Designate Documentation

Release 11.0.3.dev14

Designate Developers

Dec 05, 2023

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 Installing OpenStack DNS as a Service

1.1.1 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

`designate-agent` component A small python daemon that can be used for a limited sub set of DNS Servers Some DNS Servers requrire commands be run locally, and to do this we use this component.

> **Note:** The majority of the DNS service installs will not need this component.

`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 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:

```
# yum 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:

```
# yum 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 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;
    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
```

(continues on next page)

(continued from previous page)

```
  # 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.
      # 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-agent --config-file /etc/
↪designate/designate.conf
../usr/bin/python /usr/bin/designate-api --config-file /etc/designate/
↪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           |
| 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                                 |
```

(continues on next page)

(continued from previous page)

```
| 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.                         |
+-------------+--------------------------------------+
```

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*.

1.1.2 Quickstart with Kolla

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

1.2 Developer documentation

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

Contents:

1.2.1 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 18.04 VM or newer. 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
```

(continues on next page)

(continued from previous page)

```
VERBOSE=True
LOG_COLOR=True

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

# Enable designate
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=
```

(continues on next page)

(continued from previous page)

```
# Designate D2D Backend
# NOTEs:
# - 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
#enable_plugin rally https://opendev.org/openstack/rally.git master

# Optional Horizon
#enable_service horizon
```

(continues on next page)

(continued from previous page)

```
# Optional Glance
#enable_service g-api,g-reg

# 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:

```
$ source openrc admin admin # For the admin user, admin tenant
$ source openrc admin demo  # For the admin user, demo tenant
$ source openrc demo demo   # For the demo user, demo tenant
```

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

(continues on next page)

(continued from previous page)

```
$ 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   |
+--------------------------------------+------------------+------+----
↪-----------------------------------------------------------------+--
↪------+--------+

$ 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     |
```

(continues on next page)

(continued from previous page)

```
| 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.                         |
+-------------+--------------------------------------+
```

#openstack-dns IRC channel

There is an active IRC channel at irc://freenode.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>
```

(continues on next page)

(continued from previous page)

```
<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:

```
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>
    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

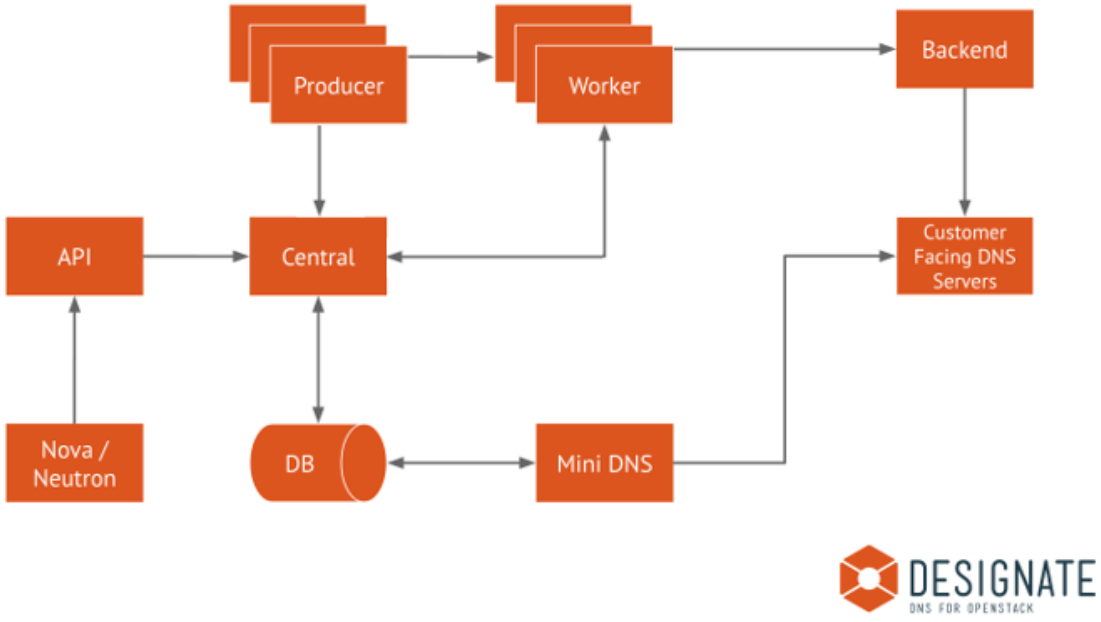
See https://docs.openstack.org/oslo.i18n/latest/user/guidelines.html

```
# 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'})
```

1.2.2 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.

1.2.3 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                          ====
========================================================================
------                          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()`
```

(continues on next page)

(continued from previous page)

```
/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]`

/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
```

(continues on next page)

(continued from previous page)

```
    `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()`

/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)`
```

(continues on next page)

(continued from previous page)

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

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

backend:agent:bind9:
  query-destination = 127.0.0.1
  rndc-config-file = None
  rndc-host = 127.0.0.1
  rndc-key-file = None
  rndc-port = 953
  zone-file-path = /opt/stack/data/designate/zones

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:
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
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
    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.json
  pool-manager-topic = pool_manager
```

(continues on next page)

(continued from previous page)

```
  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

network_api:neutron:
  admin_password = ***
  admin_tenant_name = None
  admin_username = None
  auth_strategy = keystone
  auth_url = None
  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
```

(continues on next page)

(continued from previous page)

```
  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
  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
```

(continues on next page)

(continued from previous page)

```
  pool_timeout = None
  retry_interval = 10
  slave_connection = ***
  sqlite_db = oslo.sqlite
  sqlite_synchronous = True
  use_db_reconnect = False
```

1.2.4 Monasca-Statsd based Metrics

metrics Base

class designate.metrics.**Metrics**

Bases: object

__dict__ = mappingproxy({'__module__': 'designate.metrics', '__init__': <f

__init__()

Initialize self. See help(type(self)) for accurate signature.

__module__ = 'designate.metrics'

__weakref__

list of weak references to the object (if defined)

property client

counter(*a, **kw)

gauge(*a, **kw)

init()

timer()

property timing

1.2.5 Source Code Documentation

API

API Middleware

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

Bases: oslo_middleware.base.Middleware

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

Bases: oslo_middleware.base.Middleware

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

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

Bases: oslo_middleware.base.Middleware

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

Bases: *designate.api.middleware.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: oslo_middleware.base.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: *designate.api.middleware.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: oslo_middleware.base.Middleware

class designate.api.middleware.**TestContextMiddleware**(*application*, *tenant_id=None*, *user_id=None*)
Bases: *designate.api.middleware.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: designate.service.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: designate.plugin.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*)
Delete a DNS zone.

Parameters

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

property mdns_api

ping(*context*)
Ping the Backend service

start()

stop()

update_zone(*context*, *zone*)
Update a DNS zone.

Parameters

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

Backend Akamai

class designate.backend.impl_akamai.**AkamaiBackend**(*target*)
Bases: *designate.backend.base.Backend*

create_zone(*context*, *zone*)
Create a DNS zone

delete_zone(*context*, *zone*)
Delete a DNS zone

exception designate.backend.impl_akamai.**DelegationExists**(**args*, ***kwargs*)

Bases: designate.exceptions.BadRequest, *designate.backend.impl_akamai.EnhancedDNSException*

Raised when an attempt to delete a zone which is still delegated to Akamai is made

error_type = 'delegation_exists'

exception designate.backend.impl_akamai.**DuplicateZone**(**args*, ***kwargs*)

Bases: designate.exceptions.DuplicateZone, *designate.backend.impl_akamai.EnhancedDNSException*

Raised when an attempt to create a zone which is registered to another Akamai account is made

class designate.backend.impl_akamai.**EnhancedDNSClient**(*username*, *password*)

Bases: object

EnhancedDNS SOAP API Client

buildZone(*zoneName*, *masters*, *endCustomerId*, *tsigKeyName=None*, *tsigKey=None*, *tsigAlgorithm=None*)

deleteZone(*zoneName*)

deleteZones(*zoneNames*)

getZone(*zoneName*)

setZone(*zone*)

setZones(*zones*)

exception designate.backend.impl_akamai.**EnhancedDNSException**

Bases: designate.exceptions.Backend

class designate.backend.impl_akamai.**EnhancedDNSHttpAuthenticated**(***kwargs*)

Bases: suds.transport.https.HttpAuthenticated

addenhanceddnsheaders(*request*)

logenhanceddnsheaders(*response*)

send(*request*)

Send soap message. Implementations are expected to handle:

- proxies
- I{HTTP} headers
- cookies
- sending message
- brokering exceptions into L{TransportError}

@param request: A transport request. @type request: L{Request} @return: The reply @rtype: L{Reply} @raise TransportError: On all transport errors.

exception designate.backend.impl_akamai.**Forbidden**(**args*, ***kwargs*)

Bases: designate.exceptions.Forbidden, *designate.backend.impl_akamai.EnhancedDNSException*

Raised when an attempt to modify a zone which is registered to another Akamai account is made.

This appears to be returned when creating a new subzone of zone which already exists in another Akamai account.

designate.backend.impl_akamai.**build_zone**(*client*, *target*, *zone*)

Backend Bind9

Bind 9 backend. Create and delete zones by executing rndc

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

Bases: *designate.backend.base.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*)

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: *designate.backend.base.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*)

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*, *endpoint='https://api.dynect.net:443'*, *api_version='3.5.6'*, *headers=None*, *verify=True*, *retries=1*, *timeout=10*, *timings=False*, *pool_maxsize=10*, *pool_connections=10*)

Bases: object

DynECT service client.

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

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

get(**args*, ***kwargs*)

get_timings()

login()

logout()

patch(**args*, ***kwargs*)

poll_response(*response*)

The API might return a job nr in the response in case of a async response: https://github.com/fog/fog/issues/575

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

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

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

reset_timings()

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

Bases: *designate.backend.impl_dynect.DynClientError*

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

Bases: designate.exceptions.Backend

The base exception class for all HTTP exceptions.

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

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

Bases: designate.exceptions.BadRequest, *designate.backend.impl_dynect.DynClientError*

error_type = 'operation_blocked'

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

Bases: *designate.backend.base.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*)

Delete a DNS zone.

Parameters

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

get_client()

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

Bases: designate.exceptions.Backend

A job timedout.

error_code = 408

error_type = 'dyn_timeout'

Backend Infoblox

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

Bases: *designate.backend.base.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*)

Delete a DNS zone.

Parameters

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

ping(*context*)

Ping the Backend service

Backend Nsd4

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

Bases: *designate.backend.base.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*)

Delete a DNS zone.

Parameters

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

Backend Fake

class designate.backend.impl_fake.**FakeBackend**(*target*)
Bases: *designate.backend.base.Backend*

create_zone(*context*, *zone*)
Create a DNS zone.

Parameters

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

delete_zone(*context*, *zone*)
Delete a DNS zone.

Parameters

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

Backend PowerDNS 4

class designate.backend.impl_pdns4.**PDNS4Backend**(*target*)
Bases: *designate.backend.base.Backend*

create_zone(*context*, *zone*)
Create a DNS zone

delete_zone(*context*, *zone*)
Delete a DNS zone

Agent Backend KnotDNS

backend.agent_backend.impl_knot2

Knot DNS agent backend

Create, update, delete zones locally on a Knot DNS resolver using the knotc utility.

Supported Knot versions: >= 2.1, < 3

Knot DNS 2 User documentation

Warning: Untested, do not use in production.

Note: If the backend is killed during a configuration transaction it might be required to manually abort the transaction with *sudo knotc conf-abort*

Configured in [service:agent:knot2]

class designate.backend.agent_backend.impl_knot2.**Knot2Backend**(**a*, ***kw*)

Bases: designate.backend.agent_backend.base.AgentBackend

__abstractmethods__ = frozenset({})

__backend_status__ = 'untested'

__init__(**a*, ***kw*)
Configure the backend

__module__ = 'designate.backend.agent_backend.impl_knot2'

__plugin_name__ = 'knot2'

_abc_impl = <_abc_data object>

_execute_knotc(**knotc_args*, ***kw*)
Run the Knot client and check the output

Parameters

- **expected_output** (*str*) expected output (default: OK)
- **expected_error** (*str*) expected alternative output, will be logged as info(). Default: not set.

_lock_name = 'knot2.lock'

_modify_zone(**knotc_args*, ***kw*)
Create or delete a zone while locking, and within a Knot transaction. Knot supports only one config transaction at a time.

Raises exceptions.Backend

_start_minidns_to_knot_axfr(*zone_name*)
Instruct Knot to request an AXFR from MiniDNS. No need to lock or enter a configuration transaction.

create_zone(*zone*)
Create a new Zone by executing knotc Do not raise exceptions if the zone already exists.

Parameters **zone** (*raw pythondns Zone*) zone to be created

delete_zone(*zone_name*)
Delete a new Zone by executing knotc Do not raise exceptions if the zone does not exist.

Parameters **zone_name** (*str*) zone name

find_zone_serial(*zone_name*)
Get serial from a zone by running knotc

Returns serial (int or None)

Raises exceptions.Backend

start()
Start the backend

update_zone(*zone*)
Instruct Knot DNS to perform AXFR from MiniDNS

Parameters **zone** (*raw pythondns Zone*) zone to be created

Agent Backend gdnsd

backend.agent_backend.impl_gdnsd

gdnsd agent backend

Create, update, delete zones locally on a gdnsd resolver using the gdnsd utility.

Supported Knot versions: >= 2.1, < 3

User documentation

Warning: Untested, do not use in production.

Note: If the backend is killed during a configuration transaction it might be required to manually abort the transaction with *sudo gdnsd conf-abort*

Configured in [service:agent:gdnsd]

class designate.backend.agent_backend.impl_gdnsd.**GdnsdBackend**(**a*, ***kw*)

Bases: designate.backend.agent_backend.base.AgentBackend

__abstractmethods__ = frozenset({})

__backend_status__ = 'experimental'

__init__(**a*, ***kw*)
Configure the backend

__module__ = 'designate.backend.agent_backend.impl_gdnsd'

__plugin_name__ = 'gdnsd'

_abc_impl = <_abc_data object>

_check_conf()
Run gdnsd to check its configuration

_check_dirs(**dirnames*)
Check if directories are writable

_generate_zone_filename(*zone_name*)
Generate a filename for a zone file / is traslated into @ Non-valid characters are translated into NNN where NNN is a decimal integer in the range 0 - 255 The filename is lowercase

Returns valid filename (string)

_write_zone_file(*zone*)
Create or update a zone file atomically. The zone file is written to a unique temp file and then renamed

create_zone(***kw*)
Create a DNS zone

delete_zone(***kw*)
Delete a DNS zone

find_zone_serial(*zone_name*)
Query the local resolver for a zone Times out after SOA_QUERY_TIMEOUT

start()
Start the backend, check gdnsd configuration

Raises exception.Backend on invalid configuration

update_zone(***kw*)
Update a DNS zone

designate.backend.agent_backend.impl_gdnsd.**filter_exceptions**(*fn*)

Agent Backend Djbdns

backend.agent_backend.impl_djbdns

Djbdns DNS agent backend

Create, update, delete zones locally on a Djbdns DNS resolver using the axfr-get utility.

Djbdns User documentation

> **Warning:** Untested, do not use in production.

