

---

# **Ironic Inspector Documentation**

*Release 12.4.0.dev9*

**OpenStack Foundation**

**Feb 16, 2025**



# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Release Notes</b>	<b>5</b>
<b>3</b>	<b>Using Ironic Inspector</b>	<b>7</b>
3.1	Install Guide . . . . .	7
3.2	Command References . . . . .	16
3.3	Configuration Guide . . . . .	17
3.4	User Guide . . . . .	91
3.5	Administrator Guide . . . . .	104
<b>4</b>	<b>Contributor Docs</b>	<b>109</b>
4.1	How To Contribute . . . . .	109
<b>5</b>	<b>Indices and tables</b>	<b>183</b>
	<b>Python Module Index</b>	<b>185</b>
	<b>Index</b>	<b>187</b>



**Warning**

This project is now in the maintenance mode and new deployments of it are discouraged. Please use [built-in in-band inspection in ironic](#) instead. For existing deployments, see the [migration guide](#).



## INTRODUCTION

This is an auxiliary service for discovering hardware properties for a node managed by **Ironic**. Hardware introspection or hardware properties discovery is a process of getting hardware parameters required for scheduling from a bare metal node, given its power management credentials (e.g. IPMI address, user name and password).

- Free software: Apache license
- Source: <https://opendev.org/openstack/ironic-inspector/>
- Bugs: <https://bugs.launchpad.net/ironic-inspector>
- Downloads: <https://tarballs.openstack.org/ironic-inspector/>
- Documentation: <https://docs.openstack.org/ironic-inspector/latest/>
- Python client library and CLI tool: `python-ironic-inspector-client` (documentation).

### Note

**ironic-inspector** was called *ironic-discoverd* before version 2.0.0.





## RELEASE NOTES

For information on any current or prior version, see [the release notes](#).



## USING IRONIC INSPECTOR

### 3.1 Install Guide

Install from PyPI (you may want to use virtualenv to isolate your environment):

```
pip install ironic-inspector
```

Also there is a [DevStack](#) plugin for **ironic-inspector** - see [How To Contribute](#) for the current status.

Finally, some distributions (e.g. Fedora) provide **ironic-inspector** packaged, some of them - under its old name *ironic-discoverd*.

There are several projects you can use to set up **ironic-inspector** in production. [puppet-ironic](#) provides Puppet manifests, while [bifrost](#) provides an Ansible-based standalone installer. Refer to [Configuration](#) if you plan on installing **ironic-inspector** manually.

#### Note

Please beware of *possible DNS issues* when installing **ironic-inspector** on Ubuntu.

#### 3.1.1 Sample Configuration Files

To generate a sample configuration file, run the following command from the top level of the code tree:

```
tox -egenconfig
```

For a pre-generated sample configuration file, see *Ironic Inspector Configuration Options*.

To generate a sample policy file, run the following command from the top level of the code tree:

```
tox -egenpolicy
```

For a pre-generated sample configuration file, see *Ironic Inspector Policy*.

#### 3.1.2 Installation options

Starting with Train release, **ironic-inspector** can run in a non-standalone mode, which means **ironic-inspector** API and **ironic-inspector** conductor are separated services, they can be installed on the same host or different hosts.

Following are some considerations when you run **ironic-inspector** in non-standalone mode:

- Additional packages may be required depending on the tooz backend used in the installation. For example, `etcd3gw` is required if the backend driver is configured to use `etcd3+http://`, `pymemcache` is required to use `memcached://`. Some distributions may provide packages like `python3-etcd3gw` or `python3-memcache`. Supported drivers are listed at [Tooz drivers](#).
- For `ironic-inspector` running in non-standalone mode, PXE configuration is only required on the node where `ironic-inspector` conductor service is deployed.
- Switch to a database backend other than `sqlite`.

### 3.1.3 Configuration

Copy the sample configuration files to some permanent place (e.g. `/etc/ironic-inspector/inspector.conf`). Fill in these minimum configuration values:

- The `standalone` in the `DEFAULT` section - This determines whether `ironic-inspector` services are intended to be deployed separately.
- The `keystone_auth_token` section - credentials to use when checking user authentication.
- The `ironic` section - credentials to use when accessing **ironic** API. When **ironic** is deployed standalone with no authentication, specify the following:

```
[ironic]
auth_type=none
```

When **ironic** is deployed standalone with HTTP Basic authentication, valid credentials are also required:

```
[ironic]
auth_type=http_basic
username=myName
password=myPassword
```

