K2hR3 OpenStack Notification Listener Documentation

Release 0.0.14

Hirotaka Wakabayashi

Feb 26, 2019

Contents:

CHAPTER 1

K2hR3 OpenStack Notification Listener

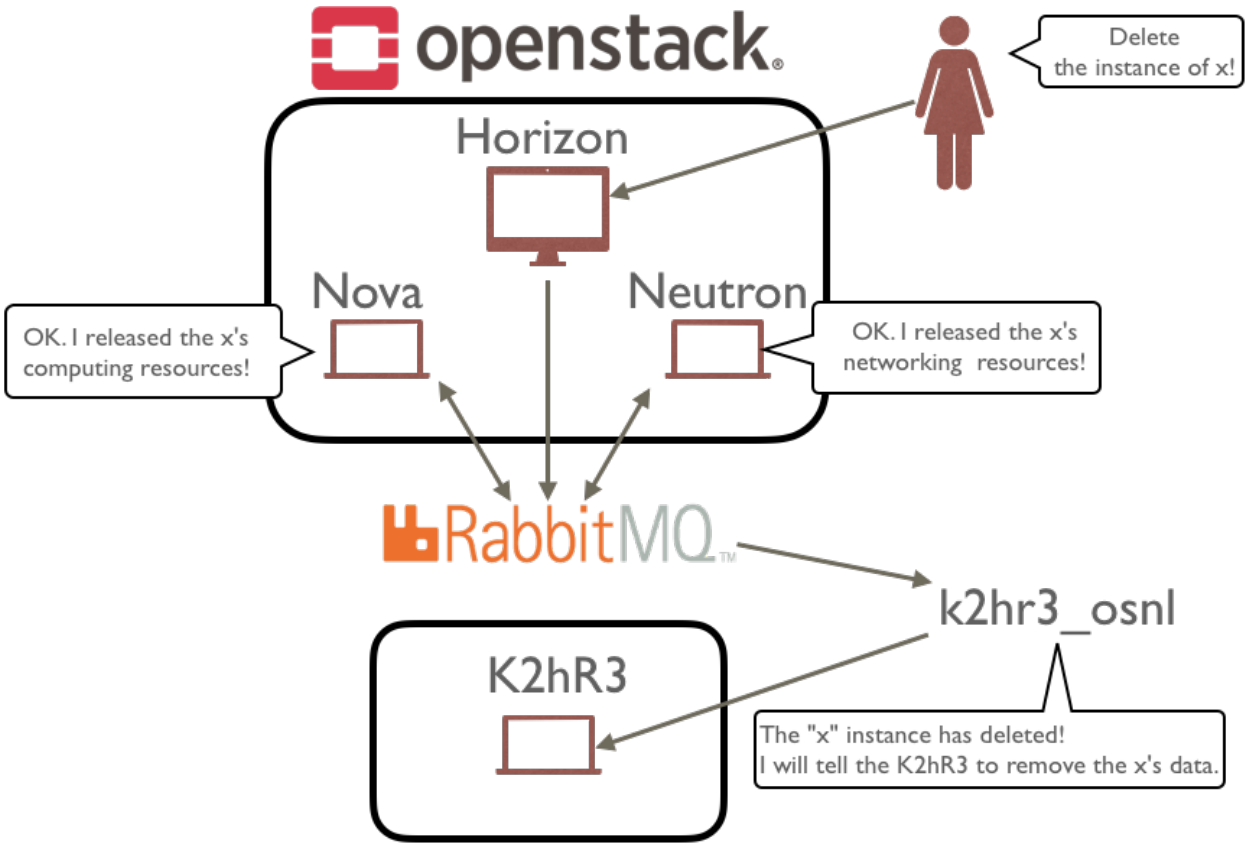
An OpenStack notification listener for the K2hR3 role-based ACL system developed in Yahoo Japan Corporation.

1.1 Overview

k2hr3_osnl is **K2hR3 O** pen **S** tack **N** otification **L** istener. It is a part of the K2hR3 system, which is a role-based ACL system developed in Yahoo Japan Corporation.

k2hr3_osnl is an OpenStack Notification Listener that listens to notifications from OpenStack services. OpenStack services emit notifications to the message bus, which is provided by oslo.messaging. oslo.messaging transports notification messages to a message broker server. The default broker server is RabbitMQ. When **k2hr3_osnl** catches a notification message from RabbitMQ, it sends the payload to the K2hR3 that is a role-based ACL system that provides access control for Openstack virtual machine instances. Figure 1 shows the relationship between the components.