Configured in [service:agent:djbdns]

Requires rootwrap (or equivalent sudo privileges) to execute:

- tcpclient
- axfr-get
- tinydns-data

class designate.backend.agent_backend.impl_djbdns.**DjbdnsBackend**(**a*, ***kw*)

Bases: designate.backend.agent_backend.base.AgentBackend

__abstractmethods__ = frozenset({})

__backend_status__ = 'experimental'

__init__(**a*, ***kw*)
Configure the backend

__module__ = 'designate.backend.agent_backend.impl_djbdns'

__plugin_name__ = 'djbdns'

_abc_impl = <_abc_data object>

static _check_dirs(**dirnames*)
Check if directories are writable

static _concatenate_zone_datafiles(*data_fn*, *path_glob*)
Concatenate all zone datafiles into data

_perform_axfr_from_minidns(*zone_name*)
Instruct axfr-get to request an AXFR from MiniDNS.

Raises exceptions.Backend on error

_rebuild_data_cdb()
Rebuild data.cdb file from zone datafiles Requires global lock

On zone creation, axfr-get creates datafiles atomically by doing rename. On zone deletion, os.remove deletes the file atomically Globbing and reading the datafiles can be done without locking on them. The data and data.cdb files are written into a unique temp directory

create_zone(***kw*)
Create a DNS zone

delete_zone(***kw*)
Delete a DNS zone

find_zone_serial(*zone_name*)
Query the local resolver for a zone Times out after SOA_QUERY_TIMEOUT

start()
Start the backend

update_zone(***kw*)
Update a DNS zone

designate.backend.agent_backend.impl_djbdns.**filter_exceptions**(*fn*)

Agent Backend MSDNS

class designate.backend.agent_backend.impl_msdns.**MSDNSBackend**(*agent_service*)
Bases: designate.backend.agent_backend.base.AgentBackend

__abstractmethods__ = frozenset({})

__backend_status__ = 'experimental'

__init__(*agent_service*)
Configure the backend

__module__ = 'designate.backend.agent_backend.impl_msdns'

__plugin_name__ = 'msdns'

_abc_impl = <_abc_data object>

create_zone(*zone*)
Create a new DNS Zone

delete_zone(*zone_name*)
Delete a DNS Zone Do not raise exception if the zone does not exist.

find_zone_serial(*zone_name*)
Return the zones serial

start()
Start the backend

update_zone(*zone*)
Instruct MSDNS to request an AXFR from MiniDNS.

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

LOGGING_BLACKLIST = ['update_service_status']

RPC_API_VERSION = '6.2'

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_pool(*context*, *pool*)

create_record(*context*, *zone_id*, *recordset_id*, *record*, *increment_serial=True*)

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_pool(*context*, *pool_id*)

delete_record(*context*, *zone_id*, *recordset_id*, *record_id*, *increment_serial=True*)

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_accept(*context*, *zone_transfer_accept_id*)

delete_zone_transfer_request(*context*, *zone_transfer_request_id*)

export_zone(*context*, *zone_id*)

find_blacklist(*context*, *criterion*)

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_record(*context*, *criterion=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*)

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(*context*, *criterion=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_accept(*context*, *zone_transfer_accept*)

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

find_zone_transfer_request(*context*, *zone_transfer_request*)

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_quota(*context*, *tenant_id*, *resource*)

get_quotas(*context*, *tenant_id*)

get_record(*context*, *zone_id*, *recordset_id*, *record_id*)

get_recordset(*context*, *zone_id*, *recordset_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*)

list_floatingips(*context*)

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

reset_quotas(*context*, *tenant_id*)

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

sync_record(*context*, *zone_id*, *recordset_id*, *record_id*)

sync_zone(*context*, *zone_id*)

sync_zones(*context*)

touch_zone(*context*, *zone_id*)

update_blacklist(*context*, *blacklist*)

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

update_pool(*context*, *pool*)

update_record(*context*, *record*, *increment_serial=True*)

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

update_service_status(*context*, *service_status*)

update_status(*context*, *zone_id*, *status*, *serial*)

update_tld(*context*, *tld*)

update_tsigkey(*context*, *tsigkey*)

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

update_zone_export(*context*, *zone_export*)

update_zone_import(*context*, *zone_import*)

update_zone_transfer_accept(*context*, *zone_transfer_accept*)

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: designate.service.RPCService

RPC_API_VERSION = '6.2'

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_pool(*context*, *pool*)

create_record(*context*, *zone_id*, *recordset_id*, *record*, *increment_serial=True*)

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_pool(*context*, *pool_id*)

delete_record(*context*, *zone_id*, *recordset_id*, *record_id*, *increment_serial=True*)

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_accept(*context*, *zone_transfer_accept_id*)

delete_zone_transfer_request(*context*, *zone_transfer_request_id*)

export_zone(*context*, *zone_id*)

find_blacklist(*context*, *criterion*)

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_record(*context*, *criterion=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_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(*context*, *criterion=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_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*)
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_quota(*context*, *tenant_id*, *resource*)

get_quotas(*context*, *tenant_id*)

get_record(*context*, *zone_id*, *recordset_id*, *record_id*)

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

get_tenant(*context*, *tenant_id*)

get_tld(*context*, *tld_id*)

get_tsigkey(*context*, *tsigkey_id*)

get_zone(*context*, *zone_id*)
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*)

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.

property mdns_api

ping(*context*)

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*)

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

sync_record(*context*, *zone_id*, *recordset_id*, *record_id*)

sync_zone(*context*, *zone_id*)

sync_zones(*context*)

target = <Target version=6.2>

touch_zone(*context*, *zone_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_record(*context*, *record*, *increment_serial=True*)

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

update_service_status(*context*, *service_status*)

update_status(*context*, *zone_id*, *status*, *serial*)

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.

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_accept(*context*, *zone_transfer_accept*)

update_zone_transfer_request(*context*, *zone_transfer_request*)

property worker_api

xfr_zone(*context*, *zone_id*)

property zone_api

designate.central.service.**notification**(*notification_type*)

designate.central.service.**synchronized_zone**(*zone_arg=1*, *new_zone=False*)
Ensures only a single operation is in progress for each zone

A Decorator which ensures only a single operation can be happening on a single zone at once, within the current designate-central instance

MDNS

MDNS Base

class designate.mdns.base.**BaseEndpoint**(*tg*)
Bases: object

RPC_API_NAMESPACE = None

RPC_API_VERSION = None

property central_api

MDNS Handler

class designate.mdns.handler.**RequestHandler**(*storage*, *tg*)
Bases: *designate.mdns.xfr.XFRMixin*

property central_api

MDNS Notify

class designate.mdns.notify.**NotifyEndpoint**(*tg*)

Bases: *designate.mdns.base.BaseEndpoint*

RPC_API_NAMESPACE = 'notify'

RPC_API_VERSION = '2.0'

get_serial_number(*context*, *zone*, *host*, *port*, *timeout*, *retry_interval*, *max_retries*, *delay*)

Get zone serial number from a resolver using retries.

Parameters

- **context** The user context.
- **zone** The designate zone object. This contains the zone name. zone.serial = expected_serial
- **host** A notify is sent to this host.
- **port** A notify is sent to this port.
- **timeout** The time (in seconds) to wait for a SOA response from nameserver.
- **retry_interval** The time (in seconds) between retries.
- **max_retries** The maximum number of retries mindns would do for an expected serial number. After this many retries, mindns returns an ERROR.
- **delay** The time to wait before sending the first request.

Returns a tuple of (status, actual_serial, retries) status is either SUCCESS or ERROR. actual_serial is either the serial number returned in the SOA message from the nameserver or None. retries is the number of retries left. The return value is just used for testing and not by pool manager. The pool manager is informed of the status with update_status.

notify_zone_changed(*context*, *zone*, *host*, *port*, *timeout*, *retry_interval*, *max_retries*, *delay*)

Parameters

- **context** The user context.
- **zone** The designate zone object. This contains the zone name.
- **host** A notify is sent to this host.
- **port** A notify is sent to this port.
- **timeout** The time (in seconds) to wait for a NOTIFY response from server.
- **retry_interval** The time (in seconds) between retries.
- **max_retries** The maximum number of retries mindns would do for sending a NOTIFY message. After this many retries, mindns gives up.
- **delay** The time to wait before sending the first NOTIFY request.

Returns a tuple of (response, current_retry) where response is the response on success or None on failure. current_retry is the current retry number. The return value is just used for testing and not by pool manager.

poll_for_serial_number(*context*, *zone*, *nameserver*, *timeout*, *retry_interval*, *max_retries*, *delay*)

MDNS RPC API

class designate.mdns.rpcapi.**MdnsAPI**(*topic=None*)

Bases: object

Client side of the mdns RPC API.

Notify API version history:

1.0 - Added notify_zone_changed and poll_for_serial_number. 1.1 - Added get_serial_number. 2.0 - Changed method signatures

XFR API version history: 1.0 - Added perform_zone_xfr.

RPC_NOTIFY_API_VERSION = '2.0'

RPC_XFR_API_VERSION = '1.0'

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_serial_number(*context*, *zone*, *host*, *port*, *timeout*, *retry_interval*, *max_retries*, *delay*)

notify_zone_changed(*context*, *zone*, *host*, *port*, *timeout*, *retry_interval*, *max_retries*, *delay*)

perform_zone_xfr(*context*, *zone*)

poll_for_serial_number(*context*, *zone*, *nameserver*, *timeout*, *retry_interval*, *max_retries*, *delay*)

designate.mdns.rpcapi.**reset**()

MDNS Service

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

Bases: designate.service.RPCService

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

MDNS XFR

class designate.mdns.xfr.**XFRMixin**
Bases: object

Utility mixin that holds common methods for XFR functionality.

zone_sync(*context*, *zone*, *servers=None*)

class designate.mdns.xfr.**XfrEndpoint**(*tg*)
Bases: *designate.mdns.base.BaseEndpoint*, *designate.mdns.xfr.XFRMixin*

RPC_API_NAMESPACE = 'xfr'

RPC_API_VERSION = '1.0'

perform_zone_xfr(*context*, *zone*)

Objects

Objects Base

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

property is_valid
Returns True if the Object is valid.

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_context

property obj_fields

obj_get_original_value(*field*)
Returns the original value of a field.

classmethod obj_get_schema()

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.

save(*context*)
Save the changed fields back to the store.

This is optional for subclasses, but is presented here in the base class for consistency among those that do.

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: oslo_versionedobjects.base.VersionedObjectRegistry

registration_hook(*cls*, *index*)

class designate.objects.base.**ListObjectMixin**(**args*, ***kwargs*)
Bases: oslo_versionedobjects.base.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.fiel

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.fiel

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.Un

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

Objects Backlist

class designate.objects.blacklist.**Blacklist**(**args*, ***kwargs*)
Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

STRING_KEYS = ['id', 'pattern']

property created_at

property description

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fiel

property id

property pattern

property updated_at

property version

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

Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE

alias of *Blacklist*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects Zone

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

Bases: *designate.objects.base.DesignateObject*, oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.SoftDeleteObjectMixin*

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

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.Unspe

get_master_by_ip(*host*)

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

property id

property masters

property minimum

property name

property parent_zone_id

property pool_id

property recordsets

property refresh

property retry

property serial

property shard

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: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*, *designate.objects.base.PagedListObjectMixin*

LIST_ITEM_TYPE

alias of *Zone*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

property total_count

Objects Pool

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

Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

STRING_KEYS = ['id', 'name']

property also_notifies

property attributes

property created_at

property description

fields = {'also_notifies': Object(default=<class 'oslo_versionedobjects.fie

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: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE

alias of *Pool*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects Quota

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

Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

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

property created_at

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fiel

property hard_limit

property id

property resource

property tenant_id

property updated_at

property version

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

Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE

alias of *Quota*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

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: *designate.objects.base.DesignateObject*, *designate.objects.base.PersistentObjectMixin*, oslo_versionedobjects.base.VersionedObjectDictCompat

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

property action

property created_at

property data

property description

fields = {'action': Enum(default=<class 'oslo_versionedobjects.fields.Unspe

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: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE

alias of *Record*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects Recordset

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

Bases: *designate.objects.base.DesignateObject*, oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*

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

property action

property created_at

property description

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fiel

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: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*, *designate.objects.base.PagedListObjectMixin*

LIST_ITEM_TYPE

alias of *RecordSet*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

property total_count

Objects Server

class designate.objects.server.**Server**(**args*, ***kwargs*)
Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

STRING_KEYS = ['id', 'name']

property created_at

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fiel

property id

property name

property updated_at

property version

class designate.objects.server.**ServerList**(**args*, ***kwargs*)
Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE
alias of *Server*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects Tenant

class designate.objects.tenant.**Tenant**(**args*, ***kwargs*)
Bases: *designate.objects.base.DesignateObject*, oslo_versionedobjects.base.VersionedObjectDictCompat

STRING_KEYS = ['id']

fields = {'id': Any(default=<class 'oslo_versionedobjects.fields.Unspecifie

property id

property zone_count

property zones

class designate.objects.tenant.**TenantList**(**args*, ***kwargs*)
Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE
alias of *Tenant*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects TLD

class designate.objects.tld.**Tld**(**args*, ***kwargs*)
Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

STRING_KEYS = ['id', 'name']

property created_at

property description

fields = {'created_at': DateTime(default=<class 'oslo_versionedobjects.fiel

property id

property name

property updated_at

property version

class designate.objects.tld.**TldList**(**args*, ***kwargs*)
Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE
alias of *Tld*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects TSigKey

class designate.objects.tsigkey.**TsigKey**(**args*, ***kwargs*)
Bases: oslo_versionedobjects.base.VersionedObjectDictCompat, *designate.objects.base.PersistentObjectMixin*, *designate.objects.base.DesignateObject*

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

property algorithm

property created_at

fields = {'algorithm': Enum(default=<class 'oslo_versionedobjects.fields.Un

property id

property name

property resource_id

property scope

property secret

property updated_at

property version

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

Bases: *designate.objects.base.ListObjectMixin*, *designate.objects.base.DesignateObject*

LIST_ITEM_TYPE

alias of *TsigKey*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects A Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *A*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects AAAA Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *AAAA*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects CNAME Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *CNAME*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects MX Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *MX*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects NS Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *NS*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects PTR Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *PTR*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects SOA Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *SOA*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects SPF Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *SPF*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects SRV Record

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

Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *SRV*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects TXT Record

class designate.objects.rrdata_txt.**TXT**(**args*, ***kwargs*)
Bases: *designate.objects.record.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.Unspe

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE
alias of *TXT*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects SSHFP Record

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

Bases: *designate.objects.record.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.Unspe

property fingerprint

property fp_type

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE

alias of *SSHFP*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects NAPTR Record

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

Bases: *designate.objects.record.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.Unspe

property flags

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: *designate.objects.record.RecordList*

LIST_ITEM_TYPE
alias of *NAPTR*

fields = {'objects': List(default=<class 'oslo_versionedobjects.fields.Unsp

property objects

Objects CAA Record

members

undoc-members

show-inheritance

Quota

Quota Base

class designate.quota.base.**Quota**
Bases: designate.plugin.DriverPlugin

Base class for quota plugins

get_default_quotas(*context*)

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

get_quotas(*context*, *tenant_id*)

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

reset_quotas(*context*, *tenant_id*)

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

Quota Storage

class designate.quota.impl_storage.**StorageQuota**
Bases: *designate.quota.base.Quota*

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

reset_quotas(*context*, *tenant_id*)

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

Sink

Sink Service

class designate.sink.service.**Service**
Bases: designate.service.Service

info(*context*, *publisher_id*, *event_type*, *payload*, *metadata*)
Processes an incoming notification, offering each extension the opportunity to handle it.

