer

Default
<None>

If set, use this value for pool_timeout with SQLAlchemy.

`use_db_reconnect`

Type
boolean

Default
`False`

Enable the experimental use of database reconnect on connection lost.

`db_retry_interval`

Type
integer

Default
1

Seconds between retries of a database transaction.

`db_inc_retry_interval`

Type
boolean

Default
`True`

If True, increases the interval between retries of a database operation up to db_max_retry_interval.

`db_max_retry_interval`

Type
integer

Default
`10`

If db_inc_retry_interval is set, the maximum seconds between retries of a database operation.

`db_max_retries`

Type
integer

Default
`20`

Maximum retries in case of connection error or deadlock error before error is raised. Set to -1 to specify an infinite retry count.

`connection_parameters`

Type
string

Default
`''`

Optional URL parameters to append onto the connection URL at connect time; specify as param1=value1&param2=value2&

handler:neutron_floatingip

`notification_topics`

Type
list

Default
`['notifications']`

notification any events from neutron

`control_exchange`

Type
string

Default
`neutron`

control-exchange for neutron notification

`zone_id`

Type
string

Default
<None>

Zone ID with each notification

`formatv4`

Type
multi-valued

Default
''

IPv4 format

`format`

Type
multi-valued

Default
''

format which replaced by formatv4/formatv6

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Replaced by formatv4/formatv6

`formatv6`

Type
multi-valued

Default
''

IPv6 format

handler:nova_fixed

`notification_topics`

Type
list

Default
['notifications']

notification any events from nova

`control_exchange`

Type
string

Default
`nova`

control-exchange for nova notification

`zone_id`

Type
string

Default
<None>

Zone ID with each notification

`formatv4`

Type
multi-valued

Default
`''`

IPv4 format

`format`

Type
multi-valued

Default
`''`

format which replaced by formatv4/formatv6

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Replaced by formatv4/formatv6

`formatv6`

Type
multi-valued

Default
`''`

IPv6 format

healthcheck

path

Type
string

Default
`/healthcheck`

The path to respond to healtcheck requests on.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

detailed

Type
boolean

Default
`False`

Show more detailed information as part of the response. Security note: Enabling this option may expose sensitive details about the service being monitored. Be sure to verify that it will not violate your security policies.

backends

Type
list

Default
`[]`

Additional backends that can perform health checks and report that information back as part of a request.

allowed_source_ranges

Type
list

Default
`[]`

A list of network addresses to limit source ip allowed to access healthcheck information. Any request from ip outside of these network addresses are ignored.

ignore_proxied_requests

Type
boolean

Default
`False`

Ignore requests with proxy headers.

`disable_by_file_path`

Type
string

Default
<None>

Check the presence of a file to determine if an application is running on a port. Used by DisableByFileHealthcheck plugin.

`disable_by_file_paths`

Type
list

Default
[]

Check the presence of a file based on a port to determine if an application is running on a port. Expects a port:path list of strings. Used by DisableByFilesPortsHealthcheck plugin.

`enable_by_file_paths`

Type
list

Default
[]

Check the presence of files. Used by EnableByFilesHealthcheck plugin.

heartbeat_emitter

`heartbeat_interval`

Type
floating point

Default
10.0

Number of seconds between heartbeats for reporting state

`emitter_type`

Type
string

Default
rpc

Emitter to use

keystone

`service_type`

Type
string

Default
<None>

The default service_type for endpoint URL discovery.

`service_name`

Type
string

Default
<None>

The default service_name for endpoint URL discovery.

`valid_interfaces`

Type
list

Default
<None>

List of interfaces, in order of preference, for endpoint URL.

`region_name`

Type
string

Default
<None>

The default region_name for endpoint URL discovery.

`endpoint_override`

Type
string

Default
<None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the *version*, *min-version*, and/or *max-version* options.

`version`

Type
string

Default
<None>

Minimum Major API version within a given Major API version for endpoint URL discovery. Mutually exclusive with min_version and max_version

`min_version`

Type
string

Default
<None>

The minimum major version of a given API, intended to be used as the lower bound of a range with max_version. Mutually exclusive with version. If min_version is given with no max_version it is as if max version is latest.

`max_version`

Type
string

Default
<None>

The maximum major version of a given API, intended to be used as the upper bound of a range with min_version. Mutually exclusive with version.

`connect_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for connection errors.

`connect_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

`status_code_retries`

Type
integer

Default
<None>

The maximum number of retries that should be attempted for retriable HTTP status codes.

`status_code_retry_delay`

Type
floating point

Default
<None>

Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

`retriable_status_codes`

Type
list

Default
<None>

List of retriable HTTP status codes that should be retried. If not set default to [503]

`interface`

Type
string

Default
<None>

The default interface for endpoint URL discovery.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Using valid-interfaces is preferrable because it is capable of accepting a list of possible interfaces.

`cafile`

Type
string

Default
<None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

`certfile`

Type
string

Default
<None>

PEM encoded client certificate cert file

`keyfile`

Type
string

Default
<None>

PEM encoded client certificate key file

`insecure`

Type
boolean

Default
`False`

Verify HTTPS connections.

`timeout`

Type
integer

Default
<None>

Timeout value for http requests

`collect_timing`

Type
boolean

Default
`False`

Collect per-API call timing information.

`split_loggers`

Type
boolean

Default
`False`

Log requests to multiple loggers.

keystone_authtoken

`www_authenticate_uri`

Type
string

Default
<None>

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint.

Table 7: Deprecated Variations

Group	Name
keystone_authtoken	auth_uri

auth_uri

Type
string

Default
<None>

Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If youre using a versioned v2 endpoint here, then this should *not* be the same endpoint the service user utilizes for validating tokens, because normal end users may not be able to reach that endpoint. This option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

Warning

This option is deprecated for removal since Queens. Its value may be silently ignored in the future.

Reason
The auth_uri option is deprecated in favor of www_authenticate_uri and will be removed in the S release.

auth_version

Type
string

Default
<None>

API version of the Identity API endpoint.

interface

Type
string

Default
internal

Interface to use for the Identity API endpoint. Valid values are public, internal (default) or admin.

delay_auth_decision

Type
boolean

Default
`False`

Do not handle authorization requests within the middleware, but delegate the authorization decision to downstream WSGI components.

`http_connect_timeout`

Type
integer

Default
<None>

Request timeout value for communicating with Identity API server.

`http_request_max_retries`

Type
integer

Default
3

How many times are we trying to reconnect when communicating with Identity API Server.

`cache`

Type
string

Default
<None>

Request environment key where the Swift cache object is stored. When auth_token middleware is deployed with a Swift cache, use this option to have the middleware share a caching backend with swift. Otherwise, use the `memcached_servers` option instead.

`certfile`

Type
string

Default
<None>

Required if identity server requires client certificate

`keyfile`

Type
string

Default
<None>

Required if identity server requires client certificate

`cafile`

Type
string

Default
<None>

A PEM encoded Certificate Authority to use when verifying HTTPs connections. Defaults to system CAs.

`insecure`

Type
boolean

Default
`False`

Verify HTTPS connections.

`region_name`

Type
string

Default
<None>

The region in which the identity server can be found.

`memcached_servers`

Type
list

Default
<None>

Optionally specify a list of memcached server(s) to use for caching. If left undefined, tokens will instead be cached in-process.

Table 8: Deprecated Variations

Group	Name
keystone_authtoken	memcache_servers

`token_cache_time`

Type
integer

Default
`300`

In order to prevent excessive effort spent validating tokens, the middleware caches previously-seen tokens for a configurable duration (in seconds). Set to -1 to disable caching completely.

`memcache_security_strategy`

Type
string

Default
None

Valid Values
None, MAC, ENCRYPT

(Optional) If defined, indicate whether token data should be authenticated or authenticated and encrypted. If MAC, token data is authenticated (with HMAC) in the cache. If ENCRYPT, token data is encrypted and authenticated in the cache. If the value is not one of these options or empty, auth_token will raise an exception on initialization.

`memcache_secret_key`

Type
string

Default
<None>

(Optional, mandatory if memcache_security_strategy is defined) This string is used for key derivation.

`memcache_pool_dead_retry`

Type
integer

Default
300

(Optional) Number of seconds memcached server is considered dead before it is tried again.

`memcache_pool_maxsize`

Type
integer

Default
10

(Optional) Maximum total number of open connections to every memcached server.

`memcache_pool_socket_timeout`

Type
integer

Default
3

(Optional) Socket timeout in seconds for communicating with a memcached server.

`memcache_pool_unused_timeout`

Type
integer

Default
60

(Optional) Number of seconds a connection to memcached is held unused in the pool before it is closed.

memcache_pool_conn_get_timeout

Type
integer

Default
`10`

(Optional) Number of seconds that an operation will wait to get a memcached client connection from the pool.

memcache_use_advanced_pool

Type
boolean

Default
`True`

(Optional) Use the advanced (eventlet safe) memcached client pool.

include_service_catalog

Type
boolean

Default
`True`

(Optional) Indicate whether to set the X-Service-Catalog header. If False, middleware will not ask for service catalog on token validation and will not set the X-Service-Catalog header.

enforce_token_bind

Type
string

Default
`permissive`

Used to control the use and type of token binding. Can be set to: disabled to not check token binding. permissive (default) to validate binding information if the bind type is of a form known to the server and ignore it if not. strict like permissive but if the bind type is unknown the token will be rejected. required any form of token binding is needed to be allowed. Finally the name of a binding method that must be present in tokens.

service_token_roles

Type
list

Default
`['service']`

A choice of roles that must be present in a service token. Service tokens are allowed to request that an expired token can be used and so this check should tightly control that only actual services should be sending this token. Roles here are applied as an ANY check so any role in this list

must be present. For backwards compatibility reasons this currently only affects the allow_expired check.

service_token_roles_required

Type
boolean

Default
`False`

For backwards compatibility reasons we must let valid service tokens pass that dont pass the service_token_roles check as valid. Setting this true will become the default in a future release and should be enabled if possible.

service_type

Type
string

Default
<None>

The name or type of the service as it appears in the service catalog. This is used to validate tokens that have restricted access rules.

memcache_sasl_enabled

Type
boolean

Default
`False`

Enable the SASL(Simple Authentication and Security Layer) if the SASL_enable is true, else disable.

memcache_username

Type
string

Default
`''`

the user name for the SASL

memcache_password

Type
string

Default
`''`

the username password for SASL

auth_type

Type
unknown type

Default
<None>

Authentication type to load

Table 9: Deprecated Variations

Group	Name
keystone_authtoken	auth_plugin

`auth_section`

Type
unknown type

Default
<None>

Config Section from which to load plugin specific options

network_api:neutron

`endpoints`

Type
list

Default
<None>

URL to use if None in the ServiceCatalog that is passed by the request context. Format: <region>|<url>

`endpoint_type`

Type
string

Default
publicURL

Endpoint type to use

`timeout`

Type
integer

Default
30

timeout value for connecting to neutron in seconds

`insecure`

Type
boolean

Default
`False`

if set, ignore any SSL validation issues

`ca_certificates_file`

Type
string

Default
<None>

Location of ca certificates file to use for neutron client requests.

`client_certificate_file`

Type
string

Default
<None>

Location of client certificate file to use for neutron client requests.

oslo_concurrency

`disable_process_locking`

Type
boolean

Default
`False`

Enables or disables inter-process locks.