Fig.1: overview



1.2 Document

https://github.com/hiwakaba/k2hr3_osnl/wiki

1.3 K2hR3

K2hR3 is a role-based ACL system developed in Yahoo Japan Corporation.

1.4 License

MIT License

1.5 AntPickax

k2hr3_osnl is an AntPickax products, which is an open source project by Yahoo Japan Corporation.

1.6 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install K2hR3 OpenStack Notification Listener, run this command in your terminal:

```
$ sudo pip install k2hr3_osnl
```

This is the preferred method to install K2hR3 OpenStack Notification Listener, as it will always install the most recent stable release.

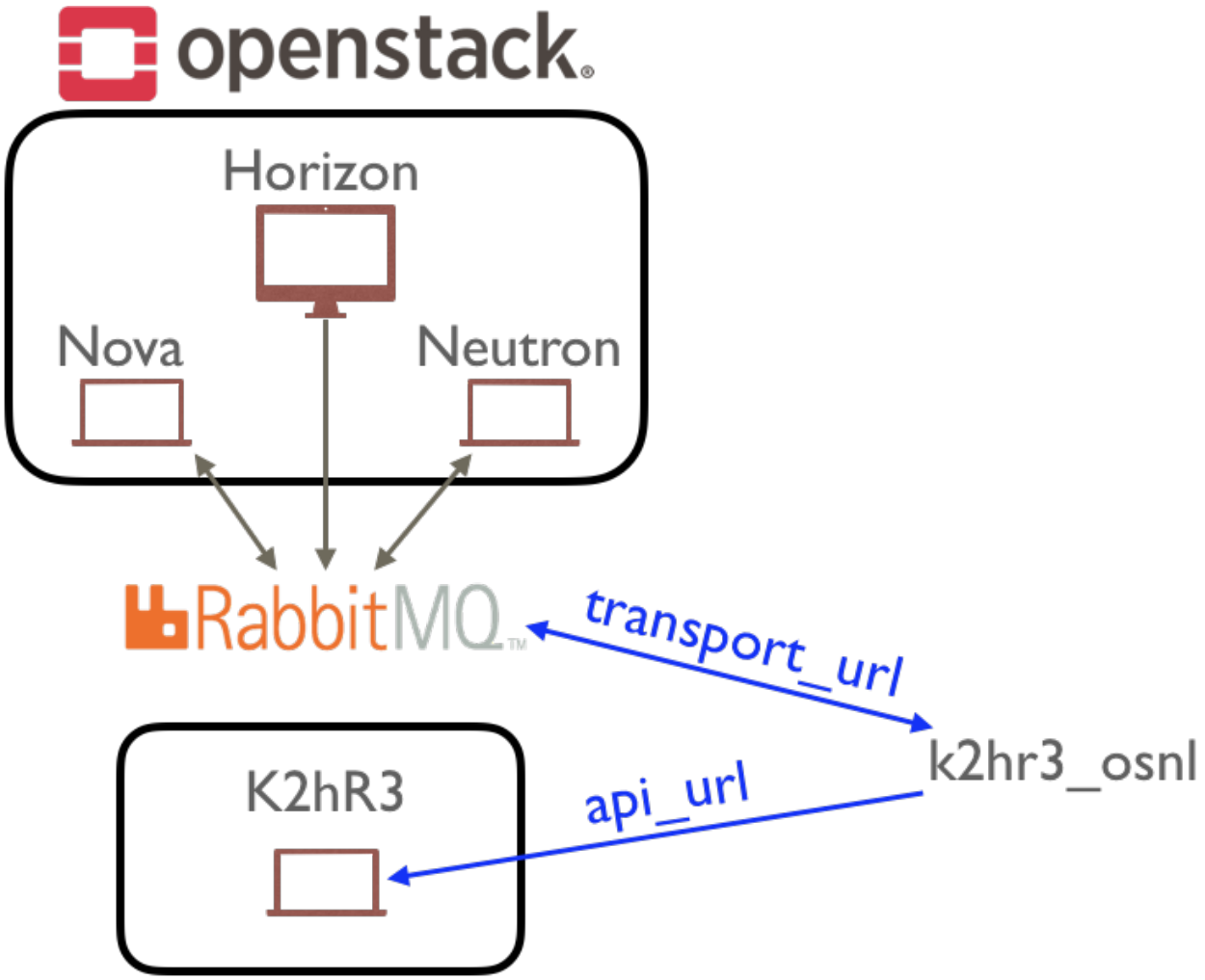
If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 Configuration

There are two primary configurations in `k2hr3_onsl.conf`:

- **`transport_url`** The message queue server location with an username and a password.
- **`api_url`** The k2hr3 WebApi server location.