- `connection` in the `database` section - SQLAlchemy connection string for the database. By default `ironic-inspector` uses `sqlite` as the database backend, if you are running `ironic-inspector` in a non-standalone mode, please change to other database backends.
- `dnsmasq_interface` in the `iptables` section - interface on which `dnsmasq` (or another DHCP service) listens for PXE boot requests (defaults to `br-ctlplane` which is a sane default for **tripleo**-based installations but is unlikely to work for other cases).
- if you wish to use the `dnsmasq` PXE/DHCP filter driver rather than the default `iptables` driver, see the [dnsmasq PXE filter](#) description.
- `store_data` in the `processing` section defines where introspection data is stored and takes one of three values:

**none**

introspection data is not stored (the default)

**database**

introspection data is stored in the database (recommended for standalone deployments)

**swift**

introspection data is stored in the Object Store service (recommended for full openstack deployments)

**Note**

It is possible to create third party storage backends using the `ironic_inspector.introspection_data.store` entry point.

See comments inside *the sample configuration* for other possible configuration options.

**Note**

Configuration file contains a password and thus should be owned by root and should have access rights like `0600`.

Here is an example *inspector.conf* (adapted from a gate run):

```
[DEFAULT]
debug = false
rootwrap_config = /etc/ironic-inspector/rootwrap.conf

[database]
connection = mysql+pymysql://root:<PASSWORD>@127.0.0.1/ironic_inspector?
↳ charset=utf8

[pxe_filter]
driver=iptables

[iptables]
dnsmasq_interface = br-ctlplane

[ironic]
os_region = RegionOne
project_name = service
password = <PASSWORD>
username = ironic-inspector
auth_url = http://127.0.0.1/identity
auth_type = password

[keystone_authtoken]
www_authenticate_uri = http://127.0.0.1/identity
project_name = service
password = <PASSWORD>
username = ironic-inspector
auth_url = http://127.0.0.1/identity_v2_admin
auth_type = password

[processing]
ramdisk_logs_dir = /var/log/ironic-inspector/ramdisk
store_data = swift

[swift]
```

(continues on next page)

(continued from previous page)

```
os_region = RegionOne
project_name = service
password = <PASSWORD>
username = ironic-inspector
auth_url = http://127.0.0.1/identity
auth_type = password
```

**Note**

Set `debug = true` if you want to see complete logs.

**ironic-inspector** requires root rights for managing iptables. It gets them by running `ironic-inspector-rootwrap` utility with `sudo`. To allow it, copy file `rootwrap.conf` and directory `rootwrap.d` to the configuration directory (e.g. `/etc/ironic-inspector/`) and create file `/etc/sudoers.d/ironic-inspector-rootwrap` with the following content:

```
Defaults:stack !requiretty
stack ALL=(root) NOPASSWD: /usr/bin/ironic-inspector-rootwrap /etc/ironic-
↪inspector/rootwrap.conf *
```

**Danger**

Be very careful about typos in `/etc/sudoers.d/ironic-inspector-rootwrap` as any typo will break sudo for **ALL** users on the system. Especially, make sure there is a new line at the end of this file.

**Note**

`rootwrap.conf` and all files in `rootwrap.d` must be writeable only by root.

**Note**

If you store `rootwrap.d` in a different location, make sure to update the `filters_path` option in `rootwrap.conf` to reflect the change.

If your `rootwrap.conf` is in a different location, then you need to update the `rootwrap_config` option in `ironic-inspector.conf` to point to that location.

Replace `stack` with whatever user you'll be using to run **ironic-inspector**.

## Configuring IPA

`ironic-python-agent` is a ramdisk developed for **ironic** and support for **ironic-inspector** was added during the Liberty cycle. This is the default ramdisk starting with the Mitaka release.

**Note**

You need at least 2 GiB of RAM on the machines to use IPA built with `diskimage-builder` and at least 384 MiB to use the *TinyIPA*.

To build an **ironic-python-agent** ramdisk, use `ironic-python-agent-builder`. Alternatively, you can download a [prebuild image](#).

For local testing and CI purposes you can use a [TinyIPA image](#).

**Configuring PXE**

For the PXE boot environment, you'll need:

- TFTP server running and accessible (see below for using `dnsmasq`). Ensure `pxelinux.0` is present in the TFTP root.

Copy `ironic-python-agent.kernel` and `ironic-python-agent.initramfs` to the TFTP root as well.

- Next, setup `$TFTPBOOT/pxelinux.cfg/default` as follows:

```
default introspect

label introspect
kernel ironic-python-agent.kernel
append initrd=ironic-python-agent.initramfs ipa-inspection-callback-
↪url=http://{IP}:5050/v1/continue systemd.journald.forward_to_console=yes

ipappend 3
```

Replace `{IP}` with IP of the machine (do not use loopback interface, it will be accessed by ramdisk on a booting machine).