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.base.**Storage**
Bases: designate.plugin.DriverPlugin

Base class for storage plugins

abstract count_records(*context*, *criterion=None*)
Count records

Parameters

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

abstract count_recordsets(*context*, *criterion=None*)
Count recordsets

Parameters

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

abstract count_tenants(*context*)
Count tenants

Parameters **context** RPC Context.

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

Parameters

- **context** RPC Context.

- **criterion** Criteria to filter by.

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

Parameters

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

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

Parameters

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

abstract 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.

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

Parameters

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

abstract 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.

abstract 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.

abstract create_tld(*context*, *tld*)
Create a TLD.

Parameters

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

abstract create_tsigkey(*context*, *tsigkey*)
Create a TSIG Key.

Parameters

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

abstract create_zone(*context*, *zone*)
Create a new Zone.

Parameters

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

abstract 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.

abstract 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.

abstract delete_blacklist(*context*, *blacklist_id*)
Delete a Blacklist via ID.

Parameters

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

abstract 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

abstract 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

abstract delete_quota(*context*, *quota_id*)
Delete a Quota via ID.

Parameters

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

abstract delete_record(*context*, *record_id*)
Delete a record

Parameters

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

abstract delete_recordset(*context*, *recordset_id*)
Delete a recordset

Parameters

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

abstract delete_tld(*context*, *tld_id*)
Delete a TLD via ID.

Parameters

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

abstract delete_tsigkey(*context*, *tsigkey_id*)
Delete a TSIG Key via ID.

Parameters

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

abstract delete_zone(*context*, *zone_id*)
Delete a Zone

Parameters

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

abstract 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

abstract 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

abstract find_blacklist(*context*, *criterion*)
Find a single Blacklist.

Parameters

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

abstract 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.

abstract find_pool(*context*, *criterion*)
Find a single Pool.

Parameters

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

abstract find_pool_attribute(*context*, *criterion*)
Find a single PoolAttribute

Parameters

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

abstract 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

abstract 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

abstract find_quota(*context*, *criterion*)

Find a single Quota.

Parameters

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

abstract 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.

abstract find_record(*context*, *criterion*)

Find a single Record.

Parameters

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

abstract 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.

abstract find_recordset(*context*, *criterion*)
Find a single RecordSet.

Parameters

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

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

abstract find_recordsets_axfr(*context*, *criterion=None*)
Find RecordSets.

Parameters

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

abstract find_service_status(*context*, *criterion*)
Find a single Service Status.

Parameters

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

abstract 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.

abstract find_tenants(*context*)
Find all Tenants.

Parameters **context** RPC Context.

abstract find_tld(*context*, *criterion*)
Find a single TLD.

Parameters

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

abstract 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.

abstract 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.

abstract find_zone(*context*, *criterion*)
Find a single Zone.

Parameters

- **context** RPC Context.

- **criterion** Criteria to filter by.

abstract find_zone_export(*context*, *criterion*)
Find a single Zone Export.

Parameters

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

abstract 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.

abstract find_zone_import(*context*, *criterion*)
Find a single Zone Import.

Parameters

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

abstract 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.

abstract 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.

abstract get_blacklist(*context*, *blacklist_id*)
Get a Blacklist via ID.

Parameters

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

abstract get_pool(*context*, *pool_id*)
Get a Pool via the id

Parameters

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

abstract 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

abstract get_quota(*context*, *quota_id*)
Get a Quota via ID.

Parameters

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

abstract get_record(*context*, *record_id*)
Get a record via ID

Parameters

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

abstract get_recordset(*context*, *recordset_id*)
Get a recordset via ID

Parameters

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

abstract get_tenant(*context*, *tenant_id*)
Get all Tenants.

Parameters

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

abstract get_tld(*context*, *tld_id*)
Get a TLD via ID.

Parameters

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

abstract get_tsigkey(*context*, *tsigkey_id*)
Get a TSIG Key via ID.

Parameters

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

abstract get_zone(*context*, *zone_id*)
Get a Zone via its ID.

Parameters

- **context** RPC Context.
- **zone_id** ID of the Zone.

abstract 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.

abstract 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.

ping(*context*)
Ping the Storage connection

abstract purge_zone(*context*, *zone*)
Purge a Zone

Parameters

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

abstract update_blacklist(*context*, *blacklist*)
Update a Blacklist

Parameters

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

abstract update_pool(*context*, *pool*)
Update the specified pool

Parameters

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

abstract update_pool_attribute(*context*, *pool_attribute*)
Update the specified pool

Parameters

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

abstract update_quota(*context*, *quota*)
Update a Quota

Parameters

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

abstract update_record(*context*, *record*)
Update a record

Parameters

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

abstract update_recordset(*context*, *recordset*)
Update a recordset

Parameters

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

abstract 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.

abstract update_tld(*context*, *tld*)
Update a TLD

Parameters

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

abstract update_tsigkey(*context*, *tsigkey*)
Update a TSIG Key

Parameters

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

abstract update_zone(*context*, *zone*)
Update a Zone

Parameters

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

abstract update_zone_export(*context*, *zone_export*)
Update a Zone Export

Parameters

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

abstract update_zone_import(*context*, *zone_import*)
Update a Zone Import

Parameters

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

1.2.6 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.
```

(continues on next page)

(continued from previous page)

```
  # 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.
  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.

1.2.7 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
# This is optional - if these credentials are not provided designate will
# use the users context and auth token to query neutron
#admin_username = designate
#admin_password = designate
#admin_tenant_name = designate
auth_url = http://localhost:35357/v2.0
insecure = False
auth_strategy = keystone
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.

1.2.8 Other modules

1.3 User guide

In this section, you will find documentation relevant for using Designate.

Contents:

1.3.1 Deprecated REST API Documentation

Intro

In the REST API examples, HTTP requests are defined as raw HTTP. For example:

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
     "name": "example.org.",
     "email": "hostmaster@example.org"
}
```

With this info we can make this request using the cURL tool. Well assume we are running Designate on *localhost*.

```
curl -X POST -i \
     -H 'Accept: application/json' \
     -H 'Content-Type: application/json' \
     -d '{"name": "example.org.", "email": "hostmaster@example.org"}' \
     http://localhost:9001/v2/zones
```

The *-i* flag is used to dump the response headers as well as the response body.

The cURL tool is extremely robust. Please take a look at the cURL tutorial for more info.

HTTP Headers

These headers work for all APIs

- **X-Designate-Edit-Managed-Records**
  - Allows admins (or users with the right role) to modify managed records (records created by designate-sink / reverse floating ip API)
- **X-Auth-All-Projects**
  - Allows admins (or users with the right role) to view and edit zones / recordsets for all tenants
- **X-Auth-Sudo-Tenant-ID / X-Auth-Sudo-Project-ID**
  - Allows admins (or users with the right role) to impersonate another tenant specified by this header

API Versions

V2 API

The V2 API is documented on the OpenStack Developer api site

Admin API

Quotas

Overview

The quotas extension can be used to retrieve a tenants absolute limits.

Note: Quotas is an extension and needs to be enabled before it can be used. If Designate returns a 404 error, ensure that the following line has been added to the designate.conf file under `[service:api]` section

```
enable_api_admin = True
enabled_extensions_admin = quotas
```

Once these lines have been added, restart the designate-central and designate-api services.

Get Quotas

GET /quotas/TENANT_ID

Retrieves quotas for tenant with the specified TENANT_ID. The following example retrieves the quotas for tenant 12345.

Example request:

```
GET /admin/quotas/12345 HTTP/1.1
Host: 127.0.0.1:9001
Accept: application/json
Content-Type: application/json
```

Example response:

```
HTTP/1.1 201 Created
Content-Type: application/json

{
  "quota": {
    "api_export_size: 1000,
    "zones": 10,
    "recordset_records": 20,
    "zone_records": 500,
    "zone_recordsets": 500
  }
}
```

Api_export_size Number of records allowed in a synchronous zone export done via API

Form Parameters

- **zones** Number of zones the tenant is allowed to own
- **recordset_records** Number of records allowed per recordset
- **zone_records** Number of records allowed per zone
- **zone_recordsets** Number of recordsets allowed per zone

Status Codes

- 200 OK Success
- 401 Unauthorized Access Denied

Update Quotas

PATCH /quotas/TENANT_ID

Updates the specified quota(s) to their new values. Negative quota values mean unlimited.

Example request:

```
PATCH /admin/quotas/12345 HTTP/1.1
Host: 127.0.0.1:9001
Accept: application/json
Content-Type: application/json

{
  "quota": {
    "zones": 1000,
    "zone_records": 50
  }
}
```

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "quota": {
    "api_export_size: 1000,
    "zones": 1000,
    "recordset_records": 20,
    "zone_records": 50,
    "zone_recordsets": 500
  }
}
```

Status Codes

- 200 OK Success
- 401 Unauthorized Access Denied

Reset Quotas to Default

DELETE /quotas/TENANT_ID

Restores the tenants quotas back to their default values.

Example request:

```
DELETE /admin/quotas/12345 HTTP/1.1
Host: 127.0.0.1:9001
Accept: application/json
Content-Type: application/json
```

Example response:

```
HTTP/1.1 204 No Content
```

Status Codes

- 204 No Content No Content
- 401 Unauthorized Access Denied

1.3.2 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 domain. 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, there are two requirements.

1. A domain that should be pointed to from the IP
2. A *in-addr.arpa.* zone entry that will receive the actual *PTR* record

Using the V2 API

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/fe078042-0aa3-4500-a81e-
↪8f328f79bf75
Content-Length: 476
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-bfcd0723-624c-4ec2-bbd5-99e985efe8db
Date: Fri, 20 Feb 2015 21:20:28 GMT
Connection: keep-alive

{
  "email": "admin@example.org",
  "project_id": "noauth-project",
  "action": "CREATE",
  "version": 1,
  "pool_id": "794ccc2c-d751-44fe-b57f-8894c9f5c842",
  "created_at": "2015-02-20T21:20:28.000000",
  "name": "example.org.",
  "id": "fe078042-0aa3-4500-a81e-8f328f79bf75",
  "serial": 1424467228,
  "ttl": 3600,
  "updated_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/fe078042-0aa3-4500-a81e-
↪8f328f79bf75"
  },
  "description": "A great example zone",
  "status": "PENDING"
}
```

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 wed like to use for our reverse DNS lookup, we need to add an *in-addr.arpa.* zone that includes the IP address well be looking 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/1bed5d24-d487-4410-b813-
↪f1c637db0ba3
Content-Length: 512
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-4e691123-045e-4f8e-ae50-b5eabb5af3fa
Date: Fri, 20 Feb 2015 21:35:41 GMT
Connection: keep-alive

{
  "email": "admin@example.org",
  "project_id": "noauth-project",
  "action": "CREATE",
  "version": 1,
  "pool_id": "794ccc2c-d751-44fe-b57f-8894c9f5c842",
  "created_at": "2015-02-20T21:35:41.000000",
  "name": "11.2.0.192.in-addr.arpa.",
  "id": "1bed5d24-d487-4410-b813-f1c637db0ba3",
  "serial": 1424468141,
  "ttl": 3600,
  "updated_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/1bed5d24-d487-4410-b813-
↪f1c637db0ba3"
  },
  "description": "A in-addr.arpa. zone for reverse lookups.",
  "status": "PENDING"
}
```

Now that we have our *in-addr.arpa.* zone, we add a new *PTR* record to the zone.

```
POST /v2/zones/1bed5d24-d487-4410-b813-f1c637db0ba3/recordsets HTTP/1.1
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
Accept: application/json

{
  "name": "11.2.0.192.in-addr.arpa.",
  "description": "A PTR recordset",
  "type": "PTR",
  "ttl": 3600,
  "records": [
    "example.org."
  ]
}
```

Here is the response.

```
HTTP/1.1 202 Accepted
Location: http://127.0.0.1:9001/v2/zones/1bed5d24-d487-4410-b813-
↪f1c637db0ba3/recordsets/a3dca24e-3eba-4523-8607-c0ad4b9a9272
Content-Length: 499
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-5b7044d0-591a-445a-839f-1403b1455824
Date: Fri, 20 Feb 2015 21:42:45 GMT
Connection: keep-alive

{
  "type": "PTR",
  "action": "CREATE",
  "version": 1,
  "created_at": "2015-02-20T21:42:45.000000",
  "zone_id": "1bed5d24-d487-4410-b813-f1c637db0ba3",
  "name": "11.2.0.192.in-addr.arpa.",
  "id": "a3dca24e-3eba-4523-8607-c0ad4b9a9272",
  "ttl": 3600,
  "records": [
    "example.org."
  ],
  "updated_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/zones/1bed5d24-d487-4410-b813-
↪f1c637db0ba3/recordsets/a3dca24e-3eba-4523-8607-c0ad4b9a9272"
  },
  "description": "A PTR recordset",
  "status": "PENDING"
}
```

We should now have a correct *PTR* record assigned in our nameserver that we can test.

Note: As the *in-addr.arpa.* zone is considered an admin zone, you may need to get admin rights in order to create the necessary subdomains.

Lets test it out!

```
$ dig @localhost -x 192.0.2.11

; <<>> DiG 9.9.5-3ubuntu0.1-Ubuntu <<>> @localhost -x 192.0.2.11
```

(continues on next page)

(continued from previous page)

```
; (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:
;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.

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.

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "2.0.192.in-addr.arpa.",
  "ttl": 3600,
  "email": "admin@example.com"
}
```

We then could use the corresponding domain to create a *PTR* record for a specific IP.

```
POST /v2/zones/$domain_uuid/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."
  ]
}
```

When we do our reverse look, we should see *cats.example.com.*

```
$ dig @localhost -x 192.0.2.3 +short
cats.example.com.
```

Success!

You can further specify *in-addr.arpa.* zones to chunks of IP addresses by using Classless in-addr.arpa. Delegation. See RFC 2317 for more information.

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 is ends up as *12.2.0.192.in-addr.arpa.*. In Designate, the name of a record MUST be a complete host name.

1.3.3 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"
                SPF             "v=spf1 +a +mx ~all"
```

(continues on next page)

(continued from previous page)

```
                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.
```

1.4 Administration guide

In this section, you will find documentation relevant for administering and operating Designate.

Contents:

1.4.1 Managing Top Level Domain Names

Designate allows management of the Top-Level Domains (TLDs) that users are allowed to create zones within.

For example, its simple to only allow users to create zones that end in `.com.` TLD.

By default, all TLDs are allowed in Designate, this is ok for most scenarios.

If for example you wanted to restrict to only `.com.` though, you could make the following API call.

