Promgen Documentation

Release 0.51.dev

Paul Traylor

Apr 02, 2021

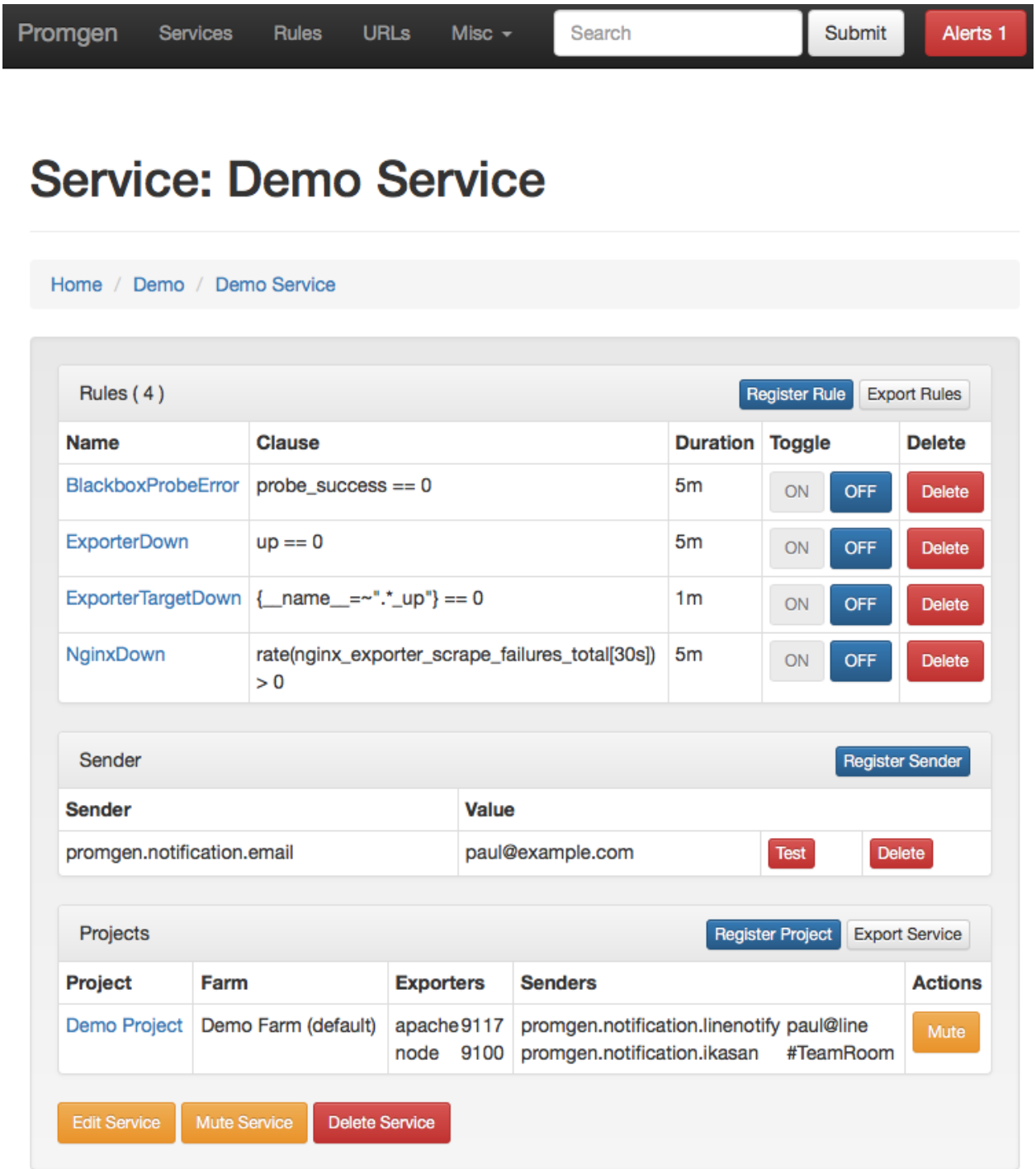
Contents

1 Overview **3**
- 1.1 Installing Promgen . . . 3
- 1.2 Using Promgen . . . 9
- 1.3 Extending Promgen . . . 13
- 1.4 Module Reference . . . 15

Python Module Index **21**

HTTP Routing Table **23**

Index **25**


Promgen
Services
Rules
URLs
Misc
Search
Submit
Alerts 1
Service: Demo Service
Home / Demo / Demo Service
Rules (4)
Register Rule
Export Rules
Name
Clause
Duration
Toggle
Delete
BlackboxProbeError
probe_success == 0
5m
ON
OFF
Delete
ExporterDown
up == 0
5m
ON
OFF
Delete
ExporterTargetDown
{__name__=~".*_up"} == 0
1m
ON
OFF
Delete
NginxDown
rate(nginx_exporter_scrape_failures_total[30s]) > 0
5m
ON
OFF
Delete
Sender
Register Sender
Sender
Value
promgen.notification.email
paul@example.com
Test
Delete
Projects
Register Project
Export Service
Project
Farm
Exporters
Senders
Actions
Demo Project
Demo Farm (default)
apache 9117
node 9100
promgen.notification.linenotify paul@line
promgen.notification.ikasan #TeamRoom
Mute
Edit Service
Mute Service
Delete Service