The following figure can help you to understand the transport_url and api_url:



The `k2hr3_osnl.conf` path depends on the pip command shipped in your OS. You can get the path by two commands.

```
$ sudo pip3 show -f k2hr3_osnl
Name: k2hr3-osnl
...
Location: /usr/local/lib/python3.5/dist-packages
Requires: oslo.messaging, oslo.config
Files:
  ../../../bin/k2hr3_osnl
  ../../../etc/k2hr3/k2hr3_osnl.conf
...
$ python -c "
> import os;
> print(
>   os.path.abspath(
>     '/usr/local/lib/python3.5/dist-packages/../../../etc/k2hr3/k2hr3_osnl.conf'
>   )
> );
> "
/usr/local/etc/k2hr3/k2hr3_osnl.conf
```

`/usr/local/etc/k2hr3/k2hr3_osnl.conf` is it. You should change the `transport_url` and the

`api_url` setting for your environment.

```
[DEFAULT]
debug_level = error

[oslo_messaging_notifications]
event_type = ^port\.delete\.end$
publisher_id = ^network.*$
transport_url = rabbit://user:pass@127.0.0.1:5672/
topic = notifications
exchange = neutron

[k2hr3]
api_url = https://localhost/v1/role
allow_self_signed_cert = False
```

FYI: The *Usage* page describes every setting parameters.

2.3 Start

This chapter instructs how to install the listener.

```
$ sudo k2hr3_osnl -c /path/to/k2hr3_osnl.conf
```

No error means the listener successfully starts to listen to the next notification message.

2.4 Service Management

While you have already successfully started the listener, you would like to prepare for following troubles.

- The listener process is dead after the OS rebooted.
- The listener is dead when you have stopped the terminal which started the listener.

Most of modern OSs provide the way to register a process as a service to the service management system which launches them at boot time and stops them at shutdown. systemd is one of such a service which is installed in Debian 9, Fedora 28, CentOS 7 and other recent Linux distributions.

An example of what systemd config file might look like is:

```
[Unit]
Description=k2hr3_osnl
After=network-online.target

[Service]
Type=simple
WorkingDirectory=/tmp
Environment=HOME=/tmp
User=nobody
Group=nobody
ExecStart=/usr/local/bin/k2hr3_osnl -c /usr/local/etc/k2hr3/k2hr3_osnl.conf
Restart=on-failure
PIDFile=/var/run/k2hr3_osnl.pid
```

(continues on next page)

(continued from previous page)

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

FYI: systemd.unit and systemd.service page describe meaning of parameters.

The syntax is the ".INI" style. `ExecStart` specifies the absolute `k2hr3_osnl` path. The path depends on your OS. I found it in `/usr/local/bin/k2hr3_osnl` in my environment.

```
$ which k2hr3_osnl
/usr/local/bin/k2hr3_osnl
```

After update the `ExecStart`, save the configuration to the `/lib/systemd/system/k2hr3_osnl.service` and register it to systemd. Please note the systemd configuration path depends on you OS.

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable k2hr3_osnl.service
```

After that, you tell systemd to look for your service at the first command and you tell systemd to enable it at the second command, so that it will start every time the system boots.

Then, you start the k2hr3_osnl as a service.

```
$ sudo systemctl start k2hr3_osnl.service
```

You can see the service status:

```
$ sudo systemctl status k2hr3_osnl.service
```

If you have got some errors, you should check logs put on stderr at first. Then please send a issue with it from issue.

CHAPTER 3

Usage

In this chapter I will explain how to configure the k2hr3_osnl. The k2hr3_osnl primarily consists of three parts of processing.

1. Listening to OpenStack service backend
2. Parsing messages from OpenStack service backend to extract an instance-id
3. Calling the k2hr3 api with the instance-id

`k2hr3_osnl.conf` defines the k2hr3_osnl behavior. You can get the path by two commands.