```
POST /v2/tlds HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "com"
}
```

Response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json
X-Openstack-Request-Id: req-432e72b4-f4e1-4f9c-8e35-53decc752260

{
  "id": "2f8bc76d-1701-4323-a101-248e09471342",
  "name": "com",
  "description": null,
  "created_at": "2020-06-01T16:25:44.000000",
  "updated_at": null,
  "links": {
    "self": "http://127.0.0.1:9001/v2/tlds/2f8bc76d-1701-4323-a101-
↪248e09471342"
  }
}
```

Using the command line client:

```
$ openstack tld create --name com
+-------------+--------------------------------------+
| Field       | Value                                |
+-------------+--------------------------------------+
| created_at  | 2020-06-01T16:25:44.000000           |
| description | None                                 |
```

(continues on next page)

(continued from previous page)

```
| id          | 2f8bc76d-1701-4323-a101-248e09471342 |
| name        | com                                  |
| updated_at  | None                                 |
+-------------+--------------------------------------+
```

Now, if someone were to try and create `example.net.`, they would encounter an error:

```
POST /v2/zones HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name": "example.net.",
  "type": "PRIMARY",
  "email": "admin@example.net"
}
```

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json
X-Openstack-Request-Id: req-3a8985fd-0155-4dd4-a7fb-584b140f1f59

{
  "code": 400,
  "type": "invalid_zone_name",
  "message": "Invalid TLD",
  "request_id": "req-3a8985fd-0155-4dd4-a7fb-584b140f1f59"
}
```

Using the command line client:

```
$ openstack zone create --email admin@example.net example.net.
Invalid TLD
```

TLDs can be deleted, just like many other resources in the API, using `DELETE /v2/tlds/<id>`:

```
DELETE /v2/tlds/2f8bc76d-1701-4323-a101-248e09471342 HTTP/1.1
Accept: application/json
Content-Type: application/json
```

Or by using the command line client:

```
$ openstack tld delete com
TLD com was deleted
```

1.4.2 DNS Server Plugin Documentation

Contents:

Agent Backend

This page documents using the various Agent backends, and its accompanying service, *designate-agent*. This backend uses an extension of the DNS protocol itself to send management requests to the remote agent processes, where the requests will be actioned.

The *rpc* traffic between designate and the *agent* is both unauthenticated and unencrypted. Do not run this traffic over unsecured networks.

Designate Configuration

For each designate-agent running, add a target to the pools.yaml configuration file, using the following template:

```
targets:
  - type: agent
    description: Agent Server 1

    # List out the designate-mdns servers from which Agent␣
↪servers should
    # request zone transfers (AXFRs) from.
    masters:
      - host: 192.0.2.1
        port: 5354

    # Agent Configuration options, this should be this targets
    # designate-agent service's host and port.
    options:
      host: 192.0.2.2
      port: 5358
```

Then update the designate pools database using the `designate-manage pool` command - see *designate-manage pool* for further details on the `designate-manage pool` command:

```
$ designate-manage pool update
```

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.
```

(continues on next page)

(continued from previous page)

```
  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

      # 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
      rndc_key_file: /etc/designate/rndc.key
```

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>";
};
```

Djbdns Agent backend

Djbdns User documentation

This page documents the Agent backend for djbdns.

The agent runs on the same host as the tinydns resolver. It receives DNS messages from Mini DNS using private DNS OPCODEs and classes and creates or deletes zones in the data.cdb file using axfr-get and tinydns-data

Setting up Djbdns on Ubuntu Trusty

Assuming no DNS resolver is already installed, run as root:

```
set -u
datadir=/var/lib/djbdns
ug_name=djbdns
tinydns_ipaddr=127.0.0.1

[[ -d $datadir ]] && echo "$datadir already exists" && exit 1
set -e
apt-get update
apt-get install dbndns daemontools
if ! getent passwd $ug_name >/dev/null; then
  adduser --quiet --system --group --no-create-home --home /nonexistent
↪$ug_name
fi
tinydns-conf $ug_name $ug_name $datadir $tinydns_ipaddr
cd $datadir/root
tinydns-data data
chown -Rv $ug_name:$ug_name $datadir
```

Setup the a Systemd service or, alternatively, an initfile to start TinyDNS.

In the contrib/djbdns directory there are example files for both.

```
systemctl daemon-reload
service tinydns start
service tinydns status
```

If needed, create the rootwrap filters, as root:

```
cat > /etc/designate/rootwrap.d/djbdns.filters <<EOF
# cmd-name: filter-name, raw-command, user, args
[Filters]
tcpclient: CommandFilter, /usr/bin/tcpclient, root
axfr-get: CommandFilter, /usr/bin/axfr-get, root
EOF

# Check the filter:
sudo /usr/local/bin/designate-rootwrap /etc/designate/rootwrap.conf␣
↪tcpclient -h
sudo /usr/local/bin/designate-rootwrap /etc/designate/rootwrap.conf axfr-
↪get -h
```

Configure the service.agent and backend.agent.djbdns sections in /etc/designate/designate.conf

Look in designate.conf.example for examples.

Create an agent pool:

```
# Fetch the existing pool(s) if needed or start from scratch
designate-manage pool generate_file --file /tmp/pool.yaml
# Edit the file (see below) and reload it as:
designate-manage pool update --file /tmp/pool.yaml
```

The targets section in pool.yaml should look like:

```
targets:
- description: gdnsd agent
  masters:
  - host: <MiniDNS IP addr>
    port: 5354
  options: {}
  options:
  - host: <Agent IP addr>
    port: 5358
  type: agent
```

Testing

Create new zones and records. Monitor the agent logfile and the contents of the TinyDNS datadir. The data.cdb file should be receiving updates.

```
openstack zone create --email example@example.org example.org.
openstack recordset create example.org. --type A foo --records 1.2.3.4
dig example.org @<tinydns_ipaddr> SOA
dig foo.example.org @<tinydns_ipaddr> A
```

Developer documentation

Devstack testbed

Follow Setting up Djbdns on Ubuntu Trusty

Configure Tinydns to do AXFR from MiniDNS on 192.168.121.131

gdnsd Agent backend

User documentation

This page documents the Agent backend for gdnsd.

The agent runs on the same host as the resolver. It receives DNS messages from Mini DNS using private DNS OPCODEs and classes and creates/updates/deletes zones on gdnsd using zone files under the gdnsd configuration directory.

The backend supports gdnsd from version 2.0

gdnsd documentation

Setting up gdnsd on Ubuntu Vivid

Run as root:

```
apt-get update
apt-get install gdnsd
```

Configuring gdnsd

Assuming gdnsd has been freshly installed on the system, run as root:

```
# Monitor syslog during the next steps
tail -f /var/log/syslog

# config check should be successful
/usr/sbin/gdnsd checkconf

# Start the daemon if needed
service gdnsd status
service gdnsd start

# gdnsd should be listening on TCP and UDP ports
netstat -lnptu | grep '/gdnsd'

# Test the daemon: it should respond with "gdnsd"
dig @127.0.0.1 CH TXT +short
```

Configure the service.agent and backend.agent.gdnsd sections in /etc/designate/designate.conf

Look in designate.conf.example for more complete examples

```
[service:agent]
backend_driver = gdnsd
# Place here the MiniDNS ipaddr and port (not the agent itself)
masters = 192.168.27.100:5354

[backend:agent:gdnsd]
#gdnsd_cmd_name = gdnsd
#confdir_path = /etc/gdnsd
#query_destination = 127.0.0.1
```

Ensure that the zones directory under confdir_path (default /etc/gdnsd) is readable and writable by the system user running the Designate Agent

Create an agent pool:

```
# Fetch the existing pool(s) if needed
designate-manage pool generate_file --file /tmp/pool.yaml
# Edit the file (see below) and reload it as:
designate-manage pool update --file /tmp/pool.yaml
```

The targets section in pool.yaml should look like:

```
targets:
- description: gdnsd agent
  masters:
  - host: <MiniDNS IP addr>
    port: 5354
  options: {}
  options:
  - host: <Agent IP addr>
    port: 5358
  type: agent
```

Start the Designate Agent. You should see log messages similar to:

```
2016-05-03 15:13:38.193 INFO designate.backend.agent_backend.impl_gdnsd [-
↪] gdnsd command: 'gdnsd'
2016-05-03 15:13:38.193 INFO designate.backend.agent_backend.impl_gdnsd [-
↪] gdnsd conf directory: '/etc/gdnsd'
2016-05-03 15:13:38.194 INFO designate.backend.agent_backend.impl_gdnsd [-
↪] Resolvers: ['127.0.0.1']
```

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.

Knot DNS 2 Agent backend

Knot DNS 2 User documentation

This page documents the Agent backend for Knot DNS.

The agent runs on the same host as the resolver. It receives DNS messages from Mini DNS using private DNS OPCODEs and classes and creates or deletes zones on Knot using the knotc tool. It also instructs Knot to request AXFR from MiniDNS when a zone is created or updated.

Support matrix:

- 2.0 and older: not supported
- 2.2.0: affected by a bug

Knot DNS documentation

Configuring Knot DNS

Assuming Knot has been freshly installed on the system, run as root:

```
# Monitor syslog during the next steps
tail -f /var/log/syslog

# Start the daemon, ensure it's running
service knot start
netstat -npltu | grep knotd

# Create the config database
knotc conf-init

# Edit /etc/default/knot
# Set the variable:
# KNOTD_ARGS="-C /var/lib/knot/confdb"

# Restart
service knot restart

# Check if the deamon is still running from the conf file in /etc/knot/
ps axuw | grep knotd

# if so, apply this workaround for bug
# https://gitlab.labs.nic.cz/labs/knot/issues/455
( cd /etc/default/ && ln -s knot knotd )
service knot restart
ps axuw | grep knotd

# Ensure the confdb is present
test -f /var/lib/knot/confdb/data.mdb && echo OK

# Create the configuration
# Populate the variable with the MiniDNS ipaddr:
MINIDNS_IPADDR=

knotc conf-begin
knotc conf-set server.listen 0.0.0.0@53
# To listen on IPv6 as well, also run this:
# knotc conf-set server.listen '::@53'
knotc conf-set remote[minidns]
knotc conf-set remote[minidns].address $MINIDNS_IPADDR@5354
knotc conf-set template[default]
knotc conf-set template[default].master minidns
knotc conf-set template[default].acl acl_minidns
knotc conf-set template[default].semantic-checks on
knotc conf-set zone[example.com]
knotc conf-set log.any info
knotc conf-set log.target syslog
```

(continues on next page)

(continued from previous page)

```
knotc conf-set acl[acl_minidns]
knotc conf-set acl[acl_minidns].address $MINIDNS_IPADDR
knotc conf-set acl[acl_minidns].action notify
# Review the changes and commit
knotc conf-diff
knotc conf-commit

# Optionally check and back up the conf
knotc conf-check
knotc conf-export knot.conf.bak && cat knot.conf.bak

# Ensure the zone survives a restart
service knot restart
knotc zone-status example.com

# Test Knot: this should return the version
dig @127.0.0.1 version.server CH TXT
```

If needed, create a rootwrap filter, as root:

```
cat > /etc/designate/rootwrap.d/knot2.filters <<EOF
# cmd-name: filter-name, raw-command, user, args
[Filters]
knotc: CommandFilter, /usr/sbin/knotc, root
EOF

# Check the filter:
sudo /usr/local/bin/designate-rootwrap /etc/designate/rootwrap.conf knotc␣
↪status
```

Configure the service.agent and backend.agent.knot2 sections in /etc/designate/designate.conf

Look in designate.conf.example for examples

Create an agent pool:

```
# Fetch the existing pool(s) if needed or start from scratch
designate-manage pool generate_file --file /tmp/pool.yaml
# Edit the file (see below) and reload it as:
designate-manage pool update --file /tmp/pool.yaml
```

The targets section in pool.yaml should look like:

```
targets:
- description: knot2 agent
  masters:
  - host: <MiniDNS IP addr>
    port: 5354
  options: {}
  options:
  - host: <Agent IP addr>
    port: 5358
  type: agent
```

Developer documentation

Devstack testbed

Follow Setting up Knot DNS on Ubuntu Trusty

Configure Knot to slave from MiniDNS on 192.168.121.131

Knotd configuration example (sudo knotc conf-export <filename>):

```
# Configuration export (Knot DNS 2.1.1)

server:
    listen: "0.0.0.0@53"

log:
- target: "syslog"
    any: "debug"

acl:
- id: "acl_minidns"
    address: [ "192.168.121.131" ]
    action: [ "notify" ]

remote:
- id: "minidns"
    address: "192.168.121.131@5354"

template:
- id: "default"
    master: "minidns"
    acl: "acl_minidns"
    semantic-checks: "on"
```

MSDNS Agent Backend

MSDNS User Documentation

This page documents using the MSDNS Agent backend.

The agent runs on the Windows host where the Microsoft DNS Server feature is installed. It receives DNS messages from Mini DNS using private DNS OPCODEs and classes and creates or deletes zones using WMI calls.

It also instructs MSDNS to request AXFR from MiniDNS when a zone is created or updated.

Microsoft DNS documentation for managing DNS zones

Setting up the Microsoft DNS server on Windows Server

The DNS Server role can be installed on the system by following the documentation available here: How to install the DNS Server role

Configuring MSDNS

Assuming the DNS Server role has been installed on the system, follow the next steps to complete the configuration.

These steps are for the Windows host which will run the designate agent. Make sure that Python 2.7 or Python 3.4 is installed on the system already.

To install Designate, clone the repository from https://github.com/openstack/designate and do a pip install. Example:

```
git clone https://github.com/openstack/designate
pip install .\\designate
```

After that, we need to configure the Designate Agent. Inside the github repository, there is a folder named etc/designate which can be used as default configuration.

Copy the folder somewhere else, for this example we will copy it to C:\etc\designate Inside the configuration folder, make a copy of designate.conf.sample and rename the copy to designate.conf Example:

```
copy C:\\etc\\designate\\designate.conf.sample C:\\etc\\designate\\
↪designate.conf
```

Configure the service.agent and backend.agent.msdns sections in C:\etc\designate\designate.conf

Look in C:\etc\designate\designate.conf.example for more complete examples.