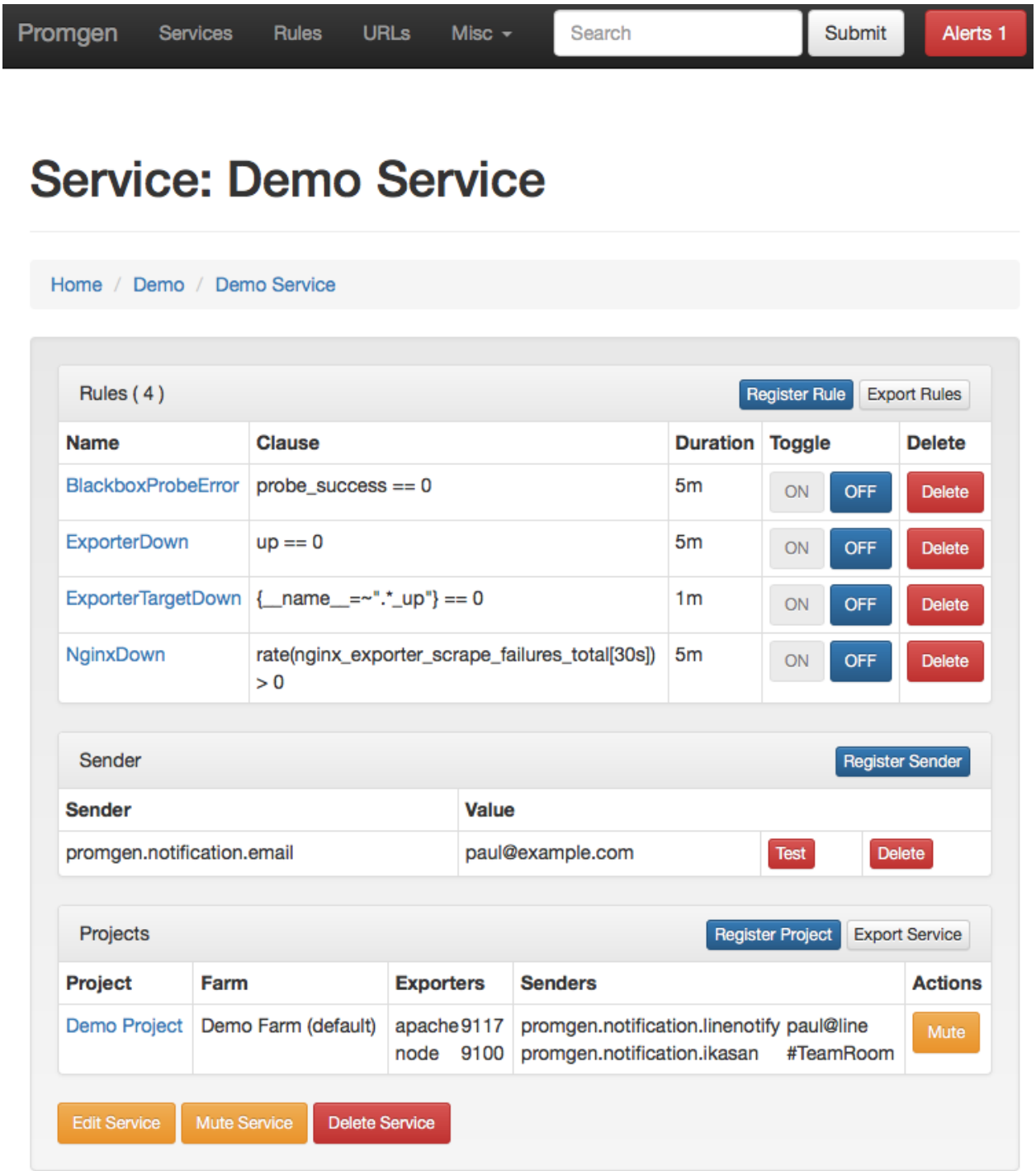
CHAPTER 1

Overview

1.1 Installing Promgen

1.1.1 Management UI

The primary management UI is a Django application and many of the concepts that apply to a typical django application will apply to Promgen.



Running under uwsgi

Promgen can run under uwsgi using a similar configuration

```
[uwsgi]
module = promgen.wsgi
virtualenv = /path/to/promgen/virtualenv
```

(continues on next page)

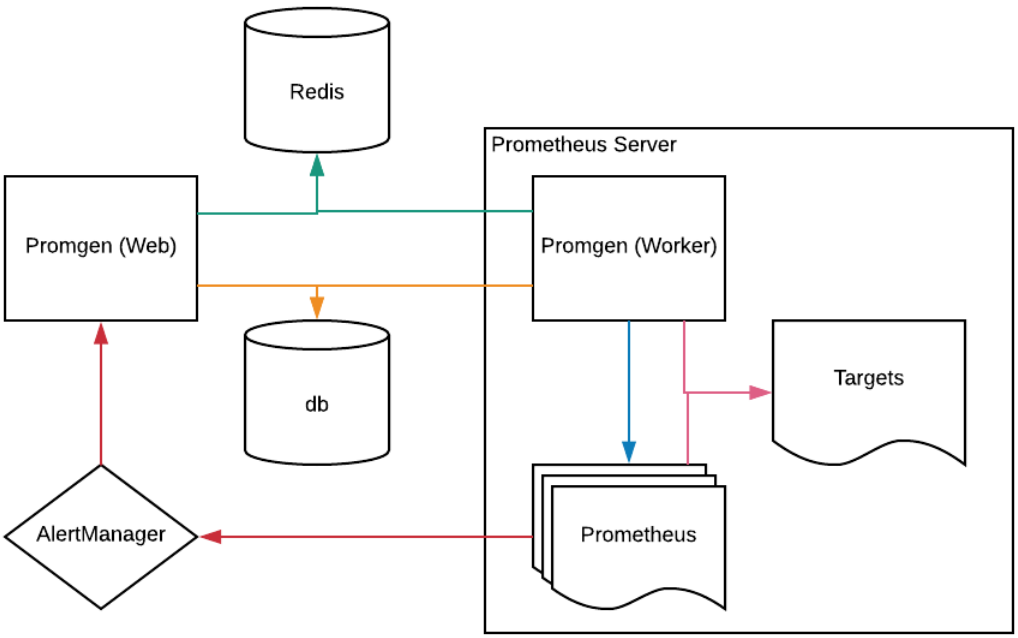
(continued from previous page)

```
plugins = python3

http-socket = :12345
# Or using a socket
# socket = /run/uwsgi/promgen.sock
master = true
processes = 5
threads = 5
buffer-size = 8192
```

1.1.2 Prometheus Server

One of Promgen's primary roles is to manage a list of targets for Prometheus to scrape. For high availability, it is generally prefered to have multiple Prometheus servers running together. There are multiple ways to deploy these targets to a Prometheus server.

Worker Model (Push)



Promgen's Push mode relies on celery to push updates to Prometheus. A Promgen worker is run on each Prometheus server which subscribes to a named queue to signal when to write out an updated configuration file, and update Prometheus.