`lock_path`

Type
string

Default
`$state_path`

Directory to use for lock files. For security, the specified directory should only be writable by the user running the processes that need locking. Defaults to environment variable OSLO_LOCK_PATH. If external locks are used, a lock path must be set.

oslo_messaging_kafka

`kafka_max_fetch_bytes`

Type
integer

Default
`1048576`

Max fetch bytes of Kafka consumer

`kafka_consumer_timeout`

Type
floating point

Default
`1.0`

Default timeout(s) for Kafka consumers

`consumer_group`

Type
string

Default
`oslo_messaging_consumer`

Group id for Kafka consumer. Consumers in one group will coordinate message consumption

`producer_batch_timeout`

Type
floating point

Default
`0.0`

Upper bound on the delay for KafkaProducer batching in seconds

`producer_batch_size`

Type
integer

Default
`16384`

Size of batch for the producer async send

`compression_codec`

Type
string

Default
`none`

Valid Values
none, gzip, snappy, lz4, zstd

The compression codec for all data generated by the producer. If not set, compression will not be used. Note that the allowed values of this depend on the kafka version

`enable_auto_commit`

Type
boolean

Default
`False`

Enable asynchronous consumer commits

`max_poll_records`

Type
integer

Default
500

The maximum number of records returned in a poll call

`security_protocol`

Type
string

Default
PLAINTEXT

Valid Values
PLAINTEXT, SASL_PLAINTEXT, SSL, SASL_SSL

Protocol used to communicate with brokers

`sasl_mechanism`

Type
string

Default
PLAIN

Mechanism when security protocol is SASL

`ssl_cafile`

Type
string

Default
''

CA certificate PEM file used to verify the server certificate

`ssl_client_cert_file`

Type
string

Default
''

Client certificate PEM file used for authentication.

`ssl_client_key_file`

Type
string

Default
''

Client key PEM file used for authentication.

`ssl_client_key_password`

Type
string

Default
''

Client key password file used for authentication.

oslo_messaging_notifications

`driver`

Type
multi-valued

Default
''

The Drivers(s) to handle sending notifications. Possible values are messaging, messagingv2, routing, log, test, noop

`transport_url`

Type
string

Default
<None>

A URL representing the messaging driver to use for notifications. If not set, we fall back to the same configuration used for RPC.

`topics`

Type
list

Default
['notifications']

AMQP topic used for OpenStack notifications.

`retry`

Type
integer

Default
-1

The maximum number of attempts to re-send a notification message which failed to be delivered due to a recoverable error. 0 - No retry, -1 - indefinite

oslo_messaging_rabbit

amqp_durable_queues

Type
boolean

Default
`False`

Use durable queues in AMQP. If rabbit_quorum_queue is enabled, queues will be durable and this value will be ignored.

amqp_auto_delete

Type
boolean

Default
`False`

Auto-delete queues in AMQP.

rpc_conn_pool_size

Type
integer

Default
`30`

Minimum Value
1

Size of RPC connection pool.

conn_pool_min_size

Type
integer

Default
`2`

The pool size limit for connections expiration policy

conn_pool_ttl

Type
integer

Default
`1200`

The time-to-live in sec of idle connections in the pool

ssl

Type
boolean

Default
`False`

Connect over SSL.

`ssl_version`

Type
string

Default
`''`

SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1_1, and TLSv1_2 may be available on some distributions.

`ssl_key_file`

Type
string

Default
`''`

SSL key file (valid only if SSL enabled).

`ssl_cert_file`

Type
string

Default
`''`

SSL cert file (valid only if SSL enabled).

`ssl_ca_file`

Type
string

Default
`''`

SSL certification authority file (valid only if SSL enabled).

`ssl_enforce_fips_mode`

Type
boolean

Default
`False`

Global toggle for enforcing the OpenSSL FIPS mode. This feature requires Python support. This is available in Python 3.9 in all environments and may have been backported to older Python versions on select environments. If the Python executable used does not support OpenSSL FIPS mode, an exception will be raised.

`heartbeat_in_pthread`

Type
boolean

Default
`False`

(DEPRECATED) It is recommend not to use this option anymore. Run the health check heartbeat thread through a native python thread by default. If this option is equal to False then the health check heartbeat will inherit the execution model from the parent process. For example if the parent process has monkey patched the stdlib by using eventlet/greenlet then the heartbeat will be run through a green thread. This option should be set to True only for the wsgi services.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The option is related to Eventlet which will be removed. In addition this has never worked as expected with services using eventlet for core service framework.

`kombu_reconnect_delay`

Type
floating point

Default
`1.0`

Minimum Value
0.0

Maximum Value
4.5

How long to wait (in seconds) before reconnecting in response to an AMQP consumer cancel notification.

`kombu_reconnect_splay`

Type
floating point

Default
`0.0`

Minimum Value
0.0

Random time to wait for when reconnecting in response to an AMQP consumer cancel notification.

`kombu_compression`

Type
string

Default
<None>

EXPERIMENTAL: Possible values are: gzip, bz2. If not set compression will not be used. This option may not be available in future versions.

`kombu_missing_consumer_retry_timeout`

Type
integer

Default
60

How long to wait a missing client before abandoning to send it its replies. This value should not be longer than rpc_response_timeout.

Table 10: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_reconnect_timeout

`kombu_failover_strategy`

Type
string

Default
round-robin

Valid Values
round-robin, shuffle

Determines how the next RabbitMQ node is chosen in case the one we are currently connected to becomes unavailable. Takes effect only if more than one RabbitMQ node is provided in config.

`rabbit_login_method`

Type
string

Default
AMQPLAIN

Valid Values
PLAIN, AMQPLAIN, EXTERNAL, RABBIT-CR-DEMO

The RabbitMQ login method.

`rabbit_retry_interval`

Type
integer

Default
1

Minimum Value
1

How frequently to retry connecting with RabbitMQ.

`rabbit_retry_backoff`

Type
integer

Default
2

Minimum Value
0

How long to backoff for between retries when connecting to RabbitMQ.

`rabbit_interval_max`

Type
integer

Default
30

Minimum Value
1

Maximum interval of RabbitMQ connection retries.

`rabbit_ha_queues`

Type
boolean

Default
`False`

Try to use HA queues in RabbitMQ (x-ha-policy: all). If you change this option, you must wipe the RabbitMQ database. In RabbitMQ 3.0, queue mirroring is no longer controlled by the x-ha-policy argument when declaring a queue. If you just want to make sure that all queues (except those with auto-generated names) are mirrored across all nodes, run: rabbitmqctl set_policy HA ^(?!amq.).* {ha-mode: all}

`rabbit_quorum_queue`

Type
boolean

Default
`False`

Use quorum queues in RabbitMQ (x-queue-type: quorum). The quorum queue is a modern queue type for RabbitMQ implementing a durable, replicated FIFO queue based on the Raft consensus algorithm. It is available as of RabbitMQ 3.8.0. If set this option will conflict with the HA queues (`rabbit_ha_queues`) aka mirrored queues, in other words the HA queues should be disabled. Quorum queues are also durable by default so the amqp_durable_queues option is ignored when this option is enabled.

`rabbit_transient_quorum_queue`

Type
boolean

Default
`False`

Use quorum queues for transients queues in RabbitMQ. Enabling this option will then make sure those queues are also using quorum kind of rabbit queues, which are HA by default.

`rabbit_quorum_delivery_limit`

Type
integer

Default
`0`

Each time a message is redelivered to a consumer, a counter is incremented. Once the redelivery count exceeds the delivery limit the message gets dropped or dead-lettered (if a DLX exchange has been configured) Used only when rabbit_quorum_queue is enabled, Default 0 which means dont set a limit.

`rabbit_quorum_max_memory_length`

Type
integer

Default
`0`

By default all messages are maintained in memory if a quorum queue grows in length it can put memory pressure on a cluster. This option can limit the number of messages in the quorum queue. Used only when rabbit_quorum_queue is enabled, Default 0 which means dont set a limit.

Table 11: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_quroum_max_memory_length

`rabbit_quorum_max_memory_bytes`

Type
integer

Default
`0`

By default all messages are maintained in memory if a quorum queue grows in length it can put memory pressure on a cluster. This option can limit the number of memory bytes used by the quorum queue. Used only when rabbit_quorum_queue is enabled, Default 0 which means dont set a limit.

Table 12: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_quroum_max_memory_bytes

`rabbit_transient_queues_ttl`

Type
integer

Default
`1800`

Minimum Value
0

Positive integer representing duration in seconds for queue TTL (x-expires). Queues which are unused for the duration of the TTL are automatically deleted. The parameter affects only reply and fanout queues. Setting 0 as value will disable the x-expires. If doing so, make sure you have a rabbitmq policy to delete the queues or you deployment will create an infinite number of queue over time.In case rabbit_stream_fanout is set to True, this option will control data retention policy (x-max-age) for messages in the fanout queue rather then the queue duration itself. So the oldest data in the stream queue will be discarded from it once reaching TTL Setting to 0 will disable x-max-age for stream which make stream grow indefinitely filling up the diskspace

`rabbit_qos_prefetch_count`

Type
integer

Default
`0`

Specifies the number of messages to prefetch. Setting to zero allows unlimited messages.

`heartbeat_timeout_threshold`

Type
integer

Default
`60`

Number of seconds after which the Rabbit broker is considered down if heartbeats keep-alive fails (0 disables heartbeat).

`heartbeat_rate`

Type
integer

Default
`3`

How often times during the heartbeat_timeout_threshold we check the heartbeat.

`direct_mandatory_flag`

Type
boolean

Default
`True`

(DEPRECATED) Enable/Disable the RabbitMQ mandatory flag for direct send. The direct send is used as reply, so the MessageUndeliverable exception is raised in case the client queue does not exist.MessageUndeliverable exception will be used to loop for a timeout to lets a chance to sender to recover.This flag is deprecated and it will not be possible to deactivate this functionality anymore

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Mandatory flag no longer deactivable.

`enable_cancel_on_failover`

Type
boolean

Default
`False`

Enable x-cancel-on-ha-failover flag so that rabbitmq server will cancel and notify consumerswhen queue is down

`use_queue_manager`

Type
boolean

Default
`False`

Should we use consistant queue names or random ones

`hostname`

Type
string

Default
`node1.example.com`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Hostname used by queue manager. Defaults to the value returned by socket.gethostname().

`processname`

Type
string

Default
`nova-api`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Process name used by queue manager

rabbit_stream_fanout

Type
boolean

Default
`False`

Use stream queues in RabbitMQ (x-queue-type: stream). Streams are a new persistent and replicated data structure (queue type) in RabbitMQ which models an append-only log with non-destructive consumer semantics. It is available as of RabbitMQ 3.9.0. If set this option will replace all fanout queues with only one stream queue.

oslo_middleware

max_request_body_size

Type
integer

Default
`114688`

The maximum body size for each request, in bytes.

Table 13: Deprecated Variations

Group	Name
DEFAULT	osapi_max_request_body_size
DEFAULT	max_request_body_size

enable_proxy_headers_parsing

Type
boolean

Default
`False`

Whether the application is behind a proxy or not. This determines if the middleware should parse the headers or not.

http_basic_auth_user_file

Type
string

Default
`/etc/htpasswd`

HTTP basic auth password file.

oslo_policy

enforce_scope

Type
boolean

Default
`True`

This option controls whether or not to enforce scope when evaluating policies. If `True`, the scope of the token used in the request is compared to the `scope_types` of the policy being enforced. If the scopes do not match, an `InvalidScope` exception will be raised. If `False`, a message will be logged informing operators that policies are being invoked with mismatching scope.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
This configuration was added temporarily to facilitate a smooth transition to the new RBAC. OpenStack will always enforce scope checks. This configuration option is deprecated and will be removed in the 2025.2 cycle.

enforce_new_defaults

Type
boolean

Default
`True`

This option controls whether or not to use old deprecated defaults when evaluating policies. If `True`, the old deprecated defaults are not going to be evaluated. This means if any existing token is allowed for old defaults but is disallowed for new defaults, it will be disallowed. It is encouraged to enable this flag along with the `enforce_scope` flag so that you can get the benefits of new defaults and `scope_type` together. If `False`, the deprecated policy check string is logically ORd with the new policy check string, allowing for a graceful upgrade experience between releases with new policies, which is the default behavior.

policy_file

Type
string

Default
`policy.yaml`

The relative or absolute path of a file that maps roles to permissions for a given service. Relative paths must be specified in relation to the configuration file setting this option.

policy_default_rule

Type
string

Default
`default`

Default rule. Enforced when a requested rule is not found.