```
$ sudo pip3 show -f k2hr3_osnl
Name: k2hr3-osnl
...
Location: /usr/local/lib/python3.5/dist-packages
Requires: oslo.messaging, oslo.config
Files:
  ../../../bin/k2hr3_osnl
  ../../../etc/k2hr3/k2hr3_osnl.conf
...
$ python -c "
> import os;
> print(os.path.abspath('/usr/local/lib/python3.5/dist-packages/../../../etc/k2hr3/
↪k2hr3_osnl.conf'));
> "
/usr/local/etc/k2hr3/k2hr3_osnl.conf
```

`/usr/local/etc/k2hr3/k2hr3_osnl.conf` is it in this case.

Please note all information here I describe is based on OpenStack Rocky.

3.1 Listening to OpenStack service backend

The following 3 parameters define the listener behavior: `transport_url`, `topic` and `exchange` in k2hr3_osnl.conf.

The `transport_url` specifies the address of OpenStack service backend and how to connect with it. oslo.messaging describes the syntax:

> transport://user:pass@host1:port[,hostN:portN]/virtual_host

The transport value specifies the notification backend as one of **amqp**, RabbitMQ, Apache Kafka and other backend. The default backend is RabbitMQ. For Instance, assuming the backend service is RabbitMQ , the file might contain:

```
[oslo_messaging_notifications]
transport_url = rabbit://guest:guestpass@127.0.0.1:5672/
```

The setting above means:

- rabbitmq is a backend server.
- user name is guest.
- password is guestpass.
- address is localhost.
- port is 5672.

Please note username and password is required for security reason. **'RabbitMQ User Management'_** describes how to add a username and password.

The `topic` parameter identifies where a message should be sent or what messages the k2hr3_osnl is listening for. The OpenStack services emit messages by the oslo.messaging Notifier which requires `topic(s)`. A default value of `topic(s)` is `notifications` which is the same with the k2hr3_osnl's default `topic` value. An example of what the file might contain is:

```
[oslo_messaging_notifications]
topic = notifications
```

Please note the `topic` must be the same between OpenStack services and the k2hr3_osnl, because it is a part of subscriber queue name in OpenStack backend that the k2hr3_osnl listens to. So please remember you would need update it if OpenStack service administrators can change it the other value.

The final parameter is `exchange`. This parameter represents a container within which each service's topics are scoped. OpenStack services register the exchange when the send notifications by calling the set_transport_defaults function in oslo.messaging. The default value of `exchange` is 'openstack'.

What I have explained in this chapter:

- k2hr3_osnl connect with OpenStack service backend by transport_url.
- OpenStack services publish notifications to the configured exchange with a configured topic.
- The default topic name is `notifications`. It can be changed.
- The exchange is almost same with the OpenStack service publishes the notification message.

3.2 Parsing a message

A message format depends on your OpenStack service settings. Currently the k2hr3_osnl can parse the following 3 kinds of message.

1. a message from OpenStack neutron
2. a versioned message from OpenStack nova
3. a non-versioned message from OpenStack nova

I will explain them one by one.

3.2.1 Parsing a message from OpenStack neutron

We assume the following message from OpenStack neutron.

```
{
    "event_type": "port.delete.end",
    "message_id": "76c35877-9d0c-4faf-b4e5-7c51828f37a0",
    "payload": {
        ...
        "device_id": "12345678-1234-5678-1234-567812345678",
        "device_owner": "compute:nova",
        "extra_dhcp_opts": [],
        "fixed_ips": [
            {
                "ip_address": "172.24.4.18",
                "subnet_id": "subnet01-ffff-ffff-ffff-ffffffffffff"
            },
            {
                "ip_address": "2001:db8::6",
                "subnet_id": "subnet02-ffff-ffff-ffff-ffffffffffff"
            }
        ],
        ...
    },
    "priority": "INFO",
    "publisher_id": "network.hostname.domain_name",
    "timestamp": "2018-11-25 09:00:06.842727"
}
```

The `event_type` represents a event which OpenStack services send notification about and the `publisher_id` identifies who published the message. Let's see the 'oslo_messaging_notifications' group configuration to catch this message.