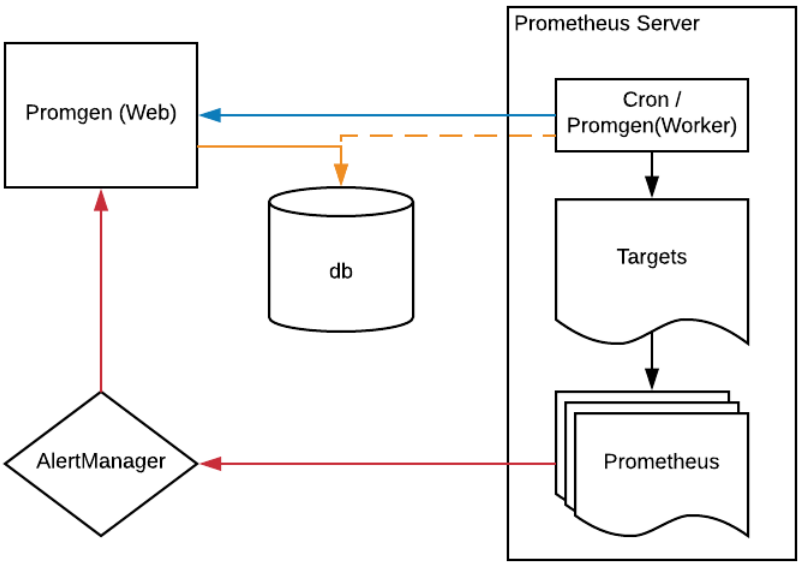
```
# Assuming we have a Prometheus shard named promshard and two servers we
# may deploy the workers like this
promgen register-server promshard prometheus001 9090
promgen register-server promshard prometheus002 9090

# Then on each Prometheus server, we would want to run a celery worker with
```

(continues on next page)

(continued from previous page)

```
# the queue name matching the name that we registered
celery -A promgen -l info --queues prometheus001
# If running within docker, the same command would look like this
docker run --rm \
    -v ~/.config/promgen:/etc/promgen/ \
    -v /etc/prometheus:/etc/prometheus \
    line/promgen worker -l info --queues prometheus001
```

Cron Model (Pull)



In some cases it is not possible (or not desired) to install Promgen beside Prometheus. In this case, Promgne's pull mode can be used by trigging a small script running from cron or any other job framework. This only requires the server where Prometheus is running, to be able to access Promgen over HTTP.

```
#!/bin/sh
set -e
# Download all the targets from Promgen to a temporary file
curl http://promgen/api/v1/targets --output /etc/prometheus/targets.tmp
# Optionally you could download from a specific service or project
# curl http://promgen/service/123/targets -o /etc/prometheus/targets.tmp
# curl http://promgen/project/456/targets -o /etc/prometheus/targets.tmp
# Move our file to make it more atomic
mv /etc/prometheus/targets.tmp /etc/prometheus/targets.json
# Tell Prometheus to reload
curl -XPOST http://localhost:9090/-/reload
```

If it's possible to install Promgen, then there is a Promgen helper command that handles the same basic steps as the above command. This will however require the Prometheus server to be able to access the same database as the Promgen web instance.

```
# Internally Promgen uses an atomic write function so you can give
# it the path where you want to save it and have it --reload automatically
promgen targets /etc/prometheus/targets.json --reload
```

Filtering Targets (Both)

In both models, you will want to ensure that the Prometheus server only scrapes the correct subset of targets. Ensure that the correct rewrite_labels is configured

```
- job_name: 'promgen'
  file_sd_configs:
    - files:
      - "/etc/prometheus/promgen.json"
  relabel_configs:
  - source_labels: [__shard]
    # Our regex value here should match the shard name (exported as __shard)
    # that shows up in Promgen. In the case we want our Prometheus server to
    # scrape all targets, then we can ommit the relable config.
    regex: promshard
    action: keep
```

1.1.3 Alert Worker

Alerts that are routed through Promgen are added to a celery queue to be processed

Running under systemd

```
[Unit]
Description=Promgen Worker
After=network.target

[Service]
Type=simple
ExecStart=/path/to/virtualenv/bin/celery -A promgen worker -l info
Restart=on-failure
User=edge-dev

[Install]
WantedBy=multi-user.target
```

1.1.4 Running with Docker

```
# Configure custom settings, database settings, etc
vim /etc/promgen/settings.yml
vim /etc/promgen/CELERY_BROKER_URL
vim /etc/promgen/DATABASE_URL
vim /etc/promgen/SECRET_KEY

# Running a Promgen Web worker
docker run -d --name promgen -p 8000:8000 --network host -v /etc/promgen/:/etc/
↪promgen/ promgen:latest web
```

(continues on next page)

(continued from previous page)

```
# Running a Promgen Alert Worker
# Alert workers do not specify a queue and use the celery default queue
docker run -d --name promgen --network host -v /etc/promgen/:/etc/promgen/ -v /etc/
↪prometheus:/etc/prometheus promgen:latest worker

# Running a Promgen Celery worker to update Prometheus settings
# Assuming our Prometheus node is called prometheus-001 we should subscribe
# to a celery queue with the same name that is registered in Promgen
docker run -d --name promgen --network host -v /etc/promgen/:/etc/promgen/ -v /etc/
↪prometheus:/etc/prometheus promgen:latest worker --queues prometheus-001
```

1.1.5 Configuring Django

Arbitrary django settings can be set for the promgen web app by adding those under the `django` key to the `promgen.yml` file.

All available django settings (not every setting may apply to promgen) are listed in the django reference.