`policy_dirs`

Type
multi-valued

Default
`policy.d`

Directories where policy configuration files are stored. They can be relative to any directory in the search path defined by the config_dir option, or absolute paths. The file defined by policy_file must exist for these directories to be searched. Missing or empty directories are ignored.

`remote_content_type`

Type
string

Default
`application/x-www-form-urlencoded`

Valid Values
application/x-www-form-urlencoded, application/json

Content Type to send and receive data for REST based policy check

`remote_ssl_verify_server_crt`

Type
boolean

Default
`False`

server identity verification for REST based policy check

`remote_ssl_ca_crt_file`

Type
string

Default
<None>

Absolute path to ca cert file for REST based policy check

`remote_ssl_client_crt_file`

Type
string

Default
<None>

Absolute path to client cert for REST based policy check

`remote_ssl_client_key_file`

Type
string

Default
<None>

Absolute path client key file REST based policy check

`remote_timeout`

Type
floating point

Default
60

Minimum Value
0

Timeout in seconds for REST based policy check

oslo_reports

`log_dir`

Type
string

Default
<None>

Path to a log directory where to create a file

`file_event_handler`

Type
string

Default
<None>

The path to a file to watch for changes to trigger the reports, instead of signals. Setting this option disables the signal trigger for the reports. If application is running as a WSGI application it is recommended to use this instead of signals.

`file_event_handler_interval`

Type
integer

Default
1

How many seconds to wait between polls when file_event_handler is set

oslo_versionedobjects

fatal_exception_format_errors

Type
boolean

Default
`False`

Make exception message format errors fatal

producer_task:delayed_notify

interval

Type
integer

Default
`5`

Run interval in seconds

per_page

Type
integer

Default
`100`

Default amount of results returned per page

batch_size

Type
integer

Default
`100`

How many zones to receive NOTIFY on each run

producer_task:periodic_exists

interval

Type
integer

Default
`3600`

Run interval in seconds

per_page

Type
integer

Default
100

Default amount of results returned per page

producer_task:periodic_secondary_refresh

interval

Type
integer

Default
3600

Run interval in seconds

per_page

Type
integer

Default
100

Default amount of results returned per page

producer_task:worker_periodic_recovery

interval

Type
integer

Default
120

Run interval in seconds

per_page

Type
integer

Default
100

Default amount of results returned per page

producer_task:zone_purge

interval

Type
integer

Default
3600

Run interval in seconds

`per_page`

Type
integer

Default
`100`

Default amount of results returned per page

`time_threshold`

Type
integer

Default
`604800`

How old deleted zones should be (deleted_at) to be purged, in seconds

`batch_size`

Type
integer

Default
`100`

How many zones to be purged on each run

proxy

`http_proxy`

Type
string

Default
<None>

Proxy HTTP requests via this proxy.

`https_proxy`

Type
string

Default
<None>

Proxy HTTPS requests via this proxy

`no_proxy`

Type
list

Default
`[]`

These addresses should not be proxied

service:api

workers

Type
integer

Default
<None>

Number of api worker processes to spawn

threads

Type
integer

Default
1000

Number of api greenthreads to spawn

enable_host_header

Type
boolean

Default
True

Enable host request headers

api_base_uri

Type
string

Default
http://127.0.0.1:9001/

the url used as the base for all API responses,This should consist of the scheme (http/https),the hostname, port, and any paths that are addedto the base of Designate is URLs,For example http://dns.openstack.example.com/dns

listen

Type
list

Default
['0.0.0.0:9001']

API host:port pairs to listen on

api_paste_config

Type
string

Default
api-paste.ini

File name for the paste.deploy config for designate-api

`auth_strategy`

Type
string

Default
`keystone`

The strategy to use for auth. Supports noauth or keystone

`enable_api_v2`

Type
boolean

Default
`True`

Enable the Designate V2 API

`enable_api_admin`

Type
boolean

Default
`False`

enable-api-admin

`pecan_debug`

Type
boolean

Default
`False`

Pecan HTML Debug Interface

`enabled_extensions_v2`

Type
list

Default
`[]`

Enabled API Extensions for the V2 API

`default_limit_v2`

Type
integer

Default
`20`

Default per-page limit for the V2 API, a value of None means show all results by default

`max_limit_v2`

Type
integer

Default
`1000`

Max per-page limit for the V2 API

`quotas_verify_project_id`

Type
boolean

Default
`False`

Verify that the requested Project ID for quota target is a valid project in Keystone.

`allow_empty_secrets_for_tsig`

Type
boolean

Default
`True`

Allow tsig creation with empty secrets. While in theory an empty string is valid for tsig secrets, it is highly not recommended

`enabled_extensions_admin`

Type
list

Default
`[]`

Enabled Admin API Extensions

`default_limit_admin`

Type
integer

Default
`20`

Default per-page limit for the Admin API, a value of None means show all results by default

`max_limit_admin`

Type
integer

Default
`1000`

Max per-page limit for the Admin API

`maintenance_mode`

Type
boolean

Default
`False`

Enable API Maintenance Mode

`maintenance_mode_role`

Type
string

Default
`admin`

Role allowed to bypass maintaince mode

service:central

`workers`

Type
integer

Default
<None>

Number of central worker processes to spawn

`threads`

Type
integer

Default
`1000`

Number of central greenthreads to spawn

`max_zone_name_len`

Type
integer

Default
`255`

Maximum zone name length

`max_recordset_name_len`

Type
integer

Default
`255`

Maximum recordset name length

Table 14: Deprecated Variations

Group	Name
service:central	max_record_name_len

`managed_resource_email`

Type
string

Default
`hostmaster@example.com`

E-Mail for Managed resources

`managed_resource_tenant_id`

Type
string

Default
`00000000-0000-0000-0000-000000000000`

The Tenant ID that will own any managed resources.

`min_ttl`

Type
integer

Default
`0`

Minimum TTL allowed

`default_pool_id`

Type
string

Default
`794ccc2c-d751-44fe-b57f-8894c9f5c842`

The name of the default pool

`topic`

Type
string

Default
`central`

RPC topic name for central

`scheduler_filters`

Type
list

Default
`['default_pool']`

Enabled Pool Scheduling filters

service:mdns

`workers`

Type
integer

Default
<None>

Number of mDNS worker processes to spawn

`threads`

Type
integer

Default
`1000`

Number of mDNS greenthreads to spawn

`listen`

Type
list

Default
`['0.0.0.0:5354']`

mDNS host:port pairs to listen on

`tcp_backlog`

Type
integer

Default
`100`

mDNS TCP Backlog

`tcp_keepidle`

Type
integer

Default
<None>

mDNS TCP Keepidle in seconds

`tcp_recv_timeout`

Type
floating point

Default
`0.5`

mDNS TCP Receive Timeout in seconds

`query_enforce_tsig`

Type
boolean

Default
`False`

Enforce all incoming queries (including AXFR) are TSIG signed

`max_message_size`

Type
integer

Default
`65535`

Maximum message size to emit

service:producer

`workers`

Type
integer

Default
<None>

Number of Producer worker processes to spawn

`threads`

Type
integer

Default
`1000`

Number of Producer greenthreads to spawn

`enabled_tasks`

Type
list

Default
<None>

Enabled tasks to run

export_synchronous

Type
boolean

Default
`True`

Whether to allow synchronous zone exports

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
Migrated to designate-worker

topic

Type
string

Default
`producer`

RPC topic name for producer

service:sink

workers

Type
integer

Default
<None>

Number of sink worker processes to spawn

threads

Type
integer

Default
`1000`

Number of sink greenthreads to spawn

enabled_notification_handlers

Type
list

Default
`[]`

Enabled Notification Handlers

`listener_pool_name`

Type
string

Default
<None>

pool name to use for oslo.messaging notification listener. Note that listener pooling is not supported by all oslo.messaging drivers.

service:worker

`workers`

Type
integer

Default
<None>

Number of Worker worker processes to spawn

`threads`

Type
integer

Default
200

Number of Worker threads to spawn per process

`threshold_percentage`

Type
integer

Default
100

The percentage of servers requiring a successful update for a domain change to be considered active

`poll_timeout`

Type
integer

Default
30

The time to wait for a response from a server

`poll_retry_interval`

Type
integer

Default
15

The time between retrying to send a request and waiting for a response from a server

`poll_max_retries`

Type
integer

Default
`10`

The maximum number of times to retry sending a request and wait for a response from a server

`poll_delay`

Type
integer

Default
`5`

The time to wait before sending the first request to a server

`export_synchronous`

Type
boolean

Default
`True`

Whether to allow synchronous zone exports

`topic`

Type
string

Default
`worker`

RPC topic name for worker

`xfr_timeout`

Type
integer

Default
`10`

Timeout in seconds for XFRs.

`serial_max_retries`

Type
integer

Default
`3`

The maximum number of times to retry fetching a zones serial.

`serial_retry_delay`

Type
integer

Default
1

The time to wait before retrying a zone serial request.

`serial_timeout`

Type
integer

Default
1

Timeout in seconds before giving up on fetching a zones serial.

`all_tcp`

Type
boolean

Default
`False`

Send all traffic over TCP

ssl

`ca_file`

Type
string

Default
<None>

CA certificate file to use to verify connecting clients.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The ca_file option is deprecated and will be removed in a future release.

`cert_file`

Type
string

Default
<None>

Certificate file to use when starting the server securely.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The cert_file option is deprecated and will be removed in a future release.

`key_file`

Type
string

Default
<None>

Private key file to use when starting the server securely.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The key_file option is deprecated and will be removed in a future release.

`version`

Type
string

Default
<None>

SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1_1, and TLSv1_2 may be available on some distributions.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The version option is deprecated and will be removed in a future release.

`ciphers`

Type
string

Default
<None>

Sets the list of available ciphers. value should be a string in the OpenSSL cipher list format.

Warning

This option is deprecated for removal. Its value may be silently ignored in the future.

Reason
The ciphers option is deprecated and will be removed in a future release.

storage:sqlalchemy

sqlite_synchronous

Type
boolean

Default
`True`

If True, SQLite uses synchronous mode.

backend

Type
string

Default
`sqlalchemy`

The back end to use for the database.

connection

Type
string

Default
<None>

The SQLAlchemy connection string to use to connect to the database.

slave_connection

Type
string

Default
<None>

The SQLAlchemy connection string to use to connect to the slave database.

asyncio_connection

Type
string

Default
<None>

The SQLAlchemy asyncio connection string to use to connect to the database.

`asyncio_slave_connection`

Type
string

Default
<None>

The SQLAlchemy asyncio connection string to use to connect to the slave database.

`mysql_sql_mode`

Type
string

Default
TRADITIONAL

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: mysql_sql_mode=

`mysql_wsrep_sync_wait`

Type
integer

Default
<None>

For Galera only, configure wsrep_sync_wait causality checks on new connections. Default is None, meaning dont configure any setting.