```
[service:agent]
backend_driver = msdns
# Place here the MiniDNS ipaddr and port (no the agent itself)
masters = <MiniDNS IP addr>:53
```

Ensure that policy_file under the [default] section is set:

```
policy_file = C:\\etc\\designate\\policy.json
```

Start the designate agent using (Python 2.7 was installed in the default location C:\Python27):

```
C:\\Python27\\Scripts\\designate-agent.exe --config-file 'C:\\etc\\
↪designate\\designate.conf'
```

You should see log messages similar to:

```
2016-06-22 02:00:47.177 3436 INFO designate.backend.agent_backend.impl_
↪msdns [-] Started msdns backend
2016-06-22 02:00:47.177 3436 INFO designate.service [-] _handle_tcp thread␣
↪started
2016-06-22 02:00:47.177 3436 INFO designate.service [-] _handle_udp thread␣
↪started
```

The following steps are for the system running the Designate controller.

Make sure to set the mDNS port to 53 in the `[service:mdns]` section. MS DNS does not support Masters that are on any port other than 53.

Create an agent pool:

```
# Fetch the existing pool(s) if needed or start from scratch
designate-manage pool generate_file --file /tmp/pool.yaml
# Edit the file (see below) and reload it as:
designate-manage pool update --file /tmp/pool.yaml
```

The targets section in pool.yaml should look like:

```
targets:
- description: Microsoft DNS agent
  masters:
  - host: <MiniDNS IP addr>
    port: 53
  options: {}
  options:
  - host: <Agent IP addr>
    port: 5358
  type: agent
```

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
```

(continues on next page)

(continued from previous page)

```
      # PowerDNS Configuration options
      options:
        host: 192.0.2.1
        port: 53
        api_endpoint: http://127.0.0.1:8081
        api_token: changeme
        # 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*

1.4.3 High Availability Guide

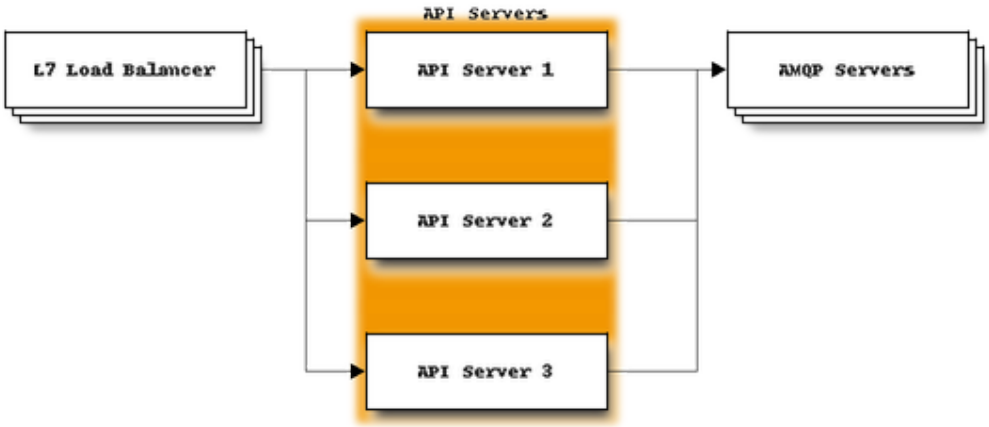
Designate supports running all of its components services in active-active HA modes.

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

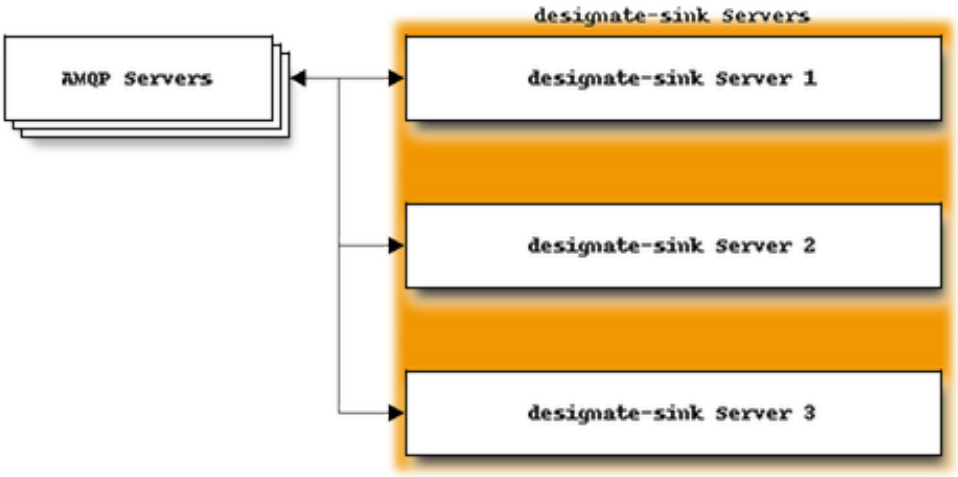
> **Warning:** Failure to set a *backend_url* can cause unexpected consequences, and may result in some periodic tasks being ran more than once.

```
[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.

1.4.4 DNS Server Pools

Overview

In designate we support the concept of multiple pools of DNS Servers.

This allows operators to scale out their DNS Service by adding more pools, avoiding the scaling problems that some DNS servers have for number of zones, and the total number of records hosted by a single server.

This also allows providers to have tiers of service (i.e. the difference between GOLD vs SILVER tiers may be the number of DNS Servers, and how they are distributed around the world.)

In a private cloud situation, it allows operators to separate internal and external facing zones.

To help users create zones on the correct pool we have a scheduler that is responsible for examining the zone being created and the pools that are available for use, and matching the zone to a pool.

The filters are pluggable (i.e. operator replaceable) and all follow a simple interface.

The zones are matched using zone attributes and pool attributes. These are key: value pairs that are attached to the zone when it is being created, and the pool. The pool attributes can be updated by the operator in the future, but it will **not** trigger zones to be moved from one pool to another.

Note: Currently the only zone attribute that is accepted is the *pool_id* attribute. As more filters are merged there will be support for dynamic filters.

Target vs. Nameserver

One thing that can be confusing about pools is the differentiation between a target and a nameserver. The target is where Designate will try to write the change, while a namserver is where Designate checks that the change exists.

A great example of this is binds stealth master system. In this configuration, there could be a stealth master that you configure as your target and a set of slaves pointed to that master as your nameservers. Designate will write to the master and then look for the changes on the slaves before considering the change active.

Another example would be where Designate uses an API backend such as DynDNS or even another Designate instance. In this situation, you will typically have a single target with a set of nameservers to test that meet your requirements.

Yet another example is when using a Designate agent. In this scenario your agent instances are the targets and the nameservers the agent updates would be checked for the correct information.

Managing Pools

In mitaka we moved the method of updating pools to a CLI in *designate-manage*

There is a YAML file that defines the pool, and is used to load this 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␣
↪nameservers.
  nameservers:
    - host: 192.0.2.2
      port: 53
```

(continues on next page)

(continued from previous page)

```
  # 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
```

Designate Manage Pools Command Reference

Update Pools Information

```
designate-manage pool update [options]
```

Options:

--file	Input file (Default: `/etc/designate/pools.yaml`)
--dry-run	This will simulate what will happen when you run this command
--delete	Any Pools not listed in the config file will be deleted

Warning:

Running with `--delete` can be **extremely** dangerous.
It will delete any pools that are not in the supplied YAML file, and any
zones that are in that Pool.
Before running with `--delete` we recommend operators run with

`--delete --dry-run` to view the outcome.

Generate YAML File

```
designate-manage pool generate_file [options]
```

Options:

--file YAML file output too (Default: `/etc/designate/pools.yaml`)

Generate YAML File from Liberty Config

```
designate-manage pool export_from_config [options]
```

Options:

--file YAML file output too (Default: `/etc/designate/pools.yaml`)

1.4.5 Pool Scheduler

In designate we have a pluggable scheduler filter interface.

You can set an ordered list of filters to run on each zone create api request.

We provide a few basic filters below, and creating custom filters follows a similar pattern to schedulers.

You can create your own by extending *designate.scheduler.filters.base.Filter* and registering a new entry point in the `designate.scheduler.filters` namespace like so in your `setup.cfg` file:

```
[entry_points]
designate.scheduler.filters =
my_custom_filter = my_extension.filters.my_custom_filter:MyCustomFilter
```

The new filter can be added to the `scheduler_filters` list in the `[service:central]` section like so:

```
[service:central]

scheduler_filters = attribute, pool_id_attribute, fallback, random, my_
↪custom_filter
```

The filters list is ran from left to right, so if the list is set to:

```
[service:central]

scheduler_filters = attribute, random
```

There will be two filters ran, the *designate.scheduler.filters.attribute_filter.AttributeFilter* followed by *designate.scheduler.filters.random_filter.RandomFilter*

Default Provided Filters

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: *designate.scheduler.filters.base.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**
Bases: *designate.scheduler.filters.base.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: *designate.scheduler.filters.base.Filter*

Randomly chooses one of the input pools if there is multiple 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: *designate.scheduler.filters.base.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: *designate.scheduler.filters.base.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.**InDoubtDefaultPoo**
Bases: *designate.scheduler.filters.base.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

1.4.6 How To Configure Multiple Pools

Designate supports pools of nameservers. A pool is a collection of nameservers and targets that Designate will write to and read from to confirm changes are successful. In some cases you might have multiple pools that you need to manage differently. For example, you might use separate pools to distribute tenants across some subset of your DNS infrastructure.

Read the section on *DNS Server Pools* to learn more about what pools are and what they can do.

Pools Configuration

Pools are configured by a *pools.yml* file. This file describes the pools and can be used to update Designate via *designate-manage* commands.

Here is an example *pools.yml* that configures two different pools. The idea is that well configure our pools to support different usage levels. Well define a *gold* and *standard* level and put zones in each based on the tenant.

Our *gold* level will provide 6 nameservers that users have access to where our *standard* will only provide 2. Both pools will have one master target we write to.

```
---

- name: golden_pool
  description: The golden pool!

  attributes:
    service_tier: gold

  ns_records:
    - hostname: ns1-gold.example.org
      priority: 1

    - hostname: ns2-gold.example.org
      priority: 2
```

(continues on next page)

(continued from previous page)

```
    - 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
        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
```

(continues on next page)

(continued from previous page)

```
    - 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
```

With our configuration in place, we can then update Designate to use the pool configuration.

```
# Do a dry run
$ designate-manage pool update --file pools.yml --dry-run
$ designate-manage pool update --file pools.yml
```

Designate now has two pools to work with. The next step will be to configure the scheduler to use the attributes when choosing what pool to store the zone on.

Pool Scheduler

The pool scheduler allows selecting a pool when a zone is created. Each scheduler acts as a filter, selecting or negating each pool based on some attributes. Designate comes with some simple schedulers to support common patterns:

- default_pool
- fallback
- random
- pool_id_attribute
- attribute

These are configured in the *service:central* section of the config.

Schedule by Pool ID Example

For example, if we wanted to allow a user to select a specific pool by id or fallback to using a default, we could use the following configuration.

```
[service:central]
default_pool_id = 794ccc2c-d751-44fe-b57f-8894c9f5c842
scheduler_filters = pool_id_attribute, fallback
```

The filters are applied 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 our tiered example, well use the *attribute* filter 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 needs the zone to go to the *gold* pool, the user needs to 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."
}
```

This ensures the zone ends up on the correct pool.

In this example, weve allowed the user to define what pool should be scheduled. If we wanted to schedule the zone based on the tenant, we could write a custom filter that looked up the appropriate group and adds the appropriate pool.

1.4.7 Blacklisting Domain Names

Zone and recordset names can be blacklisted in Designate, disallowing the creation of certain names, specified by regular expressions.

The simple use case here could be I dont want anyone to be able to create anything with `mycompany.com.` in it!, or maybe disallowing subzones on a certain zone. Or simply disallowing the creation of a single zone, like `google.com.`.

If wanted to blacklist `example.com.` and all of its subdomains, we could make the following API calls.

```
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."
}
```

Response:

```
HTTP/1.1 201 CREATED
Content-Type: application/json; charset=UTF-8
X-Openstack-Request-Id: req-bfcd0723-624c-4ec2-bbd5-99e985efe8db

{
   "description": "This blacklists \*.example.com.",
   "links": {
     "self": "http://127.0.0.1:9001/v2/blacklists/af91edb5-ede8-453f-af13-
↪feabdd088f9c"
   },
   "pattern": "^([A-Za-z0-9_\\-]+\\.)*example\\.com\\.$",
   "created_at": "2016-05-20 06:15:42",
   "updated_at": null,
   "id": "af91edb5-ede8-453f-af13-feabdd088f9c"
}
```

Now, if someone were to try and create `foo.example.com.`, or `example.com.` they would encounter an error:

```
HTTP/1.1 400 BAD REQUEST
Content-Type: application/json
X-Openstack-Request-Id: req-b7be7770-ec4f-4573-b4db-70f95475f691

{
  "message": "Blacklisted zone name",
  "code": 400,
  "type": "invalid_zone_name",
   "request_id": "req-b7be7770-ec4f-4573-b4db-70f95475f691"
}
```

Blacklists can be deleted, just like an other resource in the API, `DELETE /v2/blacklists/<id>`.

Regular Expressions

The regular expressions used here can be a bit difficult to wrap your mind around at first. Try using a tool like https://www.debuggex.com/

Its important to note that the regular expressions we enter are similar to Python regular expressions, but we need to escape certain characters when we make HTTP calls.

This means that if you wanted to debug this regex:

```
^([A-Za-z0-9_\\-]+\\.)*example\\.com\\.$
```

youre really working with this regex:

```
^([A-Za-z0-9_\\-]+\.)*example\.com\.$
```

1.4.8 View and Manage Quotas

Quotas exist in Designate for various resources, these are configurable by an operator globally, as well as on a per-tenant basis.

View Quotas

Users can view their quotas with a simple 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": 500
}
```

Administrators with the ability to use the `X-Auth-All-Projects` header can view the quotas of any user by making a similar API call to `/v2/quotas/tenant`.

Available Quotas

Zones

Quota	Description	Default
zones	The number of zone allowed per tenant	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

Editing Quotas

Global Configuration

All of the quotas above can be set as a default for all users by editing the `[DEFAULT]` configuration section, and setting each quota with `quota_$name`. for example:

```
[DEFAULT]
########################
## General Configuration
########################
quota_zones = 500
quota_zone_recordsets = 500
quota_zone_records = 500
quota_recordset_records = 20
quota_api_export_size = 1000
```

Per-Tenant via API

These quotas can be edited via API on a per-tenant basis. An administrator can edit quotas for any tenant, but they must supply the `X-Auth-All-Projects` header, and have permission to use it, theyll also need the `set-quotas` permission in `policy.json`. For example, an admin setting the zones quota for tenant X would look like:

```
PATCH /v2/quotas/tenantX HTTP/1.1
Accept: application/json
Content-Type: application/json
X-Auth-All-Projects: True

{
  "zones": 100
}
```

The response would be:

```
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": 100
}
```

Tenant Id verification

Although Designate API can accept arbitrary strings as Tenant ID to set the quota for, actual enforcement of quota will be performed only when the tenant ID that was set is matching the `project-id` in the request that attempts to create a resource.

To have some guards against possible mistakes when setting quotas, the following option can be enabled in the Designate configuration file:

```
[service:api]
quotas_verify_project_id = True
```

Additionally, the `[keystone]` section in the configuration file might have to be populated with `keystoneauth` Session- and Adapter-related options specifying how to connect to Keystone and find appropriate Keystone endpoint to perform requests against (see keystoneauth documentation for more details). Example:

```
[keystone]
cafile = /path/to/ca/bundle
valid_interfaces = internal,public
region_name = RegionWest
```

With those settings enabled, Designate will use the incoming token of user performing the `PATCH /v2/quotas/tenantX` request to make a best effort attempt to verify that the requested Tenant ID (`tenantX` part of the request) is indeed a valid Project ID in Keystone.

As a result of this verification, the `PATCH /v2/quotas/tenantX` request may return additional errors in case of:

- 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

The situation when the authorization with incoming token fails is ignored. For best results ensure that the user setting quotas is allowed to list projects in Keystone.

1.4.9 Policy Documentation

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)

primary_zone

Default `target.zone_type:SECONDARY`

(no description provided)

owner

Default `tenant:%(tenant_id)s`

(no description provided)

admin_or_owner

Default `rule:admin or rule:owner`

(no description provided)

default

Default `rule:admin_or_owner`

(no description provided)

target

Default `tenant:%(target_tenant_id)s`

(no description provided)

owner_or_target

Default `rule:target or rule:owner`

(no description provided)

admin_or_owner_or_target

Default `rule:owner_or_target or rule:admin`

(no description provided)

admin_or_target

Default `rule:admin or rule:target`

(no description provided)

zone_primary_or_admin

Default `('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR ('SECONDARY':%(zone_type)s AND is_admin:True)`

(no description provided)

create_blacklist

Default `rule:admin`

Operations

- **POST** `/v2/blacklists`

Create blacklist.

find_blacklist

Default `rule:admin`

Operations

- **GET** `/v2/blacklists`

Find blacklist.

find_blacklists

Default `rule:admin`

Operations

- **GET** `/v2/blacklists`

Find blacklists.

get_blacklist

Default `rule:admin`

Operations

- **GET** `/v2/blacklists/{blacklist_id}`

Get blacklist.

update_blacklist

Default `rule:admin`

Operations

- **PATCH** `/v2/blacklists/{blacklist_id}`

Update blacklist.

delete_blacklist

Default `rule:admin`

Operations

- **DELETE** `/v2/blacklists/{blacklist_id}`

Delete blacklist.

use_blacklisted_zone

Default `rule:admin`

Operations

- **POST** `/v2/zones`

Allowed bypass the blacklist.

all_tenants

Default `rule:admin`

Action on all tenants.

edit_managed_records

Default `rule:admin`

Edit managed records.

use_low_ttl

Default `rule:admin`

Use low TTL.

use_sudo

Default `rule:admin`

Accept sudo from user to tenant.

diagnostics_ping

Default `rule:admin`

Diagnose ping.

diagnostics_sync_zones

Default `rule:admin`

Diagnose sync zones.

diagnostics_sync_zone

Default `rule:admin`

Diagnose sync zone.

diagnostics_sync_record

Default `rule:admin`

Diagnose sync record.

create_pool

Default `rule:admin`

Create pool.

find_pools

Default `rule:admin`

Operations

- **GET** `/v2/pools`

Find pool.

find_pool

Default `rule:admin`

Operations

- **GET** `/v2/pools`

Find pools.

get_pool

Default `rule:admin`

Operations

- **GET** `/v2/pools/{pool_id}`

Get pool.

update_pool

Default `rule:admin`

Update pool.

`delete_pool`

Default `rule:admin`

Delete pool.

`zone_create_forced_pool`

Default `rule:admin`

Operations

- **POST** `/v2/zones`

load and set the pool to the one provided in the Zone attributes.

`get_quotas`

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/quotas`

View Current Projects Quotas.

`get_quota`

Default `rule:admin_or_owner`

(no description provided)

`set_quota`

Default `rule:admin`

Operations

- **PATCH** `/v2/quotas/{project_id}`

Set Quotas.

`reset_quotas`

Default `rule:admin`

Operations

- **DELETE** `/v2/quotas/{project_id}`

Reset Quotas.

`find_records`

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/reverse/floatingips/{region}:{floatingip_id}`
- **GET** `/v2/reverse/floatingips`

Find records.

count_records

Default `rule:admin_or_owner`

(no description provided)

create_recordset

Default `('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR ('SECONDARY':%(zone_type)s AND is_admin:True)`

Operations

- **POST** `/v2/zones/{zone_id}/recordsets`
- **PATCH** `/v2/reverse/floatingips/{region}:{floatingip_id}`

Create Recordset

get_recordsets

Default `rule:admin_or_owner`

(no description provided)

get_recordset

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/{zone_id}/recordsets/{recordset_id}`
- **DELETE** `/v2/zones/{zone_id}/recordsets/{recordset_id}`
- **PUT** `/v2/zones/{zone_id}/recordsets/{recordset_id}`

Get recordset

update_recordset

Default `('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR ('SECONDARY':%(zone_type)s AND is_admin:True)`

Operations

- **PUT** `/v2/zones/{zone_id}/recordsets/{recordset_id}`
- **PATCH** `/v2/reverse/floatingips/{region}:{floatingip_id}`

Update recordset

delete_recordset

Default `('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR ('SECONDARY':%(zone_type)s AND is_admin:True)`

Operations

- **DELETE** `/v2/zones/{zone_id}/recordsets/{recordset_id}`

Delete RecordSet

count_recordset

Default `rule:admin_or_owner`

Count recordsets

find_service_status

Default `rule:admin`

Operations

- **GET** `/v2/service_status/{service_id}`

Find a single Service Status

find_service_statuses

Default `rule:admin`

Operations

- **GET** `/v2/service_status`

List service statuses.

update_service_status

Default `rule:admin`

(no description provided)

find_tenants

Default `rule:admin`

Find all Tenants.

get_tenant

Default `rule:admin`

Get all Tenants.

count_tenants

Default `rule:admin`

Count tenants

create_tld

Default `rule:admin`

Operations

- **POST** `/v2/tlds`

Create Tld

find_tlds

Default `rule:admin`

Operations

- **GET** `/v2/tlds`

List Tlds

get_tld

Default `rule:admin`

Operations

- **GET** `/v2/tlds/{tld_id}`

Show Tld

update_tld

Default `rule:admin`

Operations

- **PATCH** `/v2/tlds/{tld_id}`

Update Tld

delete_tld

Default `rule:admin`

Operations

- **DELETE** `/v2/tlds/{tld_id}`

Delete Tld

create_tsigkey

Default `rule:admin`

Operations

- **POST** `/v2/tsigkeys`

Create Tsigkey

find_tsigkeys

Default `rule:admin`

Operations

- **GET** `/v2/tsigkeys`

List Tsigkeys

get_tsigkey

Default `rule:admin`

Operations

- **PATCH** `/v2/tsigkeys/{tsigkey_id}`
- **GET** `/v2/tsigkeys/{tsigkey_id}`

Show a Tsigkey

update_tsigkey

Default `rule:admin`

Operations

- **PATCH** `/v2/tsigkeys/{tsigkey_id}`

Update Tsigkey

`delete_tsigkey`

Default `rule:admin`

Operations

- **DELETE** `/v2/tsigkeys/{tsigkey_id}`

Delete a Tsigkey

`create_zone`

Default `rule:admin_or_owner`

Operations

- **POST** `/v2/zones`

Create Zone

`get_zones`

Default `rule:admin_or_owner`

(no description provided)

`get_zone`

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/{zone_id}`
- **PATCH** `/v2/zones/{zone_id}`
- **PUT** `/v2/zones/{zone_id}/recordsets/{recordset_id}`

Get Zone

`get_zone_servers`

Default `rule:admin_or_owner`

(no description provided)

`find_zones`

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones`

List existing zones

`update_zone`

Default `rule:admin_or_owner`

Operations

- **PATCH** `/v2/zones/{zone_id}`

Update Zone

`delete_zone`

Default `rule:admin_or_owner`

Operations

- **DELETE** `/v2/zones/{zone_id}`

Delete Zone

`xfr_zone`

Default `rule:admin_or_owner`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/xfr`

Manually Trigger an Update of a Secondary Zone

`abandon_zone`

Default `rule:admin`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/abandon`

Abandon Zone

`count_zones`

Default `rule:admin_or_owner`

(no description provided)

`count_zones_pending_notify`

Default `rule:admin_or_owner`

(no description provided)

`purge_zones`

Default `rule:admin`

(no description provided)

`touch_zone`

Default `rule:admin_or_owner`

(no description provided)

`zone_export`

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/tasks/exports/{zone_export_id}/export`

Retrive a Zone Export from the Designate Datastore

create_zone_export

Default `rule:admin_or_owner`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/export`

Create Zone Export

find_zone_exports

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/tasks/exports`

List Zone Exports

get_zone_export

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/tasks/exports/{zone_export_id}`
- **GET** `/v2/zones/tasks/exports/{zone_export_id}/export`

Get Zone Exports

update_zone_export

Default `rule:admin_or_owner`

Operations

- **POST** `/v2/zones/{zone_id}/tasks/export`

Update Zone Exports

create_zone_import

Default `rule:admin_or_owner`

Operations

- **POST** `/v2/zones/tasks/imports`

Create Zone Import

find_zone_imports

Default `rule:admin_or_owner`

Operations

- **GET** `/v2/zones/tasks/imports`

List all Zone Imports

get_zone_import

Default `rule:admin_or_owner`

Operations

- **GET** /v2/zones/tasks/imports/{zone_import_id}

Get Zone Imports

update_zone_import

Default rule:admin_or_owner

Operations

- **POST** /v2/zones/tasks/imports

Update Zone Imports

delete_zone_import

Default rule:admin_or_owner

Operations

- **GET** /v2/zones/tasks/imports/{zone_import_id}

Delete a Zone Import

create_zone_transfer_accept

Default rule:admin_or_owner OR tenant:%(target_tenant_id)s OR None:%(target_tenant_id)s

Operations

- **POST** /v2/zones/tasks/transfer_accepts

Create Zone Transfer Accept

get_zone_transfer_accept

Default rule:admin_or_owner

Operations

- **GET** /v2/zones/tasks/transfer_requests/{zone_transfer_accept_id}

Get Zone Transfer Accept

find_zone_transfer_accepts

Default rule:admin

Operations

- **GET** /v2/zones/tasks/transfer_accepts

List Zone Transfer Accepts

find_zone_transfer_accept

Default rule:admin

(no description provided)

update_zone_transfer_accept

Default rule:admin

Operations

- **POST** /v2/zones/tasks/transfer_accepts

Update a Zone Transfer Accept

delete_zone_transfer_accept

Default rule:admin

(no description provided)

create_zone_transfer_request

Default rule:admin_or_owner

Operations

- **POST** /v2/zones/{zone_id}/tasks/transfer_requests

Create Zone Transfer Accept

get_zone_transfer_request

Default rule:admin_or_owner OR tenant:%(target_tenant_id)s OR None:%(target_tenant_id)s

Operations

- **GET** /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
- **PATCH** /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}

Show a Zone Transfer Request

get_zone_transfer_request_detailed

Default rule:admin_or_owner

(no description provided)

find_zone_transfer_requests

Default @

Operations

- **GET** /v2/zones/tasks/transfer_requests

List Zone Transfer Requests

find_zone_transfer_request

Default @

(no description provided)

update_zone_transfer_request

Default rule:admin_or_owner

Operations

- **PATCH** /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}

Update a Zone Transfer Request

delete_zone_transfer_request

Default `rule:admin_or_owner`

Operations

- **DELETE** `/v2/zones/tasks/transfer_requests/{zone_transfer_request_id}`

Delete a Zone Transfer Request

1.4.10 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']`

Supported record types

backlog

Type integer

Default `4096`

Number of backlog requests to configure the socket with

`tcp_keepidle`

Type integer

Default `600`

Sets the value of TCP_KEEPIDLE in seconds for each server socket. Not supported on OS X.

`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

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 3: 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 4: 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 5: 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.

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.

use_eventlog

Type boolean

Default `False`

Log output to Windows Event Log.

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 setto 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 [%(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 %(tenant)s %(domain)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`

Log level name used by rate limiting: CRITICAL, ERROR, INFO, WARNING, DEBUG or empty string. 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.

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.

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.

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

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.

tcp_keepidle

Type integer

Default `600`

Sets the value of TCP_KEEPIDLE in seconds for each server socket. Not supported on OS X.

wsgi_default_pool_size

Type integer

Default `100`

Size of the pool of greenthreads used by wsgi

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).

wsgi_keep_alive

Type boolean

Default `True`

If False, closes the client socket connection explicitly.

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.

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.

rpc_conn_pool_size

Type integer

Default `30`

Minimum Value 1

Size of RPC connection pool.

Table 6: Deprecated Variations

Group	Name
DEFAULT	rpc_conn_pool_size

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

executor_thread_pool_size

Type integer

Default `64`

Size of executor thread pool when executor is threading or eventlet.

Table 7: 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

backend:agent:bind9

rndc_host

Type string

Default `127.0.0.1`

RNDC Host

rndc_port

Type integer

Default `953`

RNDC Port

rndc_config_file

Type string

Default `<None>`

RNDC Config File

rndc_key_file

Type string

Default `<None>`

RNDC Key File

zone_file_path

Type string

Default `$state_path/zones`

Path where zone files are stored

query_destination

Type string

Default `127.0.0.1`

Host to query when finding zones

backend:agent:denominator

name

Type string

Default `fake`

Name of the affected provider

config_file

Type string

Default `/etc/denominator.conf`

Path to Denominator configuration file

backend:agent:djbdns

tcpclient_cmd_name

Type string

Default `tcpclient`

tcpclient executable path or rootwrap command name

axfr_get_cmd_name

Type string

Default `axfr-get`

axfr-get executable path or rootwrap command name

tinydns_data_cmd_name

Type string

Default `tinydns-data`

tinydns-data executable path or rootwrap command name

tinydns_datadir

Type string

Default `/var/lib/djbdns`

TinyDNS data directory

query_destination

Type string

Default `127.0.0.1`

Host to query when finding zones

backend:agent:gdnsd

gdnsd_cmd_name

Type string

Default `gdnsd`

gdnsd executable path or rootwrap command name

confdir_path

Type string

Default `/etc/gdnsd`

gdnsd configuration directory path

query_destination

Type string

Default `127.0.0.1`

Host to query when finding zones

backend:agent:knot2

knotc_cmd_name

Type string

Default `knotc`

knotc executable path or rootwrap command name

query_destination

Type string

Default `127.0.0.1`

Host to query when finding zones

backend:agent:msdns

backend:akamai

enhanceddns_wsdl

Type string

Default `/path/to/EnhancedDNS.xml`

This option has a sample default set, which means that its actual default value may vary from the one documented above.

Akamai EnhancedDNS WSDL URL

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.

backend:infoblox

wapi_url

Type string

Default `<None>`

DEPRECATED: wapi_url

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

username

Type string

Default `<None>`

DEPRECATED: username

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

password

Type string

Default `<None>`

DEPRECATED: password

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

`sslverify`

Type boolean

Default `True`

DEPRECATED: sslverify

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

`multi_tenant`

Type boolean

Default `False`

DEPRECATED: multi_tenant

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

`http_pool_connections`

Type integer

Default `100`

DEPRECATED: http_pool_connections

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

`http_pool_maxsize`

Type integer

Default `100`

DEPRECATED: http_pool_maxsize

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

dns_view

Type string

Default `default`

DEPRECATED: dns_view

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

network_view

Type string

Default `default`

DEPRECATED: network_view

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

ns_group

Type string

Default `<None>`

DEPRECATED: ns_group

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** All backend options have been migrated to options in the pools.yaml file

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',
    '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.

Table 8: Deprecated Variations

Group	Name
DEFAULT	sqlite_synchronous

backend

Type string

Default `sqlalchemy`

The back end to use for the database.

Table 9: Deprecated Variations

Group	Name
DEFAULT	db_backend

connection

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the database.

Table 10: Deprecated Variations

Group	Name
DEFAULT	sql_connection
DATABASE	sql_connection
sql	connection

slave_connection

Type string

Default `<None>`

The SQLAlchemy 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_enable_ndb

Type boolean

Default `False`

If True, transparently enables support for handling MySQL Cluster (NDB).

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.

Table 11: Deprecated Variations

Group	Name
DATABASE	idle_timeout
database	idle_timeout
DEFAULT	sql_idle_timeout
DATABASE	sql_idle_timeout
sql	idle_timeout

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.

Table 12: Deprecated Variations

Group	Name
DEFAULT	sql_max_pool_size
DATABASE	sql_max_pool_size

max_retries

Type integer

Default `10`

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

Table 13: Deprecated Variations

Group	Name
DEFAULT	sql_max_retries
DATABASE	sql_max_retries

retry_interval

Type integer

Default `10`

Interval between retries of opening a SQL connection.

Table 14: Deprecated Variations

Group	Name
DEFAULT	sql_retry_interval
DATABASE	reconnect_interval

max_overflow

Type integer

Default `50`

If set, use this value for max_overflow with SQLAlchemy.

Table 15: Deprecated Variations

Group	Name
DEFAULT	sql_max_overflow
DATABASE	sqlalchemy_max_overflow

connection_debug

Type integer

Default `0`

Minimum Value 0

Maximum Value 100

Verbosity of SQL debugging information: 0=None, 100=Everything.

Table 16: Deprecated Variations

Group	Name
DEFAULT	sql_connection_debug

connection_trace

Type boolean

Default `False`

Add Python stack traces to SQL as comment strings.

Table 17: Deprecated Variations

Group	Name
DEFAULT	sql_connection_trace

pool_timeout

Type integer

Default `<None>`

If set, use this value for pool_timeout with SQLAlchemy.

Table 18: Deprecated Variations

Group	Name
DATABASE	sqlalchemy_pool_timeout

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.

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.

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.

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 19: 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 20: 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 `False`

(Optional) Use the advanced (eventlet safe) memcached client pool. The advanced pool will only work under python 2.x.

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.

auth_type

Type unknown type

Default `<None>`

Authentication type to load

Table 21: Deprecated Variations

Group	Name
keystone_authtoken	auth_plugin

auth_section

Type unknown type

Default `<None>`

Config Section from which to load plugin specific options

monasca:statsd

enabled

Type boolean

Default `False`

enable

port

Type integer

Default `8125`

UDP port

hostname

Type string