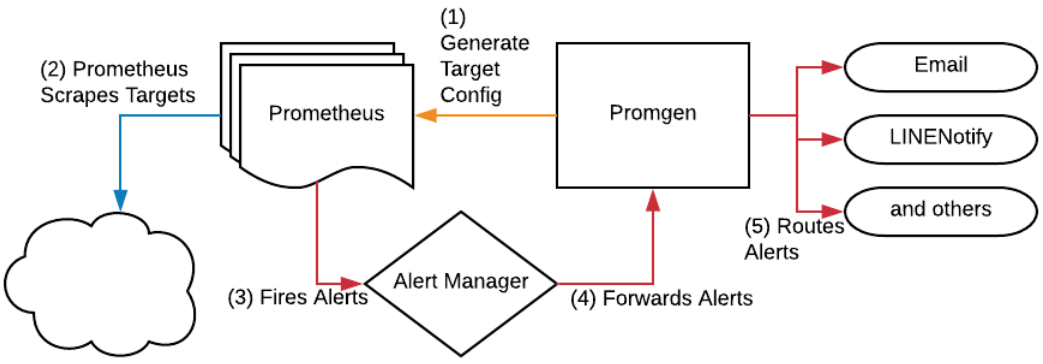
Configuring an SMTP Server

An SMTP server for sending outgoing mail can be configured this way:

```
promgen.notification.email:
  sender: promgen@example.com

django:
  EMAIL_HOST: mail.example.com
  EMAIL_PORT: 587
  EMAIL_HOST_USER: user@example.com
  EMAIL_HOST_PASSWORD: <secret password>
  EMAIL_USE_TLS: true
```

The `EMAIL_USE_TLS` and `EMAIL_USE_SSL` settings are mutually exclusive. The `EMAIL_USE_SSL` setting enables implicit TLS, the `EMAIL_USE_TLS` setting enables STARTTLS.

The django docs on email cover how emails are sent by django as well as relevant configuration parameters.



1. Promgen manages a list of Targets and Rules that it deploys to a Prometheus server

2. Prometheus will load these settings and proceed to scrape targets
3. When an alert fires, it will be sent to AlertManager
4. AlertManager will group on labels and handle de-duplication and forward to Promgen
5. Promgen will route the message based on labels to the correct notification

1.2 Using Promgen

1.2.1 Alerting Rules

Promgen supports the concept of a global set of rules and then configuring overrides at the Service or Project level

When working with many services, there are times that a global rule will not cover all use cases. In this case, Promgen allows the user to override a parent rule using an <exclude> tag.

For example, we may want to check overall file system usage across all of our machines

```
node_filesystem_free / node_filesystem_size < 0.20
```

This may be an acceptable default for our service, but perhaps a different service wants to be warned at only 10%. Since the global rule already warns us at 20% we need to do something different. Promgen supports a special <exclude> tag to handle this use case

Listing 1: Original Rules

```
node_filesystem_free{<exclude>}
    / node_filesystem_size{<exclude>} < 0.20
node_filesystem_free{service="A", <exclude>}
    / node_filesystem_size{service="A", <exclude>} < 0.10
node_filesystem_free{service="B"}
    / node_filesystem_size{service="B"} < 0.15
```

This will be properly expanded into the following rules so that we can have a general default but be more specific with certain services

Listing 2: Expanded Rules

```
node_filesystem_free{service=~"A|B"}
    / node_filesystem_size{service=~"A|B"} < 0.20
node_filesystem_free{service="A",}
    / node_filesystem_size{service="A",} < 0.10
node_filesystem_free{service="B"}
   / node_filesystem_size{service="B"} < 0.15
```

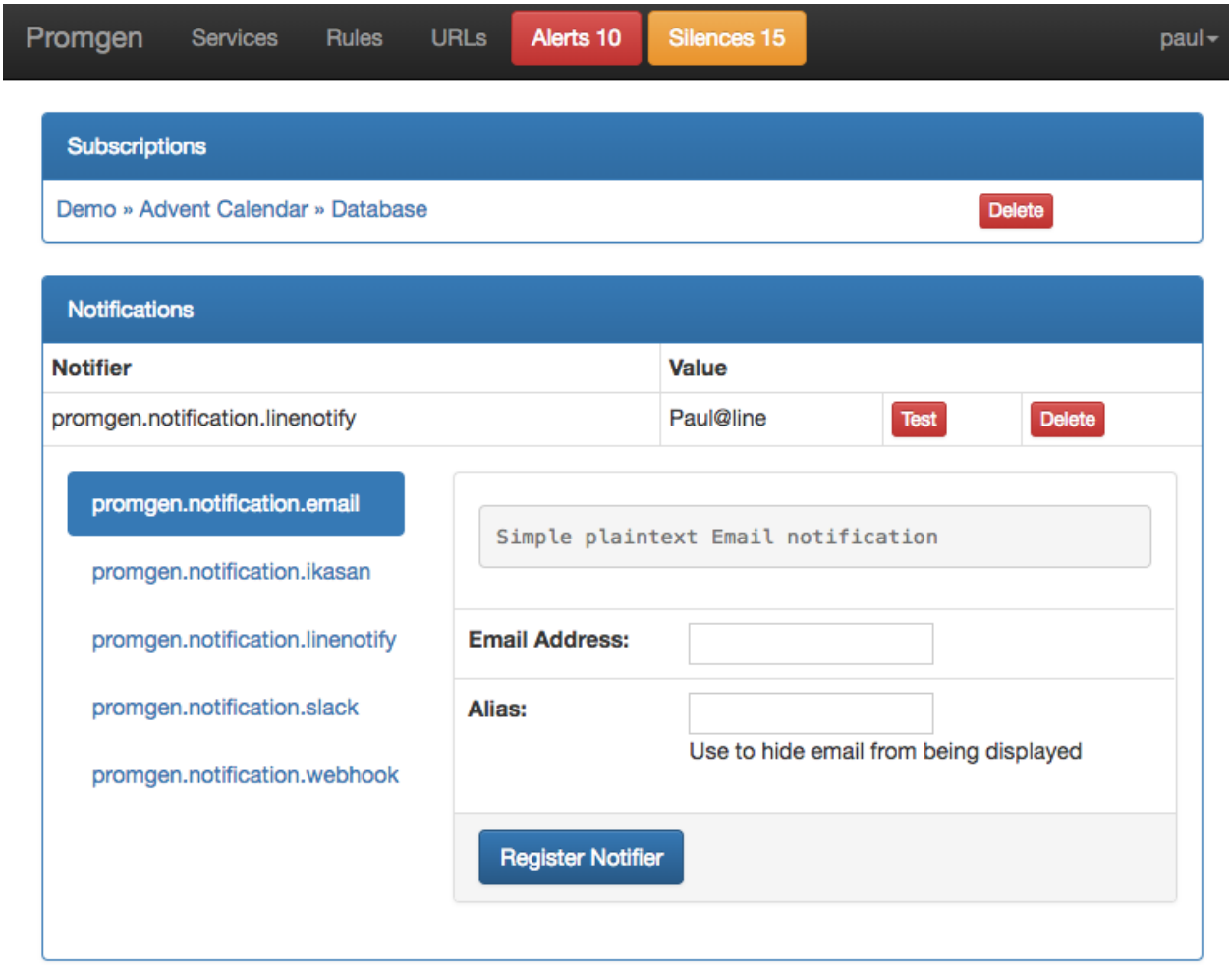
Visuallizing it as a hiarchy it would look like this

```
# Global Rule excludes children
example_rule{service!~"A|B",}:
    # Service A override includes self
    - example_rule{service="A",}
    # Service B override includes self, but excludes children
    - example_rule{service="B", project!~"C"}:
        # Project Override
        - example_rule{project="C"}
```