**Note**

While `systemd.journald.forward_to_console=yes` is not actually required, it will substantially simplify debugging if something goes wrong. You can also enable IPA debug logging by appending `ipa-debug=1`.

IPA is pluggable: you can insert introspection plugins called *collectors* into it. For example, to enable a very handy logs collector (sending ramdisk logs to **ironic-inspector**), modify the `append` line in `$TFTPBOOT/pxelinux.cfg/default`:

```
append initrd=ironic-python-agent.initramfs ipa-inspection-callback-
↪url=http://{IP}:5050/v1/continue ipa-inspection-collectors=default,logs,
↪systemd.journald.forward_to_console=yes
```

**Note**

You probably want to always keep the default collector, as it provides the basic information required for introspection.

- You need PXE boot server (e.g. *dnsmasq*) running on **the same** machine as **ironic-inspector**. Dont do any firewall configuration: **ironic-inspector** will handle it for you. In **ironic-inspector** configuration file set `dnsmasq_interface` to the interface your PXE boot server listens on. Here is an example *dnsmasq.conf*:

```
port=0
interface={INTERFACE}
bind-interfaces
dhcp-range={DHCP IP RANGE, e.g. 192.168.0.50,192.168.0.150}
enable-tftp
tftp-root={TFTP ROOT, e.g. /tftpboot}
dhcp-boot=pxelinux.0
dhcp-sequential-ip
```

#### Note

`dhcp-sequential-ip` is used because otherwise a lot of nodes booting simultaneously cause conflicts - the same IP address is suggested to several nodes.

## Configuring iPXE

iPXE allows better scaling as it primarily uses the HTTP protocol instead of slow and unreliable TFTP. You still need a TFTP server as a fallback for nodes not supporting iPXE. To use iPXE, youll need:

- TFTP server running and accessible (see above for using *dnsmasq*). Ensure `undionly.kpxe` is present in the TFTP root. If any of your nodes boot with UEFI, youll also need `ipxe.efi` there.
- You also need an HTTP server capable of serving static files. Copy `ironic-python-agent.kernel` and `ironic-python-agent.initramfs` there.
- Create a file called `inspector.ipxe` in the HTTP root (you can name and place it differently, just dont forget to adjust the *dnsmasq.conf* example below):

```
#!ipxe

:retry_dhcp
dhcp || goto retry_dhcp

:retry_boot
imgfree
kernel --timeout 30000 http://{IP}:8088/ironic-python-agent.kernel ipa-
↪inspection-callback-url=http://{IP}>:5050/v1/continue systemd.journald.
↪forward_to_console=yes BOOTIF=${mac} initrd=agent.ramdisk || goto retry_
↪boot
initrd --timeout 30000 http://{IP}:8088/ironic-python-agent.ramdisk ||
↪goto retry_boot
boot
```



**Note**

Older versions of the iPXE ROM tend to misbehave on unreliable network connection, thus we use the timeout option with retries.

Just like with PXE, you can customize the list of collectors by appending the `ipa-inspection-collectors` kernel option. For example:

```
ipa-inspection-collectors=default,logs,extra_hardware
```

- Just as with PXE, you'll need a PXE boot server. The configuration, however, will be different. Here is an example `dnsmasq.conf`:

```
port=0
interface={INTERFACE}
bind-interfaces
dhcp-range={DHCP IP RANGE, e.g. 192.168.0.50,192.168.0.150}
enable-tftp
tftp-root={TFTP ROOT, e.g. /tftpboot}
dhcp-sequential-ip
dhcp-match=ipxe,175
dhcp-match=set:efi,option:client-arch,7
dhcp-match=set:efi,option:client-arch,9
dhcp-match=set:efi,option:client-arch,11
# dhcpv6.option: Client System Architecture Type (61)
dhcp-match=set:efi6,option6:61,0007
dhcp-match=set:efi6,option6:61,0009
dhcp-match=set:efi6,option6:61,0011
dhcp-userclass=set:ipxe6,ipXE
# Client is already running iPXE; move to next stage of chainloading
dhcp-boot=tag:ipxe,http://{IP}:8088/inspector.ipxe
# Client is PXE booting over EFI without iPXE ROM,
# send EFI version of iPXE chainloader
dhcp-boot=tag:efi,tag:!ipxe,ipxe.efi
dhcp-option=tag:efi6,tag:!ipxe6,option6:bootfile-url,tftp://{IP}/ipxe.efi
# Client is running PXE over BIOS; send BIOS version of iPXE chainloader
dhcp-boot=undionly.kpxe,localhost.localdomain,{IP}
```

First, we configure the same common parameters as with PXE. Then we define `ipxe` and `efi` tags for IPv4 and `ipxe6` and `efi6` for IPv6. Nodes already supporting iPXE are ordered to download and execute `inspector.ipxe`. Nodes without iPXE booted with UEFI will get `ipxe.efi` firmware to execute, while the remaining will get `undionly.kpxe`.