`connection_recycle_time`

Type
integer

Default
3600

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

`max_pool_size`

Type
integer

Default
5

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

`max_retries`

Type
integer

Default
10

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

`retry_interval`

Type
integer

Default
10

Interval between retries of opening a SQL connection.

`max_overflow`

Type
integer

Default
50

If set, use this value for max_overflow with SQLAlchemy.

`connection_debug`

Type
integer

Default
0

Minimum Value
0

Maximum Value
100

Verbosity of SQL debugging information: 0=None, 100=Everything.

`connection_trace`

Type
boolean

Default
False

Add Python stack traces to SQL as comment strings.

`pool_timeout`

Type
integer

Default
<None>

If set, use this value for pool_timeout with SQLAlchemy.

`use_db_reconnect`

Type
boolean

Default
`False`

Enable the experimental use of database reconnect on connection lost.

`db_retry_interval`

Type
integer

Default
`1`

Seconds between retries of a database transaction.

`db_inc_retry_interval`

Type
boolean

Default
`True`

If True, increases the interval between retries of a database operation up to db_max_retry_interval.

`db_max_retry_interval`

Type
integer

Default
`10`

If db_inc_retry_interval is set, the maximum seconds between retries of a database operation.

`db_max_retries`

Type
integer

Default
`20`

Maximum retries in case of connection error or deadlock error before error is raised. Set to -1 to specify an infinite retry count.

`connection_parameters`

Type
string

Default
`''`

Optional URL parameters to append onto the connection URL at connect time; specify as param1=value1&param2=value2&

1.5.11 Notifications

Hint

In this context, notifications are not related to the DNS NOTIFY message.

Notifications are RPC calls that contain a JSON object. Designate both generates and receives notifications.

The purpose of notifications in to inform unrelated OpenStack components of events in real time and trigger actions.

Emitters

They are emitted by Central on the following events:

- dns.tld.create
- dns.tld.update
- dns.tld.delete
- dns.tsigkey.create
- dns.tsigkey.update
- dns.tsigkey.delete
- dns.domain.create
- dns.zone.create
- dns.domain.update
- dns.zone.update
- dns.domain.delete
- dns.zone.delete
- dns.zone.touch
- dns.recordset.create
- dns.recordset.update
- dns.recordset.delete
- dns.record.create
- dns.record.update
- dns.record.delete
- dns.blacklist.create
- dns.blacklist.update
- dns.blacklist.delete
- dns.pool.create
- dns.pool.update

- dns.pool.delete
- dns.domain.update
- dns.zone.update
- dns.zone_transfer_request.create
- dns.zone_transfer_request.update
- dns.zone_transfer_request.delete
- dns.zone_transfer_accept.create
- dns.zone_transfer_accept.update
- dns.zone_transfer_accept.delete
- dns.zone_import.create
- dns.zone_import.update
- dns.zone_import.delete
- dns.zone_export.create
- dns.zone_export.update
- dns.zone_export.delete
- dns.zone.share
- dns.zone.unshare

Receivers

Notification from other OpenStack component outside of Designate are received by *Designate Sink*.

Format

An example notification from Neutron:

```
{
   "priority" : "INFO",
   "message_id" : "95ecdca3-967f-40aa-9469-d9fccc91d64b",
   "event_type" : "port.delete.start",
   "_context_roles" : [
      "Member"
   ],
   "_context_tenant_id" : "c97027dd880d4c129ae7a4ba7edade05",
   "timestamp" : "2012-11-16 12:56:17.155860",
   "_context_is_admin" : false,
   "_context_user_id" : "4ce5c085e09a478ea4edcd667a92df78",
   "payload" : {
      "port_id" : "bfdcb007-f68d-46bd-8150-abcae9fb3af6"
   },
   "_context_timestamp" : "2012-11-16 12:56:17.154672",
   "publisher_id" : "network.svc02.os.lan",
   "_context_read_deleted" : "no"
}
```

More examples can be found at `designate/tests/resources/sample_notifications`

1.5.12 Production Guidelines

This document aims to provide a location for documented production configurations and considerations. Including common misconfigurations, attack mitigation techniques, and other relevant tips.

DNS Zone Squatting

Designates multi-tenant nature allows for any user to create (almost) any zone, which can result in the legitimate owner being unable to create the zone within Designate. There are several ways this can occur:

1. The squatter simply creates example.com. in Designate before the legitimate owner can.
2. The squatter creates foo.example.com. as a zone in Designate, preventing the creation of any parent zones (example.com., com.) by any other tenant.
3. The squatter creates com. as a zone in Designate, preventing the creation of any zones ending in com. by any other tenant.
4. The squatter creates co.uk. as a zone in Designate, preventing the creation of any zones ending in co.uk. by any other tenant.

Scenario #1 and #2 Mitigation

There is no automated mitigation that can reasonably be performed here, DNS providers have typically used a manual process, triggered through a support request, to identify the legitimate owner and request the illegitimate owner relinquish control, or action any other provider specific policy for handling these scenarios.

Scenario #3 Mitigation

This scenario can be mitigated by ensuring Designate has been configured, and is updated periodically, with the latest list of gTLDs published as the IANA TLD list. These TLDs can be entered into Designate through the TLD API

Scenario #4 Mitigation

This is a variation on Scenario #3, where public registration is available for a second level domain, such as is the case with co.uk.. Due to the nature of public second level domains, where the IANA has no authority, these are not included in the IANA TLD list. A Mozilla sponsored initiative has stepped up to fill this gap, crowdsourcing the list of public suffixes, which includes both standard TLDs and public second level domains. We recommend configuring, and periodically updating, Designate with Mozillas Public Suffix list. These public suffixes can be entered into Designate through the TLD API

DNS Cache Poisoning

Multi-tenant nameservers can lead to an interesting variation of DNS Cache Poisoning if nameservers are configured without consideration. Two tenants, both owning different zones, can under the right circumstances inject content into DNS responses for the other tenants zone. Lets consider an example:

Tenant A owns example.com., and has created an additional NS record within their zone pointing to ns.example.org. Tenant B, the attacker in this example, can now create the example.org. zone within their tenant. Within this zone, they can legitimately create an A record with the name ns.example.org.. Under default configurations, many DNS servers (e.g. BIND), will now include Tenant Bs A record within

responses for several queries for example.com.. Should the recursive resolver used by the end-user not be configured to ignore out-of-bailiwick responses, this potentially invalid A record for ns.example.org. will be injected into the resolvers cache, resulting in a cache poisoning attack.

This is an interesting variation of DNS cache poisoning, because the poison records are returned by the authoritative nameserver for a given zone, rather than in responses for the attackers zone.

Bug 1471159 includes additional worked examples of this attack.

BIND9 Mitigation

BIND9 by default will include out-of-zone additionals, resulting is susceptibility to this attack. We recommend BIND is configured to send minimal responses - preventing the out-of-zone additionals from being processed.

In BINDs global options clause, include the following statement:

```
minimal-responses yes;
```

PowerDNS Mitigation

PowerDNS by default will include out-of-zone additionals, resulting is susceptibility to this attack. We recommend setting the *out-of-zone-additional-processing* configuration flag set to no - preventing the out-of-zone additionals from being processed.

In the main PowerDNS configuration file, include the following statement:

```
out-of-zone-additional-processing=no
```

1.5.13 Upgrades

In this section, you will find documentation relevant for upgrading Designate.

Note

The *designate-status upgrade check* command can be used to verify a deployment before starting services with new code.

Contents:

Upgrading to Kilo from Juno

Note

This doc section is a work in progress, for now, we have some smaller hints and tips for watchout for during the upgrade.

Tips and Tricks

1. Two new Designate services

Two new Designate services were added in Kilo, designate-pool-manager and designate-mdns. Please ensure to configure and enable these services as part of the upgrade.

2. Post-Migration, existing DNS domains hosted by PowerDNS must have their masters column manually populated with the list of designate-mdns ip and port pairs, and their type switched to SECONDARY. For example:

```
UPDATE powerdns.domains SET type = "SECONDARY", masters = "192.0.2.1:5354,192.
↪0.2.2:5354" WHERE masters IS NULL;
```

Upgrading to Mitaka from Liberty

Pools Configuration

We have updated how the config data for pools is now stored.

Previously there was a mix of content in the `designate.conf` file and in the designate database.

We have moved all of the data to the database in Mitaka, to avoid confusion, and avoid the massive complexity that exists in the config file.

Warning

This part of the upgrade **requires** downtime.

We have 2 new commands in the `designate-manage` utility that are able to assist the migration.

To make the config syntax simpler we have a new YAML based config file that is used to load information into the database.

```
---

- name: default
  # The name is immutable. There will be no option to change the name after
  # creation and the only way will to change it will be to delete it
  # (and all zones associated with it) and recreate it.
  description: Default PowerDNS Pool


  # Attributes are Key:Value pairs that describe the pool. for example the␣
↪level
  # of service (i.e. service_tier:GOLD), capabilities (i.e. anycast: true) or
  # other metadata. Users can use this information to point their zones to the
  # correct pool
  attributes: {}

  # List out the NS records for zones hosted within this pool
  ns_records:
    - hostname: ns1-1.example.org.
      priority: 1
    - hostname: ns1-2.example.org.
      priority: 2

  # List out the nameservers for this pool. These are the actual PowerDNS
  # servers. We use these to verify changes have propagated to all␣
```

(continues on next page)

(continued from previous page)

```
↪nameservers.
  nameservers:
    - host: 192.0.2.2
      port: 53

  # List out the targets for this pool. For PowerDNS, this is the database
  # (or databases, if you deploy a separate DB for each PowerDNS server)
  targets:
    - type: powerdns
      description: PowerDNS Database Cluster

      # List out the designate-mdns servers from which PowerDNS servers should
      # request zone transfers (AXFRs) from.
      masters:
        - host: 192.0.2.1
          port: 5354

      # PowerDNS Configuration options
      options:
        host: 192.0.2.2
        port: 53
        connection: 'mysql+pymysql://designate:password@127.0.0.1/designate_
↪pdns?charset=utf8'

  # Optional list of additional IP/Port's for which designate-mdns will send
  # DNS NOTIFY packets to
  also_notifies:
    - host: 192.0.2.4
      port: 53

  # Optional configuration to provide a catalog zone for the pool's zones.
  # If configured, catalog_zone_fqdn is required and all other keys are
  # optional.
  catalog_zone:
      catalog_zone_fqdn: cat.example.org.
      catalog_zone_refresh: 60
      # TSIG secret and algorithm to use for securing AXFRs for catalog zones.
      catalog_zone_tsig_key: SomeSecretKey
      catalog_zone_tsig_algorithm: hmac-sha512
```

We have a command that will allow you to take your current running config, and export it to the new YAML format.

Note

You will need to have at least one instance of central running, and machine `designate-manage` is running on will need access to the messaging queue

```
designate-manage pool generate_file --file output.yml
```

This will create a YAML file, with all the currently defined pools, and all of their config.

We suggest this is then migrated into a config management system, or other document management system.

From this point on all updates to pools should be done by updating this file, and running:

```
designate-manage pool update --file /path/to/file.yml
```

Pools - Step by Step

1. Ensure there is not 2 pools with the same name.
2. Stop all Designate Services.
3. Deploy new Mitaka code
4. Start `designate-central`
5. **Run**

   ```
   designate-manage pool export_from_config --file output.yml
   ```

6. Ensure the output file is correct (reference sample file for each value)
7. Run

   ```
   designate-manage pool update --file output.yml --dry_run True --
   ↪delete True
   ```

8. Ensure the output of this command is not removing any Pools
9. Run

   ```
   designate-manage pool update --file output.yml --delete True
   ```

10. Start the remaining designate services.

Upgrading to Newton from Mitaka

The Newton release of Designate adds two new services `designate-producer`, `designate-worker`. These replace `designate-zone-manager` and `designate-pool-manager`, respectively. In a future cycle, the old services will be removed, and the new ones will be enabled by default. In Newton, you must enable the new services yourself. Designate will work with both configurations, as there is no breaking change from Mitaka.