```
[oslo_messaging_notifications]
event_type = ^port\.delete\.end$
publisher_id = ^network.*$
transport_url = rabbit://user:pass@127.0.0.1:5672/
topic = notifications
exchange = neutron
```

The `event_type` and `publisher_id` define white rules that means k2hr3_osnl only parse the messages that match the filter's rules. If your Openstack neutron emits a same message with this example, you can use the same configure with this example.

Please note we assume the OpenStack neutron use the **messagingv2** driver. If you don't know much about the driver what your OpenStack neutron uses, Please ask your OpenStack system administrator or investigate your /etc/neutron/neutron.conf. Here is my neutron.conf setting.

```
[oslo_messaging_notifications]
#
# From oslo.messaging
#
# The Drivers(s) to handle sending notifications. Possible values are
# messaging, messagingv2, routing, log, test, noop (multi valued)
```

(continues on next page)

(continued from previous page)

```
# Deprecated group/name - [DEFAULT]/notification_driver
driver = messagingv2
```

What I have explained in this chapter:

- k2hr3_osnl only listen to the message matches with defined in the configuration.
- Regular expression in filters is allowed.

3.2.2 Parsing versioned messages from OpenStack nova

In this chapter I will explain how to configure the k2hr3_osnl to parse messages from OpenStack nova. When we tested the k2hr3_osnl with OpenStack neutron with the driver configuration was **not** messagingv2, the k2hr3_osnl could not get any notification messages we expected. If you met with same situation, please try the configuration in this chapter.

We assume the following message from OpenStack nova.

```
{
    "event_type" : "instance.delete.end",
    "payload": {
        "nova_object.data": {
            "action_initiator_project": "project_string",
            ...
            "block_devices": [
                {
                    "nova_object.data": {
                        ...
                        "volume_id": "volumeid-ffff-ffff-ffff-ffffffffffff"
                    },
                    "nova_object.name": "BlockDevicePayload",
                    "nova_object.namespace": "nova",
                    "nova_object.version": "1.0"
                }
            ],
            ...
        "host": "node1.example.com",
        ...
        "uuid": "12345678-1234-5678-1234-567812345678"
    },
    "priority": "INFO",
    "publisher_id" : "nova-compute:node1.example.com",
}
```

Here is a sample `oslo_messaging_notifications` group configuration k2hr3_osnl can read the message above.

```
[oslo_messaging_notifications]
event_type = ^instance\.delete\.end$
publisher_id = ^nova-compute:.*$
transport_url = rabbit://user:pass@127.0.0.1:5672/
topic = versioned_notifications
exchange = nova
```

You will recognize all items other than transport_url are different with neutron's case! Each OpenStack service defines its own event_type. FIY: OpenStack nova defines many event types.

https://docs.openstack.org/nova/latest/reference/notifications.html#existing-versioned-notifications

What I have explained in this chapter:

- OpenStack nova publishes different messages format from OpenStack neutron.
- k2hr3_osnl can read messages from OpenStack nova too by changing the configuration.

3.3 List of configuration items

Settings in the configuration file define the k2hr3_osnl behavior except for loglevel. Loglevel augments override loglevel settings in configuration. If you want to change the loglevel only, you need not to change configuration file. use `-d` option.

```
$ k2hr3_osnl --help
usage: k2hr3_osnl [-h] [-c CONFIG_FILE] [-d {debug,info,warn,error,critical}]
                  [-l {debug,info,warn,error,critical}] [-f LOG_FILE] [-v]

An oslo.messaging notification listener for k2hr3.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file CONFIG_FILE
                        config file path
  -d {debug,info,warn,error,critical}
                        debug level. default: defined in the config_file
  -l {debug,info,warn,error,critical}
                        dependent libraries loglevel. default: defined in the
                        config_file
  -f LOG_FILE           log file path. default: defined in the config_file
  -v                    show program's version number and exit
```

The configuration file consists of 3 parts.

- **oslo_messaging_notifications** configurations for the oslo_messaging library.
- **k2hr3** configurations for the K2hR3 system.
- **DEFAULT** the others.

The configuration file syntax is the ".INI" format. Here is a default sample configuration.