1.2.2 Notification Settings

Promgen allows the developer to easily subscribe to the notifications they want for the servies they use.

Subscription



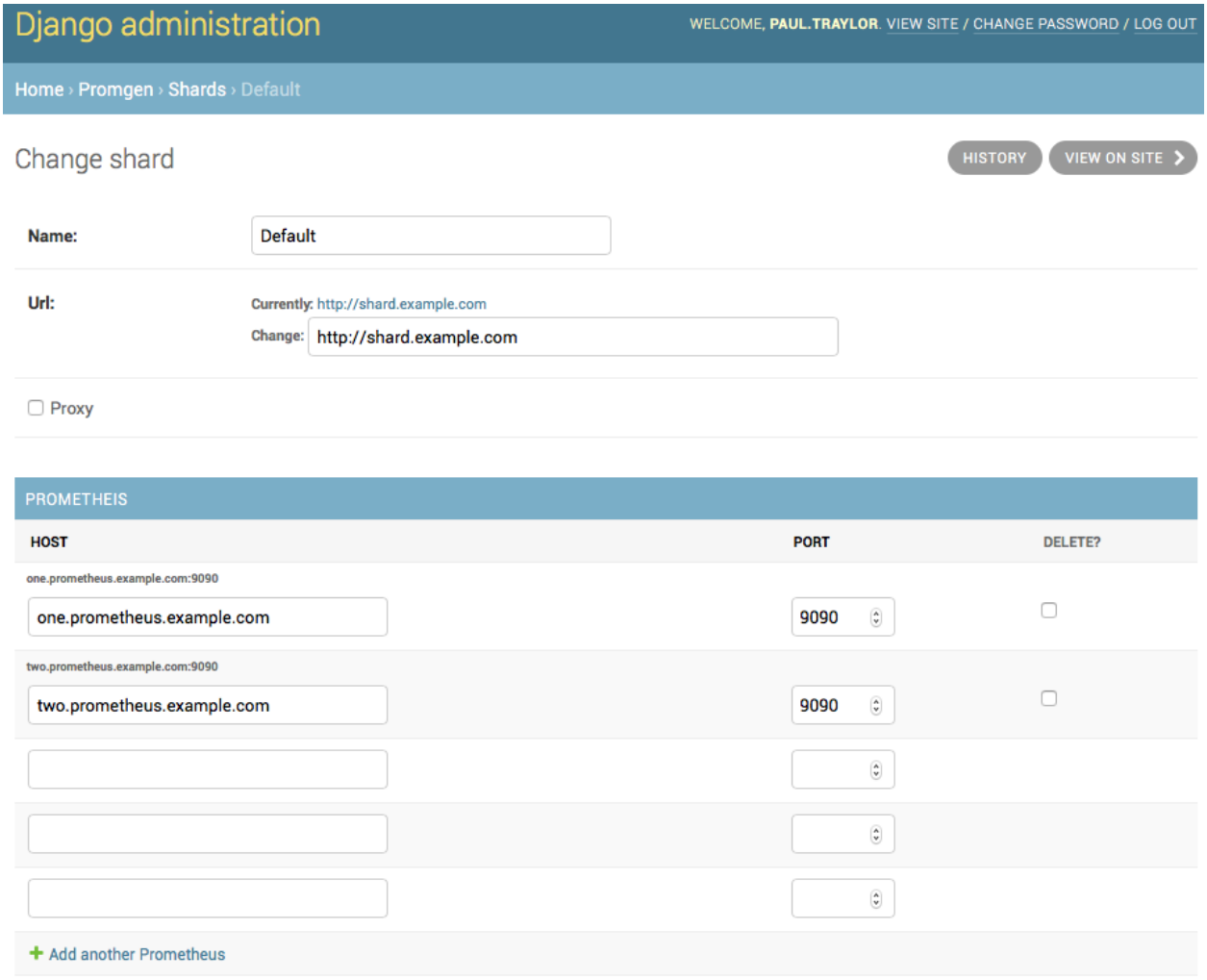
From any project or service, a developer can use the Subscribe to Notifications button to subscribe. Then from the developer's profile page, they should select the notification methods they wish to use.

Project and Service Notifications

Specific notifications can also be configured for each Project and Service.

1.2.3 Administration Tasks

Since Promgen is a standard django many tasks can be handled through Django's admin. or through management commands.



Managing Shards using CLI

```
# Register a Prometheus server running on the host prometheus002 on port 9090
# to the shard 'promshard'
promgen register-server promshard prometheus002 9090
```

1.2.4 API

Prometheus Proxy API

Promgen provides a simple proxy api to proxy certain requests across all managed Prometheus nodes. See the Prometheus API for more details