Breaking Changes

The default port the `designate-agent` service listens on has changed from 53 to 5358. This matches the port we have always used in the sample configuration, and the port used in the agent backend class.

Upgrading Code and Enabling Services

To enable the new services with minimal impact, the following process can be followed. This assumes you have all Mitaka Designate services running.

1. Deploy the Newton code.
2. Add the `[service:worker]` and `[service:producer]` sections to your configuration file. Ensure `enabled` and `notify` in the worker section are `True`.

```
[service:worker]
enabled = True
#workers = None
#threads = 1000
#threshold_percentage = 100
#poll_timeout = 30
#poll_retry_interval = 15
#poll_max_retries = 10
#poll_delay = 5
notify = True

[service:producer]
#workers = None
#threads = 1000
# Can be any/all of: periodic_exists, delayed_notify, worker_
↪periodic_recovery
# None => All tasks enabled
#enabled_tasks = None

[producer_task:domain_purge]
#interval = 3600  # 1h
#batch_size = 100
#time_threshold = 604800  # 7 days

[producer_task:delayed_notify]
#interval = 5

[producer_task:worker_periodic_recovery]
#interval = 120
```

3. Stop the `designate-pool-manager` and `designate-zone-manager` processes.
4. Restart the `designate-api`, `designate-central` and `designate-mdns` services.
5. Start the `designate-producer` and `designate-worker` services.

New Features

- `designate-mdns`, `designate-agent` and `designate-api` can now bind to multiple host:port pairs via the new listen configuration arguments for each service.
- New pool scheduler attribute filter for scheduling zones across pools. This can be enabled in the `[service:central]` section of the config by adding `attribute` to the list of values in the `filters` option.

- An experimental agent backend to support TinyDNS, the DNS resolver from the djbdns tools.
- An experimental agent backend to support Knot DNS 2
- A new recordset api `/v2/recordsets` is exposed, docs can be found here.
- Designate services now report running status. The information is exposed via api.
- The quotas API from the admin API has been ported to /v2 with some changes and is now stable.

Deprecation Notices

- `designate-api`s api_host and api_port configuration options have been deprecated, please use the new combined listen argument in place of these.
- `designate-mdns`s host and port configuration options have been deprecated, please use the new combined listen argument in place of these.
- `designate-agents`s host and port configuration options have been deprecated, please use the new combined listen argument in place of these.
- `designate-zone-manager` and `designate-pool-manager` are now deprecated and will be removed in a future release.

Upgrading to Ocata from Newton

Upgrading Code and Enabling Services

1. Deploy Ocata code or packages.
2. Restart all services. See the Newton upgrade guide for enabling `designate-producer` and `designate-worker`.

New Features

- The notifications Designate emits via MQ are now pluggable, drivers are defined by python entry-points and the new `notification_plugin` option in the `DEFAULT` config section enables selection. By default, the notifications have not changed. There is an `audit` plugin that can be used, if desired.
- Scheduling zones across pools. See *Pool Scheduler Filters* for more details.

Deprecation Notices

- `designate-zone-manager` and `designate-pool-manager` remain deprecated and will be removed in a future release.

1.5.14 Troubleshooting

I have a broken zone

A zone is considered broken when it is not receiving updates anymore. Its status can be ERROR if Designate detected the error condition or it can be stuck in PENDING for a long time.

Review the logs from the API, Central, Producer, Worker and MiniDNS. Identify the transaction ID of the last successful change and the first failing change. Using the ID, you can filter logs from the Designate components that are related to the same transaction. Look for log messages with ERROR level before and after the first failing update.

Failures in updating a zone are usually related to problems in Producer, Worker, MiniDNS or the database.

Ensure the services are running and network connectivity is not impaired.

Transient network issues can be the cause of a broken zone. Producer and Worker are stateful services and perform attempts at restoring failing zones over time. Restarting the services will trigger new attempts.

I have a broken pool

I deleted a zone but its still in the database

Deleted zones are flagged with status set to DELETED and task set to NONE once the deletion process terminates successfully.

What ports should be open?

Port numbers are configurable: review your designate.conf

The default values are:

Component (header rows optional)	Protocol	Port numbers
API	TCP	9001
Keystone (external)	TCP	35357
MiniDNS	TCP	5354
	UDP	5354
MySQL	TCP	3306
RabbitMQ	TCP	5672
Resolvers	TCP	53
	UDP	53
ZooKeeper	TCP	2181
	TCP	2888,3888

What network protocol are used?

HTTP[S] by the API, RabbitMQ and the MySQL protocol by most components, DNS (resolution and XFR), ZooKeeper, Memcached.

What needs access to the Database?

Central, MiniDNS

What needs access to RabbitMQ?

The API, Central, Producer, Worker, MiniDNS

What needs access to ZooKeeper?

Pool and Producer

What needs access to Memcached?

API and Worker

How do I monitor Designate?

Designate can be monitored by various monitoring systems listed here

What are useful metrics to monitor?

- General host monitoring, i.e. CPU load, memory usage, disk and network I/O
- MySQL performance, errors and free disk space
- Number of zones in ACTIVE, PENDING and ERROR status
- API queries per second, broken down by read and write operation on zones, records, etc
- Zone change propagation time i.e. how long does it takes for a record update to reach the resolvers
- Log messages containing having ERROR level
- Quotas utilization i.e. number of existing records/zones against the maximum allowed
- Memcached, RabbitMQ, ZooKeeper performance and errors

What are useful metrics to review first during an incident?

- Host, network and MySQL performance metrics
- Number of zones in ACTIVE, PENDING and ERROR status
- Log messages containing having ERROR level

1.5.15 Sample configuration files

Configuration files can alter how designate behaves at runtime and by default are located in `/etc/designate/`. Links to sample configuration files can be found below:

policy.yaml

Use the `policy.yaml` file to define additional access controls that apply to the DNS service:

```
#"admin": "role:admin or is_admin:True"

#"owner": "project_id:%(tenant_id)s"

#"admin_or_owner": "rule:admin or rule:owner"

#"default": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "default":"rule:admin_or_owner" has been deprecated since W in favor
# of "default":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The designate API now supports system scope and default roles.

```

(continues on next page)

(continued from previous page)

```
# Create blacklist.
# POST  /v2/blacklists
# Intended scope(s): project
#"create_blacklist": "role:admin"

# DEPRECATED
# "create_blacklist":"rule:admin" has been deprecated since W in favor
# of "create_blacklist":"role:admin".
# The blacklist API now supports system scope and default roles.

# Find blacklists.
# GET  /v2/blacklists
# Intended scope(s): project
#"find_blacklists": "role:admin"

# DEPRECATED
# "find_blacklists":"rule:admin" has been deprecated since W in favor
# of "find_blacklists":"role:admin".
# The blacklist API now supports system scope and default roles.

# Get blacklist.
# GET  /v2/blacklists/{blacklist_id}
# Intended scope(s): project
#"get_blacklist": "role:admin"

# DEPRECATED
# "get_blacklist":"rule:admin" has been deprecated since W in favor of
# "get_blacklist":"role:admin".
# The blacklist API now supports system scope and default roles.

# Update blacklist.
# PATCH  /v2/blacklists/{blacklist_id}
# Intended scope(s): project
#"update_blacklist": "role:admin"

# DEPRECATED
# "update_blacklist":"rule:admin" has been deprecated since W in favor
# of "update_blacklist":"role:admin".
# The blacklist API now supports system scope and default roles.

# Delete blacklist.
# DELETE  /v2/blacklists/{blacklist_id}
# Intended scope(s): project
#"delete_blacklist": "role:admin"

# DEPRECATED
# "delete_blacklist":"rule:admin" has been deprecated since W in favor
# of "delete_blacklist":"role:admin".
# The blacklist API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# Allowed bypass the blacklist.
# POST  /v2/zones
# Intended scope(s): project
#"use_blacklisted_zone": "role:admin"

# DEPRECATED
# "use_blacklisted_zone":"rule:admin" has been deprecated since W in
# favor of "use_blacklisted_zone":"role:admin".
# The blacklist API now supports system scope and default roles.

# Action on all tenants.
# Intended scope(s): project
#"all_tenants": "role:admin"

# DEPRECATED
# "all_tenants":"rule:admin" has been deprecated since W in favor of
# "all_tenants":"role:admin".
# The designate API now supports system scope and default roles.

# Edit managed records.
# Intended scope(s): project
#"edit_managed_records": "role:admin"

# DEPRECATED
# "edit_managed_records":"rule:admin" has been deprecated since W in
# favor of "edit_managed_records":"role:admin".
# The designate API now supports system scope and default roles.

# Use low TTL.
# Intended scope(s): project
#"use_low_ttl": "role:admin"

# DEPRECATED
# "use_low_ttl":"rule:admin" has been deprecated since W in favor of
# "use_low_ttl":"role:admin".
# The designate API now supports system scope and default roles.

# Accept sudo from user to tenant.
# Intended scope(s): project
#"use_sudo": "role:admin"

# DEPRECATED
# "use_sudo":"rule:admin" has been deprecated since W in favor of
# "use_sudo":"role:admin".
# The designate API now supports system scope and default roles.

# Clean backend resources associated with zone
# Intended scope(s): project
```

(continues on next page)

(continued from previous page)

```
#"hard_delete": "role:admin"

# DEPRECATED
# "hard_delete":"rule:admin" has been deprecated since W in favor of
# "hard_delete":"role:admin".
# The designate API now supports system scope and default roles.

# Create pool.
# Intended scope(s): project
#"create_pool": "role:admin"

# DEPRECATED
# "create_pool":"rule:admin" has been deprecated since W in favor of
# "create_pool":"role:admin".
# The pool API now supports system scope and default roles.

# Find pool.
# GET  /v2/pools
# Intended scope(s): project
#"find_pools": "role:admin"

# DEPRECATED
# "find_pools":"rule:admin" has been deprecated since W in favor of
# "find_pools":"role:admin".
# The pool API now supports system scope and default roles.

# Find pools.
# GET  /v2/pools
# Intended scope(s): project
#"find_pool": "role:admin"

# DEPRECATED
# "find_pool":"rule:admin" has been deprecated since W in favor of
# "find_pool":"role:admin".
# The pool API now supports system scope and default roles.

# Get pool.
# GET  /v2/pools/{pool_id}
# Intended scope(s): project
#"get_pool": "role:admin"

# DEPRECATED
# "get_pool":"rule:admin" has been deprecated since W in favor of
# "get_pool":"role:admin".
# The pool API now supports system scope and default roles.

# Update pool.
# Intended scope(s): project
#"update_pool": "role:admin"
```

(continues on next page)