Default `127.0.0.1`

hostname

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

admin_username

Type string

Default `<None>`

username for connecting to neutron in admin context

admin_password

Type string

Default `<None>`

password for connecting to neutron in admin context

admin_tenant_name

Type string

Default `<None>`

tenant name for connecting to neutron in admin context

auth_url

Type string

Default `<None>`

auth url for connecting to neutron in admin context

insecure

Type boolean

Default `False`

if set, ignore any SSL validation issues

auth_strategy

Type string

Default `keystone`

auth strategy for connecting to neutron in admin context

ca_certificates_file

Type string

Default `<None>`

Location of ca certificates file to use for neutron client requests.

oslo_concurrency

disable_process_locking

Type boolean

Default `False`

Enables or disables inter-process locks.

Table 22: Deprecated Variations

Group	Name
DEFAULT	disable_process_locking

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.

Table 23: Deprecated Variations

Group	Name
DEFAULT	lock_path

oslo_messaging_amqp

container_name

Type string

Default <None>

Name for the AMQP container. must be globally unique. Defaults to a generated UUID

Table 24: Deprecated Variations

Group	Name
amqp1	container_name

idle_timeout

Type integer

Default 0

Timeout for inactive connections (in seconds)

Table 25: Deprecated Variations

Group	Name
amqp1	idle_timeout

trace

Type boolean

Default False

Debug: dump AMQP frames to stdout

Table 26: Deprecated Variations

Group	Name
amqp1	trace

ssl

Type boolean

Default False

Attempt to connect via SSL. If no other ssl-related parameters are given, it will use the systems CA-bundle to verify the servers certificate.

ssl_ca_file

Type string

Default ''

CA certificate PEM file used to verify the servers certificate

Table 27: Deprecated Variations

Group	Name
amqp1	ssl_ca_file

ssl_cert_file

Type string

Default `''`

Self-identifying certificate PEM file for client authentication

Table 28: Deprecated Variations

Group	Name
amqp1	ssl_cert_file

ssl_key_file

Type string

Default `''`

Private key PEM file used to sign ssl_cert_file certificate (optional)

Table 29: Deprecated Variations

Group	Name
amqp1	ssl_key_file

ssl_key_password

Type string

Default `<None>`

Password for decrypting ssl_key_file (if encrypted)

Table 30: Deprecated Variations

Group	Name
amqp1	ssl_key_password

ssl_verify_vhost

Type boolean

Default `False`

By default SSL checks that the name in the servers certificate matches the hostname in the transport_url. In some configurations it may be preferable to use the virtual hostname instead, for example if the server uses the Server Name Indication TLS extension (rfc6066) to provide a certificate per virtual host. Set ssl_verify_vhost to True if the servers SSL certificate uses the virtual host name instead of the DNS name.

sasl_mechanisms

Type string

Default `''`

Space separated list of acceptable SASL mechanisms

Table 31: Deprecated Variations

Group	Name
amqp1	sasl_mechanisms

`sasl_config_dir`

Type string

Default `''`

Path to directory that contains the SASL configuration

Table 32: Deprecated Variations

Group	Name
amqp1	sasl_config_dir

`sasl_config_name`

Type string

Default `''`

Name of configuration file (without .conf suffix)

Table 33: Deprecated Variations

Group	Name
amqp1	sasl_config_name

`sasl_default_realm`

Type string

Default `''`

SASL realm to use if no realm present in username

`connection_retry_interval`

Type integer

Default 1

Minimum Value 1

Seconds to pause before attempting to re-connect.

`connection_retry_backoff`

Type integer

Default 2

Minimum Value 0

Increase the connection_retry_interval by this many seconds after each unsuccessful failover attempt.

connection_retry_interval_max

Type integer

Default 30

Minimum Value 1

Maximum limit for connection_retry_interval + connection_retry_backoff

link_retry_delay

Type integer

Default 10

Minimum Value 1

Time to pause between re-connecting an AMQP 1.0 link that failed due to a recoverable error.

default_reply_retry

Type integer

Default 0

Minimum Value -1

The maximum number of attempts to re-send a reply message which failed due to a recoverable error.

default_reply_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc reply message delivery.

default_send_timeout

Type integer

Default 30

Minimum Value 5

The deadline for an rpc cast or call message delivery. Only used when caller does not provide a timeout expiry.

default_notify_timeout

Type integer

Default 30

Minimum Value 5

The deadline for a sent notification message delivery. Only used when caller does not provide a timeout expiry.

default_sender_link_timeout

Type integer

Default `600`

Minimum Value 1

The duration to schedule a purge of idle sender links. Detach link after expiry.

`addressing_mode`

Type string

Default `dynamic`

Indicates the addressing mode used by the driver. Permitted values: legacy - use legacy non-routable addressing routable - use routable addresses dynamic - use legacy addresses if the message bus does not support routing otherwise use routable addressing

`pseudo_vhost`

Type boolean

Default `True`

Enable virtual host support for those message buses that do not natively support virtual hosting (such as qpidd). When set to true the virtual host name will be added to all message bus addresses, effectively creating a private subnet per virtual host. Set to False if the message bus supports virtual hosting using the hostname field in the AMQP 1.0 Open performative as the name of the virtual host.

`server_request_prefix`

Type string

Default `exclusive`

address prefix used when sending to a specific server

Table 34: Deprecated Variations

Group	Name
amqp1	server_request_prefix

`broadcast_prefix`

Type string

Default `broadcast`

address prefix used when broadcasting to all servers

Table 35: Deprecated Variations

Group	Name
amqp1	broadcast_prefix

`group_request_prefix`

Type string

Default `unicast`

address prefix when sending to any server in group

Table 36: Deprecated Variations

Group	Name
amqp1	group_request_prefix

rpc_address_prefix

Type string

Default `openstack.org/om/rpc`

Address prefix for all generated RPC addresses

notify_address_prefix

Type string

Default `openstack.org/om/notify`

Address prefix for all generated Notification addresses

multicast_address

Type string

Default `multicast`

Appended to the address prefix when sending a fanout message. Used by the message bus to identify fanout messages.

unicast_address

Type string

Default `unicast`

Appended to the address prefix when sending to a particular RPC/Notification server. Used by the message bus to identify messages sent to a single destination.

anycast_address

Type string

Default `anycast`

Appended to the address prefix when sending to a group of consumers. Used by the message bus to identify messages that should be delivered in a round-robin fashion across consumers.

default_notification_exchange

Type string

Default `<None>`

Exchange name used in notification addresses. Exchange name resolution precedence: Target.exchange if set else default_notification_exchange if set else control_exchange if set else notify

default_rpc_exchange

Type string

Default `<None>`

Exchange name used in RPC addresses. Exchange name resolution precedence: Target.exchange if set else default_rpc_exchange if set else control_exchange if set else rpc

reply_link_credit

Type integer

Default `200`

Minimum Value 1

Window size for incoming RPC Reply messages.

rpc_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming RPC Request messages

notify_server_credit

Type integer

Default `100`

Minimum Value 1

Window size for incoming Notification messages

pre_settled

Type multi-valued

Default `rpc-cast`

Default `rpc-reply`

Send messages of this type pre-settled. Pre-settled messages will not receive acknowledgement from the peer. Note well: pre-settled messages may be silently discarded if the delivery fails. Permitted values: rpc-call - send RPC Calls pre-settled rpc-reply- send RPC Replies pre-settled rpc-cast - Send RPC Casts pre-settled notify - Send Notifications pre-settled

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

pool_size

Type integer

Default `10`

Pool Size for Kafka Consumers

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_min_size

Type integer

Default `2`

The pool size limit for connections expiration policy

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

conn_pool_ttl

Type integer

Default `1200`

The time-to-live in sec of idle connections in the pool

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** Driver no longer uses connection pool.

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

Table 37: Deprecated Variations

Group	Name
DEFAULT	notification_driver

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.

Table 38: Deprecated Variations

Group	Name
DEFAULT	notification_transport_url

topics

Type list

Default ['notifications']

AMQP topic used for OpenStack notifications.

Table 39: Deprecated Variations

Group	Name
rpc_notifier2	topics
DEFAULT	notification_topics

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.

amqp_auto_delete

Type boolean

Default `False`

Auto-delete queues in AMQP.

Table 40: Deprecated Variations

Group	Name
DEFAULT	amqp_auto_delete

ssl

Type boolean

Default `False`

Connect over SSL.

Table 41: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_use_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.

Table 42: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_version

ssl_key_file

Type string

Default `''`

SSL key file (valid only if SSL enabled).

Table 43: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_keyfile

ssl_cert_file

Type string

Default ''

SSL cert file (valid only if SSL enabled).

Table 44: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_certfile

ssl_ca_file

Type string

Default ''

SSL certification authority file (valid only if SSL enabled).

Table 45: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_ssl_ca_certs

heartbeat_in_pthread

Type boolean

Default False

EXPERIMENTAL: Run the health check heartbeat thread through a native python thread. By default if this option isnt provided the health check heartbeat will inherit the execution model from the parent process. By example if the parent process have monkey patched the stdlib by using eventlet/greenlet then the heartbeat will be run through a green thread.

kombu_reconnect_delay

Type floating point

Default 1.0

How long to wait before reconnecting in response to an AMQP consumer cancel notification.

Table 46: Deprecated Variations

Group	Name
DEFAULT	kombu_reconnect_delay

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 47: 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, RABBIT-CR-DEMO

The RabbitMQ login method.

Table 48: Deprecated Variations

Group	Name
DEFAULT	rabbit_login_method

rabbit_retry_interval

Type integer

Default `1`

How frequently to retry connecting with RabbitMQ.

rabbit_retry_backoff

Type integer

Default `2`

How long to backoff for between retries when connecting to RabbitMQ.

Table 49: Deprecated Variations

Group	Name
DEFAULT	rabbit_retry_backoff

rabbit_interval_max

Type integer

Default `30`

Maximum interval of RabbitMQ connection retries. Default is 30 seconds.

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}

Table 50: Deprecated Variations

Group	Name
DEFAULT	rabbit_ha_queues

rabbit_transient_queues_ttl

Type integer

Default `1800`

Minimum Value 1

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.

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 `2`

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

oslo_middleware

`max_request_body_size`

Type integer

Default `114688`

The maximum body size for each request, in bytes.

Table 51: Deprecated Variations

Group	Name
DEFAULT	osapi_max_request_body_size
DEFAULT	max_request_body_size

`secure_proxy_ssl_header`

Type string

Default `X-Forwarded-Proto`

The HTTP Header that will be used to determine what the original request protocol scheme was, even if it was hidden by a SSL termination proxy.

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.

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.

oslo_policy

enforce_scope

Type boolean

Default `False`

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.

enforce_new_defaults

Type boolean

Default `False`

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

policy_file

Type string

Default `policy.json`

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.

Table 52: Deprecated Variations

Group	Name
DEFAULT	policy_file

policy_default_rule

Type string

Default `default`

Default rule. Enforced when a requested rule is not found.

Table 53: Deprecated Variations

Group	Name
DEFAULT	policy_default_rule

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.

Table 54: Deprecated Variations

Group	Name
DEFAULT	policy_dirs

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

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:agent

workers

Type integer

Default `<None>`

Number of agent worker processes to spawn

threads

Type integer

Default `1000`

Number of agent greenthreads to spawn

listen

Type list

Default `['0.0.0.0:5358']`

Agent host:port pairs to listen on

tcp_backlog

Type integer

Default `100`

The Agent TCP Backlog

tcp_recv_timeout

Type floating point

Default `0.5`

Agent TCP Receive Timeout

allow_notify

Type list

Default `[]`

List of IP addresses allowed to NOTIFY The Agent

masters

Type list

Default `[]`

List of masters for the Agent, format ip:port

backend_driver

Type string

Default `bind9`

The backend driver to use, e.g. bind9, djbdns, knot2

transfer_source

Type string

Default `<None>`

An IP address to be used to fetch zones transferred in

notify_delay

Type floating point

Default `0.0`

Delay after a NOTIFY arrives for a zone that the Agent will pause and drop subsequent NOTIFYs for that zone

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-api-v2 which enable in a future

enable_api_admin

Type boolean

Default `False`

enable-api-admin

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 by the Keystone v3 API with big service catalogs).

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.

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

`secure_proxy_ssl_header`

Type string

Default `X-Forwarded-Proto`

The HTTP Header that will be used to determine which the original request protocol scheme was, even if it was removed by an SSL terminating proxy.

`override_proto`

Type string

Default `<None>`

A scheme that will be used to override the request protocol scheme, even if it was set by an SSL terminating proxy.

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

`storage_driver`

Type string

Default `sqlalchemy`

The storage driver to use

`enabled_notification_handlers`

Type list

Default `[]`

Enabled Notification Handlers

`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 55: 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 `<None>`

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_recv_timeout

Type floating point

Default 0.5

mDNS TCP Receive Timeout

all_tcp

Type boolean

Default False

Send all traffic over TCP

query_enforce_tsig

Type boolean

Default False

Enforce all incoming queries (including AXFR) are TSIG signed

storage_driver

Type string

Default sqlalchemy

The storage driver to use

max_message_size

Type integer

Default `65535`

Maximum message size to emit

`topic`

Type string

Default `mdns`

RPC topic name for mdns

`xfr_timeout`

Type integer

Default `10`

Timeout in seconds for XFRs.

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

`storage_driver`

Type string

Default `sqlalchemy`

The storage driver to use

`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

`storage_driver`

Type string

Default `sqlalchemy`

The storage driver to use

`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

`notify`

Type boolean

Default `True`

Whether to allow worker to send NOTIFYs, this will noop NOTIFYs in mdns if true

> **Warning:** This option is deprecated for removal. Its value may be silently ignored in the future.
>
> **Reason** This option is being removed to reduce complexity

export_synchronous

Type boolean

Default `True`

Whether to allow synchronous zone exports

topic

Type string

Default `worker`

RPC topic name for worker

ssl

ca_file

Type string

Default `<None>`

CA certificate file to use to verify connecting clients.

Table 56: Deprecated Variations

Group	Name
DEFAULT	ssl_ca_file

cert_file

Type string

Default `<None>`

Certificate file to use when starting the server securely.

Table 57: Deprecated Variations

Group	Name
DEFAULT	ssl_cert_file

key_file

Type string

Default `<None>`

Private key file to use when starting the server securely.

Table 58: Deprecated Variations

Group	Name
DEFAULT	ssl_key_file

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.

`ciphers`

Type string

Default `<None>`

Sets the list of available ciphers. value should be a string in the OpenSSL cipher list format.

storage:sqlalchemy

`sqlite_synchronous`

Type boolean

Default `True`

If True, SQLite uses synchronous mode.

Table 59: Deprecated Variations

Group	Name
DEFAULT	sqlite_synchronous

`backend`

Type string

Default `sqlalchemy`

The back end to use for the database.

Table 60: Deprecated Variations

Group	Name
DEFAULT	db_backend

`connection`

Type string

Default `<None>`

The SQLAlchemy connection string to use to connect to the database.

Table 61: Deprecated Variations

Group	Name
DEFAULT	sql_connection
DATABASE	sql_connection
sql	connection

`slave_connection`

Type string

Default `<None>`

The SQLAlchemy 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_enable_ndb

Type boolean

Default `False`

If True, transparently enables support for handling MySQL Cluster (NDB).

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.

Table 62: Deprecated Variations

Group	Name
DATABASE	idle_timeout
database	idle_timeout
DEFAULT	sql_idle_timeout
DATABASE	sql_idle_timeout
sql	idle_timeout

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.

Table 63: Deprecated Variations

Group	Name
DEFAULT	sql_max_pool_size
DATABASE	sql_max_pool_size

max_retries

Type integer

Default `10`

Maximum number of database connection retries during startup. Set to -1 to specify an infinite retry count.

Table 64: Deprecated Variations

Group	Name
DEFAULT	sql_max_retries
DATABASE	sql_max_retries

retry_interval

Type integer

Default `10`

Interval between retries of opening a SQL connection.

Table 65: Deprecated Variations

Group	Name
DEFAULT	sql_retry_interval
DATABASE	reconnect_interval

max_overflow

Type integer

Default `50`

If set, use this value for max_overflow with SQLAlchemy.

Table 66: Deprecated Variations

Group	Name
DEFAULT	sql_max_overflow
DATABASE	sqlalchemy_max_overflow

connection_debug

Type integer

Default `0`

Minimum Value 0

Maximum Value 100

Verbosity of SQL debugging information: 0=None, 100=Everything.

Table 67: Deprecated Variations

Group	Name
DEFAULT	sql_connection_debug

connection_trace

Type boolean

Default `False`

Add Python stack traces to SQL as comment strings.

Table 68: Deprecated Variations

Group	Name
DEFAULT	sql_connection_trace

pool_timeout

Type integer

Default `<None>`

If set, use this value for pool_timeout with SQLAlchemy.

Table 69: Deprecated Variations

Group	Name
DATABASE	sqlalchemy_pool_timeout

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&

1.4.11 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

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",
```