GET /api/v1/label/(**string:** *labelname*)**/values**

GET /api/v1/series

GET /api/v1/query_range

Promgen API

POST /api/v1/alerts

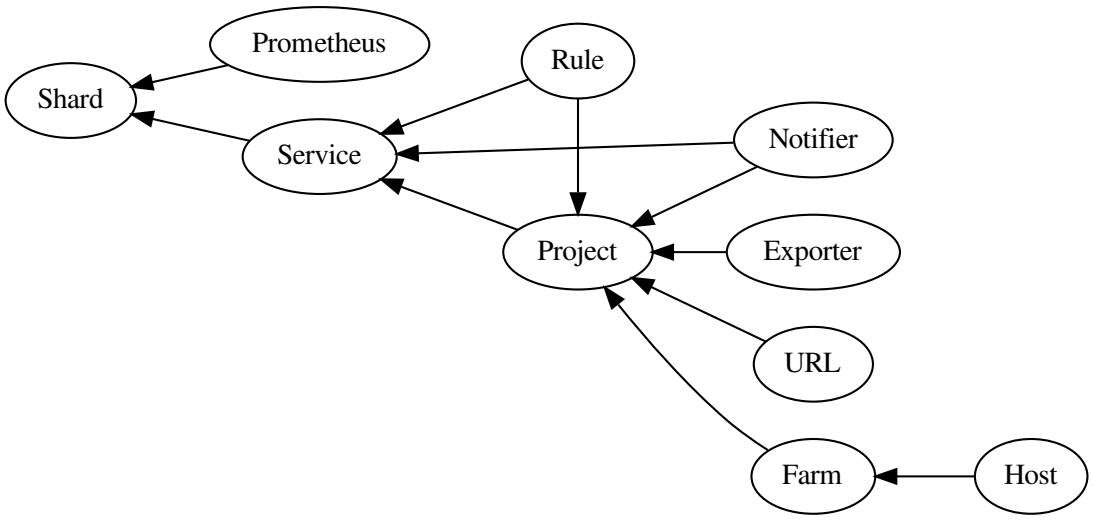
GET /api/v1/targets

GET /api/v1/rules

GET /api/v1/urls

GET /api/v1/host/(**string:** *hostname*)

1.2.5 Glossary



Shard

Shards are the top level object in Promgen which contain both Services and Prometheus servers

Service

A service in Prometheus is a group of related Projects and are assigned to a specific shard

Projects

Projects are one of the main groupings for Promgen. This represents a typical monitoring target and brings together a collection of servers (Farm) with Exporters and URL endpoints

Farm

Farm is a group of servers. Farms may be updated using *discovery plugins*

Notifiers

Notifiers are used for routing messages to a specific destination such as Email or LINE Notify

Rule

Rules represent the rules that Prometheus itself uses for alerting

1.3 Extending Promgen

1.3.1 Authentication Plugins

Promgen uses auth plugins based on Django's auth backend and the Python Social Auth. They are registered in Promgen's settings file

```
# Add keys and secrets to promgen.yml
# Make sure you add Django's ModelBackend if you want to allow users to
# login with a Django user
django:
  AUTHENTICATION_BACKENDS:
  - module.path.auth.ExampleAuth
  - social_core.backends.github.GithubOAuth2
  - django.contrib.auth.backends.ModelBackend
  SOCIAL_AUTH_EXAMPLE_KEY: foo
  SOCIAL_AUTH_EXAMPLE_SECRET: bar
  SOCIAL_AUTH_GITHUB_KEY: a1b2c3d4
  SOCIAL_AUTH_GITHUB_SECRET: e5f6g7h8i9
```

1.3.2 Discovery Plugins

Promgen uses discovery plugins to bridge non-natively supported discovery mechanisms to Prometheus's file configuration format. They should be registered using setuptools entry_points.

```
entry_points={
    'promgen.discovery': [
        'example = module.path.discovery:DiscoveryExample',
    ],
}
```

Plugins should inherit from DiscoveryBase, should implement *farms()* and *fetch()* methods

```
from promgen.discovery import DiscoveryBase

EXAMPLE = {
  'Farm-A': ['AA', 'AB', 'AC']
  'Farm-B': ['BA', 'BB', 'BC', 'BD']
}

class DiscoveryExample(DiscoveryBase):
  def fetch(self, farm_name):
      return EXAMPLE[farm_name]
```

(continues on next page)

(continued from previous page)

```
  def farms(self):
      return EXAMPLE.keys()
```

1.3.3 Notification Plugins

Promgen uses notifier plugins to route notifications to different services such as Email or LINE Notify. Sender plugins are registered using setuptools entry_points.

```
entry_points={
    'promgen.notification': [
        'sender_name = module.path.notification:NotificationExample',
    ],
}
```

Plugins should inherit from SenderBase, and at a minimum impelement a *_send()* method

```
from promgen.celery import app as celery
from promgen.notification import NotificationBase

class NotificationExample(NotificationBase):
    @celery.task(bind=True)
    def _send(task, target, alert, data):
        ## Code specific to sender
        print(target)
        print(alert)
        return True
```

Notes about Celery

Because of the way Celery works, when you wrap a method call with `@celery.task`, you lose access to the self instance of the Sender class. If you use `@celery.task(bind=True)` then you can get an instance of the task. If you need to have an instance of the class, you can use this to get an instance of the class

```
from promgen.celery import app as celery
from promgen.notification import NotificationBase

class SenderCeleryExample(NotificationBase):
    @celery.task(bind=True)
    def _send(task, target, alert, data):
        self = task.__klass__()
        ## Code specific to sender
        print(target)
        print(alert)
        return True
# Set an instance we can retrive from within our _send method
SenderCeleryExample._send.__klass__ = SenderCeleryExample
```

1.4 Module Reference

1.4.1 Discovery Plugins

class promgen.discovery.default.**DiscoveryPromgen**
Promgen local database discovery plugin

This is the default discovery plugin for farms and hosts stored locally in promgen's database. They are queried directly from Django's ORM

farms()
Fetch farms from local database

fetch(*farm_name*)
Fetch list of hosts for a farm from the local database

1.4.2 Notification Modules

class promgen.notification.**FormSenderBase**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

class promgen.notification.**NotificationBase**
Base Notification class

config(*key*)
Plugin specific configuration

This wraps our PROMGEN settings so that a plugin author does not need to be concerned with how the configuration files are handled but only about the specific key.

form
alias of *FormSenderBase*

class promgen.notification.email.**FormEmail**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

class promgen.notification.email.**NotificationEmail**
Simple plaintext Email notification

form
alias of *FormEmail*

class promgen.notification.linenotify.**FormLineNotify**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

class promgen.notification.linenotify.**NotificationLineNotify**
Send messages to line notify

https://notify-bot.line.me/en/

form
alias of *FormLineNotify*

Simple webhook bridge Accepts alert json from Alert Manager and then POSTs individual alerts to configured webhook destinations

class promgen.notification.webhook.**FormWebhook**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *field_order=None*, *use_required_attribute=None*, *renderer=None*)

class promgen.notification.webhook.**NotificationWebhook**
Post notifications to a specific web endpoint

form
alias of *FormWebhook*

1.4.3 Prometheus Interactions

promgen.prometheus.**check_rules**(*rules*)
Use promtool to check to see if a rule is valid or not