(continued from previous page)

```
# DEPRECATED
# "update_pool":"rule:admin" has been deprecated since W in favor of
# "update_pool":"role:admin".
# The pool API now supports system scope and default roles.

# Delete pool.
# Intended scope(s): project
#"delete_pool": "role:admin"

# DEPRECATED
# "delete_pool":"rule:admin" has been deprecated since W in favor of
# "delete_pool":"role:admin".
# The pool API now supports system scope and default roles.

# load and set the pool to the one provided in the Zone attributes.
# POST  /v2/zones
# Intended scope(s): project
#"zone_create_forced_pool": "role:admin"

# DEPRECATED
# "zone_create_forced_pool":"rule:admin" has been deprecated since W
# in favor of "zone_create_forced_pool":"role:admin".
# The pool API now supports system scope and default roles.

# View Current Project's Quotas.
# GET  /v2/quotas
# Intended scope(s): project
#"get_quotas": "(role:admin) or (role:reader and project_id:%(project_id)s)␣
↪or (True:%(all_tenants)s and role:reader)"

# DEPRECATED
# "get_quotas":"rule:admin_or_owner" has been deprecated since W in
# favor of "get_quotas":"(role:admin) or (role:reader and
# project_id:%(project_id)s) or (True:%(all_tenants)s and
# role:reader)".
# The quota API now supports system scope and default roles.

# Set Quotas.
# PATCH  /v2/quotas/{project_id}
# Intended scope(s): project
#"set_quota": "role:admin"

# DEPRECATED
# "set_quota":"rule:admin" has been deprecated since W in favor of
# "set_quota":"role:admin".
# The quota API now supports system scope and default roles.

# Reset Quotas.
```

(continues on next page)

(continued from previous page)

```
# DELETE  /v2/quotas/{project_id}
# Intended scope(s): project
#"reset_quotas": "role:admin"

# DEPRECATED
# "reset_quotas":"rule:admin" has been deprecated since W in favor of
# "reset_quotas":"role:admin".
# The quota API now supports system scope and default roles.

# Find records.
# GET  /v2/reverse/floatingips/{region}:{floatingip_id}
# GET  /v2/reverse/floatingips
# Intended scope(s): project
#"find_records": "(role:admin) or (role:reader and project_id:%(project_id)s)"

# DEPRECATED
# "find_records":"rule:admin_or_owner" has been deprecated since W in
# favor of "find_records":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The records API now supports system scope and default roles.

# Intended scope(s): project
#"count_records": "(role:admin) or (role:reader and project_id:%(project_id)s)
↪"

# DEPRECATED
# "count_records":"rule:admin_or_owner" has been deprecated since W in
# favor of "count_records":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The records API now supports system scope and default roles.

# Create Recordset
# POST  /v2/zones/{zone_id}/recordsets
# Intended scope(s): project
#"create_recordset": "(role:member and project_id:%(project_id)s) and ('PRIMARY
↪':%(zone_type)s) or (role:admin) and ('PRIMARY':%(zone_type)s) or␣
↪(role:admin) and ('SECONDARY':%(zone_type)s) or ('True':%(zone_shared)s) and (
↪'PRIMARY':%(zone_type)s)"

# DEPRECATED
# "create_recordset":"('PRIMARY':%(zone_type)s AND
# (rule:admin_or_owner OR 'True':%(zone_shared)s)) OR
# ('SECONDARY':%(zone_type)s AND is_admin:True)" has been deprecated
# since W in favor of "create_recordset":"(role:member and
# project_id:%(project_id)s) and ('PRIMARY':%(zone_type)s) or
# (role:admin) and ('PRIMARY':%(zone_type)s) or (role:admin) and
# ('SECONDARY':%(zone_type)s) or ('True':%(zone_shared)s) and
# ('PRIMARY':%(zone_type)s)".
# The record set API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# Intended scope(s): project
#"get_recordsets": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "get_recordsets":"rule:admin_or_owner" has been deprecated since W
# in favor of "get_recordsets":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The record set API now supports system scope and default roles.

# Get recordset
# GET  /v2/zones/{zone_id}/recordsets/{recordset_id}
# Intended scope(s): project
#"get_recordset": "(role:admin) or (role:reader and project_id:%(project_
↪id)s) or ('True':%(zone_shared)s)"

# DEPRECATED
# "get_recordset":"rule:admin_or_owner or ('True':%(zone_shared)s)"
# has been deprecated since W in favor of
# "get_recordset":"(role:admin) or (role:reader and
# project_id:%(project_id)s) or ('True':%(zone_shared)s)".
# The record set API now supports system scope and default roles.

# List a Recordset in a Zone
# Intended scope(s): project
#"find_recordset": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "find_recordset":"rule:admin_or_owner" has been deprecated since W
# in favor of "find_recordset":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The record set API now supports system scope and default roles.

# List Recordsets in a Zone
# GET  /v2/zones/{zone_id}/recordsets
# Intended scope(s): project
#"find_recordsets": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "find_recordsets":"rule:admin_or_owner" has been deprecated since W
# in favor of "find_recordsets":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The record set API now supports system scope and default roles.

# Update recordset
# PUT  /v2/zones/{zone_id}/recordsets/{recordset_id}
```

(continues on next page)

(continued from previous page)

```
# Intended scope(s): project
#"update_recordset": "(role:member and project_id:%(project_id)s) and ('PRIMARY
↪':%(zone_type)s) or (role:admin) and ('PRIMARY':%(zone_type)s) or␣
↪(role:admin) and ('SECONDARY':%(zone_type)s) or role:member and (project_id:
↪%(recordset_project_id)s) and ('PRIMARY':%(zone_type)s)"

# DEPRECATED
# "update_recordset":"rule:admin or ('PRIMARY':%(zone_type)s and
# (rule:owner or project_id:%(recordset_project_id)s))" has been
# deprecated since W in favor of "update_recordset":"(role:member and
# project_id:%(project_id)s) and ('PRIMARY':%(zone_type)s) or
# (role:admin) and ('PRIMARY':%(zone_type)s) or (role:admin) and
# ('SECONDARY':%(zone_type)s) or role:member and
# (project_id:%(recordset_project_id)s) and
# ('PRIMARY':%(zone_type)s)".
# The record set API now supports system scope and default roles.

# Delete RecordSet
# DELETE  /v2/zones/{zone_id}/recordsets/{recordset_id}
# Intended scope(s): project
#"delete_recordset": "(role:member and project_id:%(project_id)s) and ('PRIMARY
↪':%(zone_type)s) or (role:admin) and ('PRIMARY':%(zone_type)s) or␣
↪(role:admin) and ('SECONDARY':%(zone_type)s) or role:member and (project_id:
↪%(recordset_project_id)s) and ('PRIMARY':%(zone_type)s)"

# DEPRECATED
# "delete_recordset":"rule:admin or ('PRIMARY':%(zone_type)s and
# (rule:owner or project_id:%(recordset_project_id)s))" has been
# deprecated since W in favor of "delete_recordset":"(role:member and
# project_id:%(project_id)s) and ('PRIMARY':%(zone_type)s) or
# (role:admin) and ('PRIMARY':%(zone_type)s) or (role:admin) and
# ('SECONDARY':%(zone_type)s) or role:member and
# (project_id:%(recordset_project_id)s) and
# ('PRIMARY':%(zone_type)s)".
# The record set API now supports system scope and default roles.

# Count recordsets
# Intended scope(s): project
#"count_recordset": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "count_recordset":"rule:admin_or_owner" has been deprecated since W
# in favor of "count_recordset":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The record set API now supports system scope and default roles.

# Find a single Service Status
# GET  /v2/service_status/{service_id}
```

(continues on next page)

(continued from previous page)

```
# Intended scope(s): project
#"find_service_status": "role:admin"

# DEPRECATED
# "find_service_status":"rule:admin" has been deprecated since W in
# favor of "find_service_status":"role:admin".
# The service status API now supports system scope and default roles.

# List service statuses.
# GET  /v2/service_status
# Intended scope(s): project
#"find_service_statuses": "role:admin"

# DEPRECATED
# "find_service_statuses":"rule:admin" has been deprecated since W in
# favor of "find_service_statuses":"role:admin".
# The service status API now supports system scope and default roles.

# Intended scope(s): project
#"update_service_status": "role:admin"

# DEPRECATED
# "update_service_status":"rule:admin" has been deprecated since W in
# favor of "update_service_status":"role:admin".
# The service status API now supports system scope and default roles.

# Get a Zone Share
# GET  /v2/zones/{zone_id}/shares/{zone_share_id}
# Intended scope(s): project
#"get_zone_share": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "get_zone_share":"rule:admin_or_owner" has been deprecated since W
# in favor of "get_zone_share":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The shared zones API now supports system scope and default roles.

# Share a Zone
# POST  /v2/zones/{zone_id}/shares
# Intended scope(s): project
#"share_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "share_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "share_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The shared zones API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# List Shared Zones
# GET  /v2/zones/{zone_id}/shares
#"find_zone_shares": "@"

# Check the can query for a specific projects shares.
# Intended scope(s): project
#"find_project_zone_share": "(role:admin) or (role:member and project_id:
↪%(project_id)s)"

# DEPRECATED
# "find_project_zone_share":"rule:admin_or_owner" has been deprecated
# since W in favor of "find_project_zone_share":"(role:admin) or
# (role:member and project_id:%(project_id)s)".
# The shared zones API now supports system scope and default roles.

# Unshare Zone
# DELETE  /v2/zones/{zone_id}/shares/{shared_zone_id}
# Intended scope(s): project
#"unshare_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "unshare_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "unshare_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The shared zones API now supports system scope and default roles.

# Find all Tenants.
# Intended scope(s): project
#"find_tenants": "role:admin"

# DEPRECATED
# "find_tenants":"rule:admin" has been deprecated since W in favor of
# "find_tenants":"role:admin".
# The tenant API now supports system scope and default roles.

# Get all Tenants.
# Intended scope(s): project
#"get_tenant": "role:admin"

# DEPRECATED
# "get_tenant":"rule:admin" has been deprecated since W in favor of
# "get_tenant":"role:admin".
# The tenant API now supports system scope and default roles.

# Count tenants
# Intended scope(s): project
#"count_tenants": "role:admin"

# DEPRECATED
```

(continues on next page)

(continued from previous page)

```
# "count_tenants":"rule:admin" has been deprecated since W in favor of
# "count_tenants":"role:admin".
# The tenant API now supports system scope and default roles.

# Create Tld
# POST  /v2/tlds
# Intended scope(s): project
#"create_tld": "role:admin"

# DEPRECATED
# "create_tld":"rule:admin" has been deprecated since W in favor of
# "create_tld":"role:admin".
# The top-level domain API now supports system scope and default
# roles.

# List Tlds
# GET  /v2/tlds
# Intended scope(s): project
#"find_tlds": "role:admin"

# DEPRECATED
# "find_tlds":"rule:admin" has been deprecated since W in favor of
# "find_tlds":"role:admin".
# The top-level domain API now supports system scope and default
# roles.

# Show Tld
# GET  /v2/tlds/{tld_id}
# Intended scope(s): project
#"get_tld": "role:admin"

# DEPRECATED
# "get_tld":"rule:admin" has been deprecated since W in favor of
# "get_tld":"role:admin".
# The top-level domain API now supports system scope and default
# roles.

# Update Tld
# PATCH  /v2/tlds/{tld_id}
# Intended scope(s): project
#"update_tld": "role:admin"

# DEPRECATED
# "update_tld":"rule:admin" has been deprecated since W in favor of
# "update_tld":"role:admin".
# The top-level domain API now supports system scope and default
# roles.

# Delete Tld
```

(continues on next page)