```
[DEFAULT]
debug_level = error
log_file = sys.stderr
libs_debug_level = warning

[oslo_messaging_notifications]
event_type = ^port\.delete\.end$
publisher_id = ^network.*$
transport_url = rabbit://user:pass@127.0.0.1:5672/
topic = notifications
exchange = neutron
executor = threading
pool = k2hr3_osnl
allow_requeue = True

[k2hr3]
```

(continues on next page)

(continued from previous page)

```
api_url = https://localhost/v1/role
timeout_seconds = 30
retries = 3
retry_interval_seconds = 60
allow_self_signed_cert = False
requeue_on_error = False
```

3.3.1 [DEFAULT]

debug_level logging level. Each of debug, info, warning or error. (**default:** warning).

log_file destination of logging. (**default:** sys.stderr)

libs_debug_level log level. (**default:** warning)

3.3.2 [oslo_messaging_notifications]

event_type event type of the notification message(**default:** ^port.delete.end$).

publisher_id publisher id of the notification message(**default:** ^network.*$)

transport_url transport url(**default:** rabbit://guest:guest@127.0.0.1:5672/)

topic topic of the notification message(**default:** notifications)

exchange exchange of the notification message(**default:** neutron)

executor executor of the listener(**default:** threading)

pool n pool identification of message queue(**default:** k2hr3_osnl)

allow_requeue allow requeue if error occurred(**default:** True)

3.3.3 [k2hr3]

api_url K2hR3 WebAPI Url(**default:** https://localhost/v1/role).

timeout_seconds connection timeout in second(**default:** 30)

retries retries(**default:** 3)

retry_interval_seconds interval(**default:** 60)

allow_self_signed_cert certification(**default:** True)

requeue_on_error error(**default:** True)

CHAPTER 4

k2hr3_osnl

4.1 k2hr3_osnl package

4.1.1 Submodules

4.1.2 k2hr3_osnl.cfg module

Parses a config file and stores configurations.

class k2hr3_osnl.cfg.**K2hr3Conf**(*path: pathlib.Path*)

Bases: oslo_config.cfg.ConfigOpts

Parses and stores configurations.

This class is a wrapper of oslo_config.cfg.ConfigOpts class. https://github.com/openstack/oslo.config/blob/master/oslo_config/cfg.py

Simple usage:

```
>>> from k2hr3_osnl.exceptions import K2hr3ConfError
>>> from k2hr3_osnl.cfg import K2hr3Conf
>>> from pathlib import Path
... try:
...         conf = K2hr3Conf(Path('etc/k2hr3_osnl.conf'))
...         print(conf.oslo_messaging_notifications.event_type)
... except K2hr3ConfError as error:
...         print('{}'.format(error))
...
^port\.delete\.end$
```

4.1.3 k2hr3_osnl.endpoint module

An endpoint for the oslo_messaging notification message listener.

class k2hr3_osnl.endpoint.**K2hr3NotificationEndpoint**(*conf: k2hr3_osnl.cfg.K2hr3Conf*)

Bases: object

An endpoint called by a OpenStack dispatcher when a filtered notification message arrives.

Simple usage:

```
>>> from k2hr3_osnl.cfg import K2hr3Conf
>>> from k2hr3_osnl.exceptions import K2hr3Error
>>> from k2hr3_osnl.endpoint import K2hr3NotificationEndpoint
>>> import k2hr3_osnl
>>> from pathlib import Path
>>> try:
...     conf = K2hr3Conf(Path('etc/k2hr3_osnl.conf'))
...     endpoints = [K2hr3NotificationEndpoint(conf)]
...     k2hr3_osnl.listen(endpoints)
... except K2hr3Error as error:
...     print(error)
```

conf

Returns the K2hr3Conf object.

info(*context: Dict[str, object], publisher_id: str, event_type: str, payload: Dict[str, object], metadata: Dict[str, object]*)

Notification endpoint in info priority.

Notification messages that match the filter's rules will be passed to the endpoint's methods. The oslo_messaging's callback function dispatcher calls when messages in 'info' priority have arrived.

Reference:

- https://docs.openstack.org/oslo.messaging/latest/reference/notification_listener.html
- https://github.com/openstack/oslo.messaging/blob/master/oslo_messaging/notify/dispatcher.py#L74

Note: This function catches all exceptions to avoid an infinite loop. If this function hasn't handled unexpected exceptions, the caller(dispatcher) would have caught them and returned the NotificationResult.REQUEUE to the message queue server which can cause infinite loop. To avoid the posibility of inifinite loop, we catches standard exception in this function.

Parameters

- **context** (*dict*) – Context of a notification for NotificationFilter.
- **publisher_id** (*str*) – Publisher_id of a notification for NotificationFilter
- **event_type** (*str*) – Event_type of a notification for NotificationFilter
- **payload** (*dict*) – Payload of a notification for NotificationFilter.
- **metadata** (*dict*) – Metadata of a notification for NotificationFilter.

Returns NotificationResult.HANDLED or NotificationResult.REQUEUE

4.1.4 k2hr3_osnl.exceptions module

Exception classes for the oslo_messaging notification message listener.

exception k2hr3_osnl.exceptions.**K2hr3ConfError**(*msg: str = None*)

Bases: *k2hr3_osnl.exceptions.K2hr3Error*

Raised when failed to instantiate a k2hr3Conf class.

exception k2hr3_osnl.exceptions.**K2hr3Error**
Bases: Exception

A base class of various exceptions from k2hr3_osnl package classes.

exception k2hr3_osnl.exceptions.**K2hr3NotificationEndpointError**(*msg: str = None*)
Bases: *k2hr3_osnl.exceptions.K2hr3Error*

Raised when failed to instantiate a K2hr3NotificationEndpoint class.

4.1.5 k2hr3_osnl.useragent module

Sends http requests to the k2hr3 api. Classes in this module are not public.

4.1.6 Module contents

K2hr3 OpenStack Notification message Listener.

class k2hr3_osnl.**K2hr3Conf**(*path: pathlib.Path*)
Bases: oslo_config.cfg.ConfigOpts

Parses and stores configurations.

This class is a wrapper of oslo_config.cfg.ConfigOpts class. https://github.com/openstack/oslo.config/blob/master/oslo_config/cfg.py

Simple usage:

```
>>> from k2hr3_osnl.exceptions import K2hr3ConfError
>>> from k2hr3_osnl.cfg import K2hr3Conf
>>> from pathlib import Path
... try:
...         conf = K2hr3Conf(Path('etc/k2hr3_osnl.conf'))
...         print(conf.oslo_messaging_notifications.event_type)
... except K2hr3ConfError as error:
...         print('{}'.format(error))
...
^port\.delete\.end$
```

exception k2hr3_osnl.**K2hr3ConfError**(*msg: str = None*)
Bases: *k2hr3_osnl.exceptions.K2hr3Error*

Raised when failed to instantiate a k2hr3Conf class.

class k2hr3_osnl.**K2hr3NotificationEndpoint**(*conf: k2hr3_osnl.cfg.K2hr3Conf*)
Bases: object

An endpoint called by a OpenStack dispatcher when a filtered notification message arrives.

Simple usage:

```
>>> from k2hr3_osnl.cfg import K2hr3Conf
>>> from k2hr3_osnl.exceptions import K2hr3Error
>>> from k2hr3_osnl.endpoint import K2hr3NotificationEndpoint
>>> import k2hr3_osnl
>>> from pathlib import Path
```

(continues on next page)

(continued from previous page)