The command name changed slightly from 1.x -> 2.x but this uses promtool to verify if the rules are correct or not. This can be bypassed by setting a dummy command such as /usr/bin/true that always returns true

promgen.prometheus.**import_config**(*config*, *replace_shard=None*)

promgen.prometheus.**import_rules_v2**(*config*, *content_object=None*)
Loop through a dictionary and add rules to the database

This assumes a dictonary in the 2.x rule format. See promgen/tests/examples/import.rule.yml for an example

promgen.prometheus.**render_config**(*service=None*, *project=None*)

promgen.prometheus.**render_rules**(*rules=None*)
Render rules in a format that Prometheus understands

Parameters

- **rules** (*list(Rule)*) – List of rules

- **version** (*int*) – Prometheus rule format (1 or 2)

Returns Returns rules in yaml or Prometheus v1 format

Return type bytes

This function can render in either v1 or v2 format We call prefetch_related_objects within this function to populate the other related objects that are mostly used for the sub lookups.

promgen.prometheus.**render_urls**()

promgen.prometheus.**silence**(*labels*, *duration=None*, ***kwargs*)
Post a silence message to Alert Manager Duration should be sent in a format like 1m 2h 1d etc

1.4.4 Template Tags

promgen.templatetags.promgen.**breadcrumb**(*instance=None*, *label=None*)
Create HTML Breadcrumb from instance

Starting with the instance, walk up the tree building a bootstrap3 compatiable breadcrumb

promgen.templatetags.promgen.**diff_json**(*a*, *b*)

promgen.templatetags.promgen.**klass**(*value*)

promgen.templatetags.promgen.**pretty_json**(*data*)

promgen.templatetags.promgen.**pretty_yaml**(*data*)

promgen.templatetags.promgen.**qs_replace**(*context*, *k*, *v*)
Query string handler for paginators

Assuming we have a query string like ?page=1&search=foo, there are several cases in which we want to replace only the page key, while leaving the rest alone. This tag allows us to replace individual values (like the current page) while carrying over other values (like a search string)

Example: {% qs_replace 'page' page_obj.next_page_number %}

promgen.templatetags.promgen.**qsfilter**(*request*, *k*, *v*)
Helper to rewrite query string for URLs

{% qsfilter request 'foo' 'baz' %} When passed the request object, it will take a querystring like ?foo=bar&donottouch=1 and change it to ?foo=baz&donottouch=1

Useful when working with filtering on a page that also uses pagination to avoid losing other query strings {% qsfilter request 'page' page_obj.previous_page_number %}

promgen.templatetags.promgen.**rule_dict**(*rule*)

promgen.templatetags.promgen.**rulemacro**(*rule*, *clause=None*)
Macro rule expansion

Assuming a list of rules with children and parents, expand our macro to exclude child rules

Can optionally pass expression to render in the context of the passed rule

```
foo{<exclude>} / bar{<exclude>} > 5 # Parent Rule
foo{project="A", <exclude>} / bar{project="A", <exclude>} > 3 # Child Rule
foo{project="B"} / bar{project="B"} > 4 # Child Rule

foo{project~="A|B"} / bar{project~="A|B"} > 5
foo{project="A", } / bar{project="A"} > 3
foo{project="B"} / bar{project="B"} > 4
```

promgen.templatetags.promgen.**strftime**(*timestamp*, *fmt*)

1.4.5 Promgen Models

class promgen.models.**Alert**(*id*, *created*, *body*, *sent_count*, *error_count*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**AlertError**(*id*, *alert*, *created*, *message*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**AlertLabel**(*id*, *alert*, *name*, *value*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Audit**(*id*, *body*, *created*, *data*, *old*, *content_type*, *object_id*, *user*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**DefaultExporter**(*id*, *job*, *port*, *path*, *scheme*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Exporter**(*id*, *job*, *port*, *path*, *scheme*, *project*, *enabled*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Farm**(*id*, *name*, *source*)

exception DoesNotExist

exception MultipleObjectsReturned

driver
Return configured driver for Farm model instance

classmethod driver_set()
Return the list of drivers for Farm model

class promgen.models.**Filter**(*id*, *sender*, *name*, *value*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Host**(*id*, *name*, *farm*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Probe**(*id*, *module*, *description*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Project**(*id*, *name*, *description*, *owner*, *service*, *shard*, *farm*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Prometheus**(*id*, *shard*, *host*, *port*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Rule**(*id*, *name*, *clause*, *duration*, *enabled*, *parent*, *content_type*, *object_id*, *description*)

exception DoesNotExist

exception MultipleObjectsReturned

copy_to(*content_type*, *object_id*)
Make a copy under a new service

It's important that we set pk to None so a new object is created, but we also need to ensure the new name is unique by appending some unique data to the end of the name

class promgen.models.**RuleAnnotation**(*id*, *name*, *value*, *rule*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**RuleLabel**(*id*, *name*, *value*, *rule*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Sender**(*id*, *sender*, *value*, *alias*, *content_type*, *object_id*, *owner*, *enabled*)

exception DoesNotExist

exception MultipleObjectsReturned

driver
Return configured driver for Sender model instance

classmethod driver_set()
Return the list of drivers for Sender model

filtered(*alert*)
Check filters for a specific sender

If no filters are defined, then we let the message through If filters are defined, then we check to see if at least one filter matches If no filters match, then we assume it's filtered out

test()
Test sender plugin

Uses the same test json from our unittests but subs in the currently tested object as part of the test data

class promgen.models.**Service**(*id*, *name*, *description*, *owner*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Shard**(*id*, *name*, *url*, *proxy*, *enabled*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**Site**(*id*, *domain*, *name*)

exception DoesNotExist

exception MultipleObjectsReturned

class promgen.models.**URL**(*id*, *url*, *project*, *probe*)

exception DoesNotExist

exception MultipleObjectsReturned

Python Module Index

p

promgen.discovery.default, 15
promgen.models, 18
promgen.notification, 15
promgen.notification.email, 15
promgen.notification.linenotify, 15
promgen.notification.webhook, 16
promgen.prometheus, 16
promgen.templatetags.promgen, 17