(continued from previous page)

```
# DELETE  /v2/tlds/{tld_id}
# Intended scope(s): project
#"delete_tld": "role:admin"

# DEPRECATED
# "delete_tld":"rule:admin" has been deprecated since W in favor of
# "delete_tld":"role:admin".
# The top-level domain API now supports system scope and default
# roles.

# Create Tsigkey
# POST  /v2/tsigkeys
# Intended scope(s): project
#"create_tsigkey": "role:admin"

# DEPRECATED
# "create_tsigkey":"rule:admin" has been deprecated since W in favor
# of "create_tsigkey":"role:admin".
# The tsigkey API now supports system scope and default roles.

# List Tsigkeys
# GET  /v2/tsigkeys
# Intended scope(s): project
#"find_tsigkeys": "role:admin"

# DEPRECATED
# "find_tsigkeys":"rule:admin" has been deprecated since W in favor of
# "find_tsigkeys":"role:admin".
# The tsigkey API now supports system scope and default roles.

# Show a Tsigkey
# GET  /v2/tsigkeys/{tsigkey_id}
# Intended scope(s): project
#"get_tsigkey": "role:admin"

# DEPRECATED
# "get_tsigkey":"rule:admin" has been deprecated since W in favor of
# "get_tsigkey":"role:admin".
# The tsigkey API now supports system scope and default roles.

# Update Tsigkey
# PATCH  /v2/tsigkeys/{tsigkey_id}
# Intended scope(s): project
#"update_tsigkey": "role:admin"

# DEPRECATED
# "update_tsigkey":"rule:admin" has been deprecated since W in favor
# of "update_tsigkey":"role:admin".
# The tsigkey API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# Delete a Tsigkey
# DELETE  /v2/tsigkeys/{tsigkey_id}
# Intended scope(s): project
#"delete_tsigkey": "role:admin"

# DEPRECATED
# "delete_tsigkey":"rule:admin" has been deprecated since W in favor
# of "delete_tsigkey":"role:admin".
# The tsigkey API now supports system scope and default roles.

# Create Zone
# POST  /v2/zones
# Intended scope(s): project
#"create_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "create_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "create_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Intended scope(s): project
#"get_zones": "(role:admin) or (role:reader and project_id:%(project_id)s)"

# DEPRECATED
# "get_zones":"rule:admin_or_owner" has been deprecated since W in
# favor of "get_zones":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Get Zone
# GET  /v2/zones/{zone_id}
# Intended scope(s): project
#"get_zone": "(role:admin) or (role:reader and project_id:%(project_id)s) or (
↪'True':%(zone_shared)s)"

# DEPRECATED
# "get_zone":"rule:admin_or_owner or ('True':%(zone_shared)s)" has
# been deprecated since W in favor of "get_zone":"(role:admin) or
# (role:reader and project_id:%(project_id)s) or
# ('True':%(zone_shared)s)".
# The zone API now supports system scope and default roles.

# Intended scope(s): project
#"get_zone_servers": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
```

(continues on next page)

(continued from previous page)

```
# "get_zone_servers":"rule:admin_or_owner" has been deprecated since W
# in favor of "get_zone_servers":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Get the Name Servers for a Zone
# GET  /v2/zones/{zone_id}/nameservers
# Intended scope(s): project
#"get_zone_ns_records": "(role:admin) or (role:reader and project_id:
↪%(project_id)s)"

# DEPRECATED
# "get_zone_ns_records":"rule:admin_or_owner" has been deprecated
# since W in favor of "get_zone_ns_records":"(role:admin) or
# (role:reader and project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# List existing zones
# GET  /v2/zones
# Intended scope(s): project
#"find_zones": "(role:admin) or (role:reader and project_id:%(project_id)s)"

# DEPRECATED
# "find_zones":"rule:admin_or_owner" has been deprecated since W in
# favor of "find_zones":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Update Zone
# PATCH  /v2/zones/{zone_id}
# Intended scope(s): project
#"update_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "update_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "update_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Delete Zone
# DELETE  /v2/zones/{zone_id}
# Intended scope(s): project
#"delete_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "delete_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "delete_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# Manually Trigger an Update of a Secondary Zone
# POST  /v2/zones/{zone_id}/tasks/xfr
# Intended scope(s): project
#"xfr_zone": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "xfr_zone":"rule:admin_or_owner" has been deprecated since W in
# favor of "xfr_zone":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Abandon Zone
# POST  /v2/zones/{zone_id}/tasks/abandon
# Intended scope(s): project
#"abandon_zone": "role:admin"

# DEPRECATED
# "abandon_zone":"rule:admin" has been deprecated since W in favor of
# "abandon_zone":"role:admin".
# The zone API now supports system scope and default roles.

# Intended scope(s): project
#"count_zones": "(role:admin) or (role:reader and project_id:%(project_id)s)"

# DEPRECATED
# "count_zones":"rule:admin_or_owner" has been deprecated since W in
# favor of "count_zones":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Intended scope(s): project
#"count_zones_pending_notify": "(role:admin) or (role:reader and project_id:
↪%(project_id)s)"

# DEPRECATED
# "count_zones_pending_notify":"rule:admin_or_owner" has been
# deprecated since W in favor of
# "count_zones_pending_notify":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone API now supports system scope and default roles.

# Intended scope(s): project
#"purge_zones": "role:admin"

# DEPRECATED
# "purge_zones":"rule:admin" has been deprecated since W in favor of
# "purge_zones":"role:admin".
# The zone API now supports system scope and default roles.
```

(continues on next page)

(continued from previous page)

```
# Pool Move Zone
# POST  /v2/zones/{zone_id}/tasks/pool_move
# Intended scope(s): project
#"pool_move_zone": "role:admin"

# DEPRECATED
# "pool_move_zone":"rule:admin" has been deprecated since W in favor
# of "pool_move_zone":"role:admin".
# The zone API now supports system scope and default roles.

# Retrive a Zone Export from the Designate Datastore
# GET  /v2/zones/tasks/exports/{zone_export_id}/export
# Intended scope(s): project
#"zone_export": "(role:admin) or (role:member and project_id:%(project_id)s)"

# DEPRECATED
# "zone_export":"rule:admin_or_owner" has been deprecated since W in
# favor of "zone_export":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# Create Zone Export
# POST  /v2/zones/{zone_id}/tasks/export
# Intended scope(s): project
#"create_zone_export": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "create_zone_export":"rule:admin_or_owner" has been deprecated since
# W in favor of "create_zone_export":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# List Zone Exports
# GET  /v2/zones/tasks/exports
# Intended scope(s): project
#"find_zone_exports": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "find_zone_exports":"rule:admin_or_owner" has been deprecated since
# W in favor of "find_zone_exports":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# Get Zone Exports
# GET  /v2/zones/tasks/exports/{zone_export_id}
# Intended scope(s): project
```

(continues on next page)

(continued from previous page)

```
#"get_zone_export": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "get_zone_export":"rule:admin_or_owner" has been deprecated since W
# in favor of "get_zone_export":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# Update Zone Exports
# POST  /v2/zones/{zone_id}/tasks/export
# Intended scope(s): project
#"update_zone_export": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "update_zone_export":"rule:admin_or_owner" has been deprecated since
# W in favor of "update_zone_export":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# Delete a zone export
# DELETE  /v2/zones/tasks/exports/{zone_export_id}
# Intended scope(s): project
#"delete_zone_export": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "delete_zone_export":"rule:admin_or_owner" has been deprecated since
# W in favor of "delete_zone_export":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone export API now supports system scope and default roles.

# Create Zone Import
# POST  /v2/zones/tasks/imports
# Intended scope(s): project
#"create_zone_import": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "create_zone_import":"rule:admin_or_owner" has been deprecated since
# W in favor of "create_zone_import":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone import API now supports system scope and default roles.

# List all Zone Imports
# GET  /v2/zones/tasks/imports
# Intended scope(s): project
#"find_zone_imports": "(role:admin) or (role:reader and project_id:%(project_
```

(continues on next page)

(continued from previous page)

```
↪id)s)"

# DEPRECATED
# "find_zone_imports":"rule:admin_or_owner" has been deprecated since
# W in favor of "find_zone_imports":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone import API now supports system scope and default roles.

# Get Zone Imports
# GET  /v2/zones/tasks/imports/{zone_import_id}
# Intended scope(s): project
#"get_zone_import": "(role:admin) or (role:reader and project_id:%(project_
↪id)s)"

# DEPRECATED
# "get_zone_import":"rule:admin_or_owner" has been deprecated since W
# in favor of "get_zone_import":"(role:admin) or (role:reader and
# project_id:%(project_id)s)".
# The zone import API now supports system scope and default roles.

# Update Zone Imports
# POST  /v2/zones/tasks/imports
# Intended scope(s): project
#"update_zone_import": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "update_zone_import":"rule:admin_or_owner" has been deprecated since
# W in favor of "update_zone_import":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone import API now supports system scope and default roles.

# Delete a Zone Import
# DELETE  /v2/zones/tasks/imports/{zone_import_id}
# Intended scope(s): project
#"delete_zone_import": "(role:admin) or (role:member and project_id:%(project_
↪id)s)"

# DEPRECATED
# "delete_zone_import":"rule:admin_or_owner" has been deprecated since
# W in favor of "delete_zone_import":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone import API now supports system scope and default roles.

# Create Zone Transfer Accept
# POST  /v2/zones/tasks/transfer_accepts
# Intended scope(s): project
#"create_zone_transfer_accept": "((role:admin) or (role:member and project_id:
↪%(project_id)s)) or project_id:%(target_project_id)s or None:%(target_
```

(continues on next page)

(continued from previous page)

```
↪project_id)s"

# DEPRECATED
# "create_zone_transfer_accept":"rule:admin_or_owner OR
# project_id:%(target_tenant_id)s OR None:%(target_tenant_id)s" has
# been deprecated since W in favor of
# "create_zone_transfer_accept":"((role:admin) or (role:member and
# project_id:%(project_id)s)) or project_id:%(target_project_id)s or
# None:%(target_project_id)s".
# The zone transfer accept API now supports system scope and default
# roles.

# Get Zone Transfer Accept
# GET  /v2/zones/tasks/transfer_requests/{zone_transfer_accept_id}
# Intended scope(s): project
#"get_zone_transfer_accept": "(role:admin) or (role:reader and project_id:
↪%(project_id)s)"

# DEPRECATED
# "get_zone_transfer_accept":"rule:admin_or_owner" has been deprecated
# since W in favor of "get_zone_transfer_accept":"(role:admin) or
# (role:reader and project_id:%(project_id)s)".
# The zone transfer accept API now supports system scope and default
# roles.

# List Zone Transfer Accepts
# GET  /v2/zones/tasks/transfer_accepts
# Intended scope(s): project
#"find_zone_transfer_accepts": "role:admin"

# DEPRECATED
# "find_zone_transfer_accepts":"rule:admin" has been deprecated since
# W in favor of "find_zone_transfer_accepts":"role:admin".
# The zone transfer accept API now supports system scope and default
# roles.

# Create Zone Transfer Accept
# POST  /v2/zones/{zone_id}/tasks/transfer_requests
# Intended scope(s): project
#"create_zone_transfer_request": "(role:admin) or (role:member and project_id:
↪%(project_id)s)"

# DEPRECATED
# "create_zone_transfer_request":"rule:admin_or_owner" has been
# deprecated since W in favor of
# "create_zone_transfer_request":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone transfer request API now supports system scope and default
# roles.
```

(continues on next page)