### Configuring PXE for aarch64

For aarch64 Bare Metals, the PXE boot environment is basically the same as `x86_64`, you'll need:

- TFTP server running and accessible (see below for using `dnsmasq`). Ensure `grubaa64.efi` is present in the TFTP root. The firmware can be retrieved from the installation distributions for aarch64.
- Copy `ironic-agent.kernel` and `ironic-agent.initramfs` to the TFTP root as well.

Note that the ramdisk needs to be pre-built on an aarch64 machine with tools like `ironic-python-agent-builder`, see <https://docs.openstack.org/ironic-python-agent-builder/latest/admin/dib.html> for how to build ramdisk for aarch64.

- Next, setup `$TFTPBOOT/EFI/BOOT/grub.cfg` as follows:

```
set default="1"
set timeout=5

menuentry 'Introspection for aarch64' {
    linux ironic-agent.kernel text showopts selinux=0 ipa-inspection-
    ↪callback-url=http://{IP}:5050/v1/continueãipa-inspection-
    ↪collectors=defaultãipa-collect-lldp=1ãsystemd.journald.forward_to_
    ↪console=no
    initrd ironic-agent.initramfs
}
```

Replace `{IP}` with IP of the machine (do not use loopback interface, it will be accessed by ramdisk on a booting machine).

- Update DHCP options for aarch64, here is an example `dnsmasq.conf`:

```
port=0
interface={INTERFACE}
bind-interfaces
dhcp-range={DHCP IP RANGE, e.g. 192.168.0.50,192.168.0.150}
enable-tftp
dhcp-match=aarch64, option:client-arch, 11 # aarch64
dhcp-boot=tag:aarch64, grubaa64.efi
tftp-root={TFTP ROOT, e.g. /tftpboot}
dhcp-sequential-ip
```

### Configuring PXE for Multi-arch

If the environment consists of bare metals with different architectures, normally different ramdisks are required for each architecture. The grub built-in variable `grub_cpu` could be used to locate the correct config file for each of them.

For example, setup `$TFTPBOOT/EFI/BOOT/grub.cfg` as following:

```
set default=master
set timeout=5
set hidden_timeout_quiet=false

menuentry "master" {
    configfile /tftpboot/grub-${grub_cpu}.cfg
}
```

Prepare specific grub config for each existing architectures, e.g. `grub-arm64.cfg` for ARM64 and `grub-x86_64.cfg` for x86\_64.

Update dnsmasq configuration to contain options for supported architectures.

### 3.1.4 Managing the ironic-inspector Database

**ironic-inspector** provides a command line client for managing its database. This client can be used for upgrading, and downgrading the database using **alembic** migrations.

If this is your first time running **ironic-inspector** to migrate the database, simply run:

```
ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf
↪upgrade
```

If you have previously run a version of **ironic-inspector** earlier than 2.2.0, the safest thing is to delete the existing SQLite database and run **upgrade** as shown above. However, if you want to save the existing database, to ensure your database will work with the migrations, you'll need to run an extra step before upgrading the database. You only need to do this the first time running version 2.2.0 or later.

If you are upgrading from **ironic-inspector** version 2.1.0 or lower:

```
ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf
↪stamp --revision 578f84f38d
ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf
↪upgrade
```

If you are upgrading from a git master install of the **ironic-inspector** after *rules* were introduced:

```
ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf
↪stamp --revision d588418040d
ironic-inspector-dbsync --config-file /etc/ironic-inspector/inspector.conf
↪upgrade
```

Other available commands can be discovered by running:

```
ironic-inspector-dbsync --help
```

### 3.1.5 Running

#### Running in standalone mode

Execute:

```
ironic-inspector --config-file /etc/ironic-inspector/inspector.conf
```

#### Running in non-standalone mode

API service can be started in development mode with:

```
ironic-inspector-api-wsgi -p 5050 -- --config-file /etc/ironic-inspector/
↪inspector.conf
```

For production, the **ironic-inspector** API service should be hosted under a web service. Below is a sample configuration for Apache with module `mod_wsgi`:

```
Listen 5050
<VirtualHost *:5050>
```