```
>>> try:
...     conf = K2hr3Conf(Path('etc/k2hr3_osnl.conf'))
...     endpoints = [K2hr3NotificationEndpoint(conf)]
...     k2hr3_osnl.listen(endpoints)
... except K2hr3Error as error:
...     print(error)
```

conf
Returns the K2hr3Conf object.

info(*context: Dict[str, object], publisher_id: str, event_type: str, payload: Dict[str, object], metadata: Dict[str, object]*)
Notification endpoint in info priority.

Notification messages that match the filter's rules will be passed to the endpoint's methods. The oslo_messaging's callback function dispatcher calls when messages in 'info' priority have arrived.

Reference:

- https://docs.openstack.org/oslo.messaging/latest/reference/notification_listener.html
- https://github.com/openstack/oslo.messaging/blob/master/oslo_messaging/notify/dispatcher.py#L74

Note: This function catches all exceptions to avoid an infinite loop. If this function hasn't handled unexpected exceptions, the caller(dispatcher) would have caught them and returned the NotificationResult.REQUEUE to the message queue server which can cause infinite loop. To avoid the posibility of inifinite loop, we catches standard exception in this function.

Parameters

- **context** (*dict*) – Context of a notification for NotificationFilter.
- **publisher_id** (*str*) – Publisher_id of a notification for NotificationFilter
- **event_type** (*str*) – Event_type of a notification for NotificationFilter
- **payload** (*dict*) – Payload of a notification for NotificationFilter.
- **metadata** (*dict*) – Metadata of a notification for NotificationFilter.

Returns NotificationResult.HANDLED or NotificationResult.REQUEUE

exception k2hr3_osnl.**K2hr3NotificationEndpointError**(*msg: str = None*)
Bases: *k2hr3_osnl.exceptions.K2hr3Error*

Raised when failed to instantiate a K2hr3NotificationEndpoint class.

k2hr3_osnl.**listen**(*endpoints: List[k2hr3_osnl.endpoint.K2hr3NotificationEndpoint]*) → int
Runs a oslo_messaging notification listener for k2hr3.

This function is a library endpoint to start a oslo_messaging notification listener for k2hr3.

Parameters **endpoints** (*list of K2hr3NotificationEndpoint*) – endpoint to be called by dispatcher when notification messages arrive.

Returns 0 if success, otherwise 1.

Return type int

k2hr3_osnl.**main**() → int
Runs a oslo_messaging notification listener for k2hr3.

You can configure the listener by the config file.

Simple usage:

$ k2hr3_osnl -c etc/k2hr3_osnl.config

Returns 0 if success, otherwise 1.

Return type int

k2hr3_osnl.**version**() → str

Returns a version of k2hr3_osnl package.

Returns version

Return type str

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/hiwakaba/k2hr3_osnl/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

K2hR3 OpenStack Notification Listener could always use more documentation, whether as part of the official K2hR3 OpenStack Notification Listener docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/hiwakaba/k2hr3_osnl/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *k2hr3_osnl* for local development.

1. Fork the *k2hr3_osnl* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/k2hr3_osnl.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ cd k2hr3_osnl/
$ pip3 install pipenv
$ python3 -m pipenv install -dev --python /path/to/python3
$ pipenv shell
```

4. Create a branch for local development:

```
(k2hr3_osnl) $ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8, mypy, pylint and the tests, including testing other Python versions with make lint test

   Make sure you are in virtual enviroment:

```
(k2hr3_osnl) $ make lint test
```

6. Commit your changes and push your branch to GitHub:

```
(k2hr3_osnl) $ git add .
(k2hr3_osnl) $ git commit -m "Your detailed description of your changes."
(k2hr3_osnl) $ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6 and 3.7. Check https://travis-ci.org/hiwakaba/k2hr3_osnl/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ make test
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then visit the release page at https://github.com/hiwakaba/k2hr3_osnl/releases and create a new release note.

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Hirotaka Wakabayashi <hiwakaba@yahoo-corp.jp>

6.2 Contributors

- Takeshi Nakatani <nakatani@yahoo-corp.jp>

CHAPTER 7

History

7.1 0.0.14 (2019-02-04)

- Fixed variable '_nameToLevel' in global scope should not be mixedCase

7.2 0.0.13 (2019-01-22)

- Added a systemd Unit file and settings.
- Renamed filename(from k2hr3_osnl to k2hr3-osnl)

7.3 0.0.12 (2019-01-06)

- Added rst syntax checker
- Fixed rst syntax errors

7.4 0.0.11 (2019-01-06)

- Fixed rpmlint warings

7.5 0.0.10 (2019-01-06)

- Update docs.

7.6 0.0.9 (2019-01-05)

- Fixed errors on fc30.

7.7 0.0.8 (2019-01-03)

- Added the spec file.
- Changed comments on author, copyright and license.
- Changed the logging setting of test cases.
- Changed the dependent libraries version.

7.8 0.0.7 (2018-12-07)

- Update docs.

7.9 0.0.6 (2018-12-03)

- Deploy test from .travis.yaml

7.10 0.0.5 (2018-12-03)

- Upload test from travis dpl.

7.11 0.0.4 (2018-11-24)

- Fixed docs/*.rst.

7.12 0.0.3 (2018-11-22)

- Fixed syntax error in README.rst.
- Added version checker in make dist.

7.13 0.0.2 (2018-11-22)

- Tested on Python3.5.

7.14 0.0.1 (2018-11-15)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

k

Index