HTTP Routing Table

/api

GET /api/v1/host/(string:hostname), 12
GET /api/v1/label/(string:labelname)/values, 11
GET /api/v1/query_range, 11
GET /api/v1/rules, 12
GET /api/v1/series, 11
GET /api/v1/targets, 12
GET /api/v1/urls, 12
POST /api/v1/alerts, 12

Index

A

Alert (*class in promgen.models*), 18
Alert.DoesNotExist, 18
Alert.MultipleObjectsReturned, 18
AlertError (*class in promgen.models*), 18
AlertError.DoesNotExist, 18
AlertError.MultipleObjectsReturned, 18
AlertLabel (*class in promgen.models*), 18
AlertLabel.DoesNotExist, 18
AlertLabel.MultipleObjectsReturned, 18
Audit (*class in promgen.models*), 18
Audit.DoesNotExist, 18
Audit.MultipleObjectsReturned, 18

B

breadcrumb() (*in module promgen.templatetags.promgen*), 17

C

check_rules() (*in module promgen.prometheus*), 16
config() (*promgen.notification.NotificationBase method*), 15
copy_to() (*promgen.models.Rule method*), 19

D

DefaultExporter (*class in promgen.models*), 18
DefaultExporter.DoesNotExist, 18
DefaultExporter.MultipleObjectsReturned, 18
diff_json() (*in module promgen.templatetags.promgen*), 17
DiscoveryPromgen (*class in promgen.discovery.default*), 15
driver (*promgen.models.Farm attribute*), 18
driver (*promgen.models.Sender attribute*), 19
driver_set() (*promgen.models.Farm class method*), 18
driver_set() (*promgen.models.Sender class method*), 19

E

Exporter (*class in promgen.models*), 18
Exporter.DoesNotExist, 18
Exporter.MultipleObjectsReturned, 18

F

Farm (*class in promgen.models*), 18
Farm.DoesNotExist, 18
Farm.MultipleObjectsReturned, 18
farms() (*promgen.discovery.default.DiscoveryPromgen method*), 15
fetch() (*promgen.discovery.default.DiscoveryPromgen method*), 15
Filter (*class in promgen.models*), 18
Filter.DoesNotExist, 18
Filter.MultipleObjectsReturned, 18
filtered() (*promgen.models.Sender method*), 19
form (*promgen.notification.email.NotificationEmail attribute*), 15
form (*promgen.notification.linenotify.NotificationLineNotify attribute*), 16
form (*promgen.notification.NotificationBase attribute*), 15
form (*promgen.notification.webhook.NotificationWebhook attribute*), 16
FormEmail (*class in promgen.notification.email*), 15
FormLineNotify (*class in promgen.notification.linenotify*), 15
FormSenderBase (*class in promgen.notification*), 15
FormWebhook (*class in promgen.notification.webhook*), 16

H

Host (*class in promgen.models*), 18
Host.DoesNotExist, 19
Host.MultipleObjectsReturned, 19

I

import_config() (*in module promgen.prometheus*), 16

import_rules_v2() (*in module promgen.prometheus*), 16

K

klass() (*in module promgen.templatetags.promgen*), 17

N

NotificationBase (*class in promgen.notification*), 15
NotificationEmail (*class in promgen.notification.email*), 15
NotificationLineNotify (*class in promgen.notification.linenotify*), 16
NotificationWebhook (*class in promgen.notification.webhook*), 16

P

pretty_json() (*in module promgen.templatetags.promgen*), 17
pretty_yaml() (*in module promgen.templatetags.promgen*), 17
Probe (*class in promgen.models*), 19
Probe.DoesNotExist, 19
Probe.MultipleObjectsReturned, 19
Project (*class in promgen.models*), 19
Project.DoesNotExist, 19
Project.MultipleObjectsReturned, 19
Prometheus (*class in promgen.models*), 19
Prometheus.DoesNotExist, 19
Prometheus.MultipleObjectsReturned, 19
promgen.discovery.default (*module*), 15
promgen.models (*module*), 18
promgen.notification (*module*), 15
promgen.notification.email (*module*), 15
promgen.notification.linenotify (*module*), 15
promgen.notification.webhook (*module*), 16
promgen.prometheus (*module*), 16
promgen.templatetags.promgen (*module*), 17

Q

qs_replace() (*in module promgen.templatetags.promgen*), 17
qsfilter() (*in module promgen.templatetags.promgen*), 17

R

render_config() (*in module promgen.prometheus*), 16
render_rules() (*in module promgen.prometheus*), 16
render_urls() (*in module promgen.prometheus*), 17
Rule (*class in promgen.models*), 19
Rule.DoesNotExist, 19
Rule.MultipleObjectsReturned, 19
rule_dict() (*in module promgen.templatetags.promgen*), 17
RuleAnnotation (*class in promgen.models*), 19
RuleAnnotation.DoesNotExist, 19
RuleAnnotation.MultipleObjectsReturned, 19
RuleLabel (*class in promgen.models*), 19
RuleLabel.DoesNotExist, 19
RuleLabel.MultipleObjectsReturned, 19
rulemacro() (*in module promgen.templatetags.promgen*), 17

S

Sender (*class in promgen.models*), 19
Sender.DoesNotExist, 19
Sender.MultipleObjectsReturned, 19
Service (*class in promgen.models*), 20
Service.DoesNotExist, 20
Service.MultipleObjectsReturned, 20
Shard (*class in promgen.models*), 20
Shard.DoesNotExist, 20
Shard.MultipleObjectsReturned, 20
silence() (*in module promgen.prometheus*), 17
Site (*class in promgen.models*), 20
Site.DoesNotExist, 20
Site.MultipleObjectsReturned, 20
strftime() (*in module promgen.templatetags.promgen*), 17

T

test() (*promgen.models.Sender method*), 20

U

URL (*class in promgen.models*), 20
URL.DoesNotExist, 20
URL.MultipleObjectsReturned, 20