(continues on next page)

(continued from previous page)

```
    "_context_read_deleted" : "no"
}
```

More examples can be found at `designate/tests/resources/sample_notifications`

1.4.12 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
```

1.4.13 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␣
↪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
```

(continues on next page)

(continued from previous page)

```
    port: 53
```

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-agentss` 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 entrypoints 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* for more details.

Deprecation Notices

- `designate-zone-manager` and `designate-pool-manager` remain deprecated and will be removed in a future release.

1.4.14 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
Agent	TCP	5358
	UDP	5358
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

OpenStack recommends Monasca

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

1.4.15 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"

#"primary_zone": "target.zone_type:SECONDARY"

#"owner": "tenant:%(tenant_id)s"

#"admin_or_owner": "rule:admin or rule:owner"

#"default": "rule:admin_or_owner"

#"target": "tenant:%(target_tenant_id)s"

#"owner_or_target": "rule:target or rule:owner"
```

(continues on next page)

(continued from previous page)

```
#"admin_or_owner_or_target": "rule:owner_or_target or rule:admin"

#"admin_or_target": "rule:admin or rule:target"

#"zone_primary_or_admin": "('PRIMARY':%(zone_type)s and rule:admin_or_
↪owner) OR ('SECONDARY':%(zone_type)s AND is_admin:True)"

# Create blacklist.
# POST  /v2/blacklists
#"create_blacklist": "rule:admin"

# Find blacklist.
# GET  /v2/blacklists
#"find_blacklist": "rule:admin"

# Find blacklists.
# GET  /v2/blacklists
#"find_blacklists": "rule:admin"

# Get blacklist.
# GET  /v2/blacklists/{blacklist_id}
#"get_blacklist": "rule:admin"

# Update blacklist.
# PATCH  /v2/blacklists/{blacklist_id}
#"update_blacklist": "rule:admin"

# Delete blacklist.
# DELETE  /v2/blacklists/{blacklist_id}
#"delete_blacklist": "rule:admin"

# Allowed bypass the blacklist.
# POST  /v2/zones
#"use_blacklisted_zone": "rule:admin"

# Action on all tenants.
#"all_tenants": "rule:admin"

# Edit managed records.
#"edit_managed_records": "rule:admin"

# Use low TTL.
#"use_low_ttl": "rule:admin"

# Accept sudo from user to tenant.
#"use_sudo": "rule:admin"

# Diagnose ping.
#"diagnostics_ping": "rule:admin"

# Diagnose sync zones.
#"diagnostics_sync_zones": "rule:admin"

# Diagnose sync zone.
#"diagnostics_sync_zone": "rule:admin"
```

(continues on next page)

(continued from previous page)

```
# Diagnose sync record.
#"diagnostics_sync_record": "rule:admin"

# Create pool.
#"create_pool": "rule:admin"

# Find pool.
# GET  /v2/pools
#"find_pools": "rule:admin"

# Find pools.
# GET  /v2/pools
#"find_pool": "rule:admin"

# Get pool.
# GET  /v2/pools/{pool_id}
#"get_pool": "rule:admin"

# Update pool.
#"update_pool": "rule:admin"

# Delete pool.
#"delete_pool": "rule:admin"

# load and set the pool to the one provided in the Zone attributes.
# POST  /v2/zones
#"zone_create_forced_pool": "rule:admin"

# View Current Project's Quotas.
# GET  /v2/quotas
#"get_quotas": "rule:admin_or_owner"

#"get_quota": "rule:admin_or_owner"

# Set Quotas.
# PATCH  /v2/quotas/{project_id}
#"set_quota": "rule:admin"

# Reset Quotas.
# DELETE  /v2/quotas/{project_id}
#"reset_quotas": "rule:admin"

# Find records.
# GET  /v2/reverse/floatingips/{region}:{floatingip_id}
# GET  /v2/reverse/floatingips
#"find_records": "rule:admin_or_owner"

#"count_records": "rule:admin_or_owner"

# Create Recordset
# POST  /v2/zones/{zone_id}/recordsets
# PATCH  /v2/reverse/floatingips/{region}:{floatingip_id}
#"create_recordset": "('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR␣
↪('SECONDARY':%(zone_type)s AND is_admin:True)"

#"get_recordsets": "rule:admin_or_owner"
```

(continues on next page)

(continued from previous page)

```
# Get recordset
# GET  /v2/zones/{zone_id}/recordsets/{recordset_id}
# DELETE  /v2/zones/{zone_id}/recordsets/{recordset_id}
# PUT  /v2/zones/{zone_id}/recordsets/{recordset_id}
#"get_recordset": "rule:admin_or_owner"

# Update recordset
# PUT  /v2/zones/{zone_id}/recordsets/{recordset_id}
# PATCH  /v2/reverse/floatingips/{region}:{floatingip_id}
#"update_recordset": "('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR␣
↪('SECONDARY':%(zone_type)s AND is_admin:True)"

# Delete RecordSet
# DELETE  /v2/zones/{zone_id}/recordsets/{recordset_id}
#"delete_recordset": "('PRIMARY':%(zone_type)s and rule:admin_or_owner) OR␣
↪('SECONDARY':%(zone_type)s AND is_admin:True)"

# Count recordsets
#"count_recordset": "rule:admin_or_owner"

# Find a single Service Status
# GET  /v2/service_status/{service_id}
#"find_service_status": "rule:admin"

# List service statuses.
# GET  /v2/service_status
#"find_service_statuses": "rule:admin"

#"update_service_status": "rule:admin"

# Find all Tenants.
#"find_tenants": "rule:admin"

# Get all Tenants.
#"get_tenant": "rule:admin"

# Count tenants
#"count_tenants": "rule:admin"

# Create Tld
# POST  /v2/tlds
#"create_tld": "rule:admin"

# List Tlds
# GET  /v2/tlds
#"find_tlds": "rule:admin"

# Show Tld
# GET  /v2/tlds/{tld_id}
#"get_tld": "rule:admin"

# Update Tld
# PATCH  /v2/tlds/{tld_id}
#"update_tld": "rule:admin"
```

(continues on next page)

(continued from previous page)

```
# Delete Tld
# DELETE  /v2/tlds/{tld_id}
#"delete_tld": "rule:admin"

# Create Tsigkey
# POST  /v2/tsigkeys
#"create_tsigkey": "rule:admin"

# List Tsigkeys
# GET  /v2/tsigkeys
#"find_tsigkeys": "rule:admin"

# Show a Tsigkey
# PATCH  /v2/tsigkeys/{tsigkey_id}
# GET  /v2/tsigkeys/{tsigkey_id}
#"get_tsigkey": "rule:admin"

# Update Tsigkey
# PATCH  /v2/tsigkeys/{tsigkey_id}
#"update_tsigkey": "rule:admin"

# Delete a Tsigkey
# DELETE  /v2/tsigkeys/{tsigkey_id}
#"delete_tsigkey": "rule:admin"

# Create Zone
# POST  /v2/zones
#"create_zone": "rule:admin_or_owner"

#"get_zones": "rule:admin_or_owner"

# Get Zone
# GET  /v2/zones/{zone_id}
# PATCH  /v2/zones/{zone_id}
# PUT  /v2/zones/{zone_id}/recordsets/{recordset_id}
#"get_zone": "rule:admin_or_owner"

#"get_zone_servers": "rule:admin_or_owner"

# List existing zones
# GET  /v2/zones
#"find_zones": "rule:admin_or_owner"

# Update Zone
# PATCH  /v2/zones/{zone_id}
#"update_zone": "rule:admin_or_owner"

# Delete Zone
# DELETE  /v2/zones/{zone_id}
#"delete_zone": "rule:admin_or_owner"

# Manually Trigger an Update of a Secondary Zone
# POST  /v2/zones/{zone_id}/tasks/xfr
#"xfr_zone": "rule:admin_or_owner"

# Abandon Zone
```

(continues on next page)

(continued from previous page)

```
# POST  /v2/zones/{zone_id}/tasks/abandon
#"abandon_zone": "rule:admin"

#"count_zones": "rule:admin_or_owner"

#"count_zones_pending_notify": "rule:admin_or_owner"

#"purge_zones": "rule:admin"

#"touch_zone": "rule:admin_or_owner"

# Retrive a Zone Export from the Designate Datastore
# GET  /v2/zones/tasks/exports/{zone_export_id}/export
#"zone_export": "rule:admin_or_owner"

# Create Zone Export
# POST  /v2/zones/{zone_id}/tasks/export
#"create_zone_export": "rule:admin_or_owner"

# List Zone Exports
# GET  /v2/zones/tasks/exports
#"find_zone_exports": "rule:admin_or_owner"

# Get Zone Exports
# GET  /v2/zones/tasks/exports/{zone_export_id}
# GET  /v2/zones/tasks/exports/{zone_export_id}/export
#"get_zone_export": "rule:admin_or_owner"

# Update Zone Exports
# POST  /v2/zones/{zone_id}/tasks/export
#"update_zone_export": "rule:admin_or_owner"

# Create Zone Import
# POST  /v2/zones/tasks/imports
#"create_zone_import": "rule:admin_or_owner"

# List all Zone Imports
# GET  /v2/zones/tasks/imports
#"find_zone_imports": "rule:admin_or_owner"

# Get Zone Imports
# GET  /v2/zones/tasks/imports/{zone_import_id}
#"get_zone_import": "rule:admin_or_owner"

# Update Zone Imports
# POST  /v2/zones/tasks/imports
#"update_zone_import": "rule:admin_or_owner"

# Delete a Zone Import
# GET  /v2/zones/tasks/imports/{zone_import_id}
#"delete_zone_import": "rule:admin_or_owner"

# Create Zone Transfer Accept
# POST  /v2/zones/tasks/transfer_accepts
#"create_zone_transfer_accept": "rule:admin_or_owner OR tenant:%(target_
↪tenant_id)s OR None:%(target_tenant_id)s"
```

(continues on next page)

(continued from previous page)

```
# Get Zone Transfer Accept
# GET  /v2/zones/tasks/transfer_requests/{zone_transfer_accept_id}
#"get_zone_transfer_accept": "rule:admin_or_owner"

# List Zone Transfer Accepts
# GET  /v2/zones/tasks/transfer_accepts
#"find_zone_transfer_accepts": "rule:admin"

#"find_zone_transfer_accept": "rule:admin"

# Update a Zone Transfer Accept
# POST  /v2/zones/tasks/transfer_accepts
#"update_zone_transfer_accept": "rule:admin"

#"delete_zone_transfer_accept": "rule:admin"

# Create Zone Transfer Accept
# POST  /v2/zones/{zone_id}/tasks/transfer_requests
#"create_zone_transfer_request": "rule:admin_or_owner"

# Show a Zone Transfer Request
# GET  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
# PATCH  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
#"get_zone_transfer_request": "rule:admin_or_owner OR tenant:%(target_
↪tenant_id)s OR None:%(target_tenant_id)s"

#"get_zone_transfer_request_detailed": "rule:admin_or_owner"

# List Zone Transfer Requests
# GET  /v2/zones/tasks/transfer_requests
#"find_zone_transfer_requests": "@"

#"find_zone_transfer_request": "@"

# Update a Zone Transfer Request
# PATCH  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
#"update_zone_transfer_request": "rule:admin_or_owner"

# Delete a Zone Transfer Request
# DELETE  /v2/zones/tasks/transfer_requests/{zone_transfer_request_id}
#"delete_zone_transfer_request": "rule:admin_or_owner"
```

designate.conf

Please refer to the online version of this documentation for a full config file example.

1.4.16 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

Backends - Summary

Backend	Status	Type	In Tree	Notes
Bind9	Integrated	xfr	✓	None
Power DNS 4	Integrated	xfr	✓	None
Agent	Untested	xfr	✓	None
Akamai DNS v2	Untested	xfr	✓	None
Bind9 (Agent)	Untested	agent	✓	None
Denominator	Untested	agent	✓	None
Designate to Designate	Untested	xfr	✓	None
DynECT	Untested	xfr	✓	None
Infoblox (XFR)	Untested	xfr	✓	None
Microsoft DNS (Agent)	Untested	agent	✓	None
NSD4	Untested	xfr	✓	None
Akamai eDNS	Known Broken	xfr	✓	Akamai has turned off the eDNS API - see https://community.akamai.com/customers/s/article/Big-Changes-Coming-to-Fast-DNS-in-2018
Djbdns (Agent)	Experimental	agent	✓	None
Gdnsd (Agent)	Experimental	agent	✓	None
Knot2 (Agent)	Experimental	agent	✓	None

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	Known Broken
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

Agent

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Bind9 (Agent)

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Denominator

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Knot2 (Agent)

Grade	Experimental
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Djbdns (Agent)

Grade	Experimental
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Gdnsd (Agent)

Grade	Experimental
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

Microsoft DNS (Agent)

Grade	Untested
In Tree	✓
Maintainers	Designate Team
Repository	Designate Repository
Notes	None

1.5 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

1.6 Command-Line Interface Reference

Information on the commands available through Designates Command Line Interface (CLI) can be found in this section.

1.6.1 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

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 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

1.6.2 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.

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

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z