(continued from previous page)

```
# Show a Zone Transfer Request
# GET  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
# Intended scope(s): project
#"get_zone_transfer_request": "((role:admin) or (role:member and project_id:
↪%(project_id)s)) or project_id:%(target_project_id)s or None:%(target_
↪project_id)s"

# DEPRECATED
# "get_zone_transfer_request":"rule:admin_or_owner OR
# project_id:%(target_tenant_id)s OR None:%(target_tenant_id)s" has
# been deprecated since W in favor of
# "get_zone_transfer_request":"((role:admin) or (role:member and
# project_id:%(project_id)s)) or project_id:%(target_project_id)s or
# None:%(target_project_id)s".
# The zone transfer request API now supports system scope and default
# roles.

# Intended scope(s): project
#"get_zone_transfer_request_detailed": "(role:admin) or (role:reader and␣
↪project_id:%(project_id)s)"

# DEPRECATED
# "create_zone_transfer_request":"rule:admin_or_owner" has been
# deprecated since W in favor of
# "get_zone_transfer_request_detailed":"(role:admin) or (role:reader
# and project_id:%(project_id)s)".
# The zone transfer request API now supports system scope and default
# roles.
# WARNING: A rule name change has been identified.
#          This may be an artifact of new rules being
#          included which require legacy fallback
#          rules to ensure proper policy behavior.
#          Alternatively, this may just be an alias.
#          Please evaluate on a case by case basis
#          keeping in mind the format for aliased
#          rules is:
#          "old_rule_name": "new_rule_name".
# "create_zone_transfer_request": "rule:get_zone_transfer_request_detailed"

# List Zone Transfer Requests
# GET  /v2/zones/tasks/transfer_requests
#"find_zone_transfer_requests": "@"

# Update a Zone Transfer Request
# PATCH  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
# Intended scope(s): project
#"update_zone_transfer_request": "(role:admin) or (role:member and project_id:
↪%(project_id)s)"
```

(continues on next page)

(continued from previous page)

```
# DEPRECATED
# "update_zone_transfer_request":"rule:admin_or_owner" has been
# deprecated since W in favor of
# "update_zone_transfer_request":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone transfer request API now supports system scope and default
# roles.

# Delete a Zone Transfer Request
# DELETE  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
# Intended scope(s): project
#"delete_zone_transfer_request": "(role:admin) or (role:member and project_id:
↪%(project_id)s)"

# DEPRECATED
# "delete_zone_transfer_request":"rule:admin_or_owner" has been
# deprecated since W in favor of
# "delete_zone_transfer_request":"(role:admin) or (role:member and
# project_id:%(project_id)s)".
# The zone transfer request API now supports system scope and default
# roles.
```

designate.conf

Please refer to the online version of this documentation for a full config file example.

1.5.16 DNS Server Driver Support Matrix

This info should be maintained along with the list of current driver maintainers responsible for the Non Integrated backends. The upkeep of this list will fall on the PTL or his/her delegate.

Should a backends grade be in dispute, it falls on the current project PTL to make the final decision after listening to all sides concerns.

Grades

Grade	Description
Integrated	Tested on every commit by the OpenStack CI Infrastructure, and maintained by designate developers as a reference backend
Master Compatible	Tested on every commit by 3rd party testing, and has a person or group dedicated to maintaining compatibility on a regular basis
Release Compatible	Not necessarily tested on every commit, but has a maintainer committed to ensuring compatibility for each release
Untested	All other backends in the designate repository
Failing	Backends that were previously Compatible, but tests are now failing on a regular basis.
Known Broken	Backends that do not work, and have been broken with no sign of any fixes
Experimental	Backends that are under development, and may change at any time
Deprecated	Backends have been superseded, and will be removed in the future
End of Life	A backend that has reached its end of life and has been removed from the code.

Backends - Summary

Backend	Status	Type	In Tree	Notes
Bind9	Integrated	xfr	✓	None
Power DNS 4	Integrated	xfr	✓	None
Akamai DNS v2	Untested	xfr	✓	None
Designate to Designate	Untested	xfr	✓	None
DynECT	Untested	xfr	✓	None
Infoblox (XFR)	Untested	xfr	✓	None
NS1 DNS	Untested	xfr	✓	None
NSD4	Untested	xfr	✓	None
Akamai eDNS	End of Life	xfr	×	Akamai has turned off the eDNS API - see https://community.akamai.com/customers/s/article/Big-Changes-Coming-to-Fast-DNS-in-2018

Backend Details

Bind9

Grade	Integrated
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Power DNS 4

Grade	Integrated
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Designate to Designate

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

DynECT

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Akamai eDNS

Grade	End of Life
In Tree	×
Maintainers	Designate Team
Repository	Designate Repository
Notes	Akamai has turned off the eDNS API - see https://community.akamai.com/customers/s/article/Big-Changes-Coming-to-Fast-DNS-in-2018

Akamai DNS v2

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Infoblox (XFR)

Grade	Untested
In Tree	✓
Maintainers	Infoblox OpenStack Team <openstack-maintainer@infoblox.com>
Repository	Designate Repository
Notes	None

NSD4

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

NS1 DNS

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

1.6 Designate Configuration Guide

Designate configuration is needed for getting it work correctly either with real OpenStack environment or without OpenStack environment.

NOTE: The most of the following operations should performed in designate directory.

1. You can generate full sample *designate.conf* (if it does not already exist):

```
$ oslo-config-generator --config-file etc/designate/designate-config-
↪generator.conf --output-file /etc/designate/designate.conf
```

2. You can generate full sample of default policies *policy.yaml* (if it does not already exist):

```
$ oslopolicy-sample-generator --config-file etc/designate/designate-
↪policy-generator.conf --output-file /etc/designate/policy.yaml
```

For more information on Designate configuration see the following sections

1.7 Command-Line Interface Reference

Users can interact with designate using the Openstack client via the commands provided by the designate plugin

Information on the commands available through Designates Command Line Interface (CLI) can be found in this section.

1.7.1 Designate Manage CLI

This chapter documents **designate-manage**

For help on a specific **designate** command, enter:

```
$ designate-manage COMMAND --help
```

designate-manage

designate-manage usage

```
usage: designate-manage [-h] [--config-dir DIR] [--config-file PATH] [--debug]
                        [--log-config-append PATH] [--log-date-format DATE_
↪FORMAT]
                        [--log-dir LOG_DIR] [--log-file PATH] [--nodebug]
                        [--nouse-syslog] [--nouse-syslog-rfc-format] [--
↪noverbose]
                        [--nowatch-log-file]
                        [--syslog-log-facility SYSLOG_LOG_FACILITY] [--use-
↪syslog]
                        [--use-syslog-rfc-format] [--verbose] [--version]
                        [--watch-log-file]
```

designate optional arguments

--config-dir DIR
: Path to a config directory to pull *.conf files from. This file set is sorted, so as to provide a predictable parse order if individual options are over-ridden. The set is parsed after the file(s) specified via previous config-file, arguments hence over-ridden options in the directory take precedence.

--config-file PATH
: Path to a config file to use. Multiple config files can be specified, with values in later files taking precedence. Defaults to None.

--debug, -d
: If set to true, the logging level will be set to DEBUG instead of the default INFO level.

--log-config-append PATH, --log_config PATH
The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, logging_context_format_string).

--log-date-format DATE_FORMAT
Defines the format string for %(asctime)s in log records. Default: None . This option is ignored if log_config_append is set.

--log-dir LOG_DIR, --logdir LOG_DIR
(Optional) The base directory used for relative log_file paths. This option is ignored if log_config_append is set.

--log-file PATH, --logfile PATH
(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

--nodebug
The inverse of debug

--nouse-syslog
The inverse of use-syslog

--nouse-syslog-rfc-format
The inverse of use-syslog-rfc-format

--noverbose
The inverse of verbose

--nowatch-log-file
The inverse of watch-log-file

--syslog-log-facility SYSLOG_LOG_FACILITY
Syslog facility to receive log lines. This option is ignored if log_config_append is set.

--use-syslog
Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if log_config_append is set.

--use-syslog-rfc-format
Enables or disables syslog rfc5424 format for logging. If enabled, prefixes the MSG part of the syslog message with APP-NAME (RFC5424). This option is ignored if log_config_append is set.

--verbose, -v
If set to false, the logging level will be set to WARNING instead of the default INFO level.

--watch-log-file
Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if log_file option is specified and Linux platform is used. This option is ignored if log_config_append is set.

designate-manage pool

```
usage: designate pool [-h] {generate_file,show_config,update} ...

positional arguments:
  {generate_file,show_config,update}
```

designate-manage pool generate_file

```
usage: designate-manage pool generate_file [-h] [--file FILE]
```

Export a YAML copy of the current running pool config

Optional arguments:

-h, --help
show this help message and exit

--file FILE
The path to the file the yaml output should be written to (Defaults to /etc/designate/pools.yaml)

designate-manage pool update

```
usage: designate-manage pool update [-h] [--file FILE] [--delete]
                                    [--dry-run]
```

Update the running pool config from a YAML file

Optional arguments:

-h, --help
show this help message and exit

--file FILE
The path to the file that should be used to update the pools config (Defaults to /etc/designate/pools.yaml)

--delete
Any Pools not listed in the config file will be deleted. .. warning:: This will delete any zones left in this pool

--dry-run
This will simulate what will happen when you run this command

designate-manage pool show

```
usage: designate-manage pool show_config [-h] [--pool_id POOL_ID]
                                         [--all_pools]
```

Show the deployed pools configuration

Optional arguments:

-h, --help
show this help message and exit

--pool_id POOL_ID
ID of the pool to be examined

--all_pools
show the config of all the pools

designate-manage database

designate-manage database sync

```
usage: designate-manage database sync [-h] [--revision REVISION]
```

Update the designate database schema

Optional arguments:

-h, --help
show this help message and exit

--revision REVISION
The version that the designate database should be synced to. (Defaults to latest version)

designate-manage database version

```
usage: designate-manage database version [-h]
```

Show what version of the database schema is currently in place

Optional arguments:

-h, --help
show this help message and exit

1.7.2 Designate Status CLI

This chapter documents **designate-status**.

For help on a specific **designate-status** command, enter:

```
$ designate-status COMMAND --help
```

designate-status

designate-status is a tool that provides routines for checking the status of a Designate deployment.

The standard pattern for executing a **designate-status** command is:

```
designate-status <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
designate-status
```

Categories are:

- upgrade

Detailed descriptions are below.

You can also run with a category argument such as `upgrade` to see a list of all commands in that category:

```
designate-status upgrade
```

The following sections describe the available categories and arguments for **`designate-status`**.

designate-status upgrade

designate-status upgrade check

`designate-status upgrade check`

Performs a release-specific readiness check before running db sync for the new version. This command expects to have complete configuration and access to the database.

Return Codes

Return code	Description
0	All upgrade readiness checks passed successfully and there is nothing to do.
1	At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.
2	There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.
255	An unexpected error occurred.

History of Checks

8.0.0 (Stein)

- Checks that duplicate entries do not exist in the `service_statuses` table.

1.8 Designate Reference

1.8.1 Designate Glossary

The following is a glossary of terms that may be used througout the Designate documentation and code.

Fully Qualified Domain Name
A domain name that includes all levels of the domain hierarchy, including the root domain (represented by a period at the end). Fully Qualified Domain Name is sometimes abreviated as FQDN. Example: `www.example.com.`

Record
The data (also known as the RDATA in RFC1034) part of a recordset. Recordsets may have one or more records. An example of a record for a recordset of type **A** would be an IP address, such as 192.0.2.1.

Recordset
A recordset represents one or more DNS *records* that share the same *Name* and *Type*. For example, a recordset *named* `www.example.com.`, with a *Type* of **A**, may contain two records; 192.0.2.1 and 192.0.2.2.

Zone

A zone represents a namespace in DNS, for example the zone `example.com.` may contain a *recordset* for `www`.

For information on the Designate API, see the API Reference.

This documentation is generated by the Sphinx toolkit and lives in the source tree.

INDEX

A

B

C

D

E

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z