(continues on next page)

(continued from previous page)

```
WSGIDaemonProcess ironic-inspector user=stack group=stack threads=10
<!--display-name=${GROUP}
WSGIScriptAlias / /usr/local/bin/ironic-inspector-api-wsgi

SetEnv APACHE_RUN_USER stack
SetEnv APACHE_RUN_GROUP stack
WSGIProcessGroup ironic-inspector

ErrorLog /var/log/apache2/ironic_inspector_error.log
LogLevel info
CustomLog /var/log/apache2/ironic_inspector_access.log combined

<Directory /opt/stack/ironic-inspector/ironic_inspector/cmd>
    WSGIProcessGroup ironic-inspector
    WSGIApplicationGroup %{GLOBAL}
    AllowOverride All
    Require all granted
</Directory>
</VirtualHost>
```

You can refer to [ironic installation document](#) for more guides.

ironic-inspector conductor can be started with:

```
ironic-inspector-conductor --config-file /etc/ironic-inspector/inspector.conf
```

## 3.2 Command References

Here are references for commands not elsewhere documented.

### 3.2.1 ironic-inspector-status

#### Synopsis

```
ironic-inspector-status <category> <command> [<args>]
```

#### Description

**ironic-inspector-status** is a tool that provides routines for checking the status of the ironic-inspector deployment.

#### Options

The standard pattern for executing a **ironic-inspector-status** command is:

```
ironic-inspector-status <category> <command> [<args>]
```

Run without arguments to see a list of available command categories:

```
ironic-inspector-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:

```
ironic-inspector-status upgrade
```

These sections describe the available categories and arguments for **ironic-inspector-status**.

## Upgrade

### **ironic-status upgrade check**

Performs a release-specific readiness check before restarting services with new code. This command expects to have complete configuration and access to databases and services.

#### 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

##### Wallaby

- Adds initial status check command as it was not previously needed as the database structure and use of ironic-inspectors of ironic-inspector did not require the command previously.
- Adds a check to validate the configured policy file is not JSON based as JSON based policies have been deprecated.

## 3.3 Configuration Guide

The ironic-inspector service operation is defined by a configuration file. The overview of configuration file options follow.

### 3.3.1 ironic-inspector.conf

#### DEFAULT

##### **listen\_address**

**Type**  
string

**Default**  
::

IP to listen on.

### **listen\_port**

**Type**

port number

**Default**

5050

**Minimum Value**

0

**Maximum Value**

65535

Port to listen on.

### **listen\_unix\_socket**

**Type**

string

**Default**

<None>

Unix socket to listen on. Disables listen\_address and listen\_port.

### **listen\_unix\_socket\_mode**

**Type**

unknown type

**Default**

<None>

File mode (an octal number) of the unix socket to listen on. Ignored if listen\_unix\_socket is not set.

### **host**

**Type**

string

**Default**

localhost

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. This can be an opaque identifier. It is not necessarily a hostname, FQDN, or IP address. However, the node name must be valid within an AMQP key, and if using ZeroMQ, a valid hostname, FQDN, or IP address.

### **auth\_strategy**

**Type**

string

**Default**

keystone

**Valid Values**

noauth, keystone, http\_basic

Authentication method used on the ironic-inspector API. noauth, keystone or http\_basic are valid options. noauth will disable all authentication.

**Possible values**

**noauth**

no authentication

**keystone**

use the Identity service for authentication

**http\_basic**

HTTP basic authentication

**http\_basic\_auth\_user\_file**

**Type**

string

**Default**

/etc/ironic-inspector/htpasswd

Path to Apache format user authentication file used when auth\_strategy=http\_basic

**timeout**

**Type**

integer

**Default**

3600

**Maximum Value**

315576000

Timeout after which introspection is considered failed, set to 0 to disable.

**clean\_up\_period**

**Type**

integer

**Default**

60

**Minimum Value**

0

Amount of time in seconds, after which repeat clean up of timed out nodes and old nodes status information. WARNING: If set to a value of 0, then the periodic task is disabled and inspector will not sync with ironic to complete the internal clean-up process. Not advisable if the deployment uses a PXE filter, and will result in the ironic-inspector ceasing periodic cleanup activities.

**leader\_election\_interval**

**Type**

integer

**Default**

10

Interval (in seconds) between leader elections.

**use\_ssl**

**Type**

boolean

**Default**

False

SSL Enabled/Disabled

**max\_concurrency**

**Type**

integer

**Default**

1000

**Minimum Value**

2

The green thread pool size.

**introspection\_delay**

**Type**

integer

**Default**

5

Delay (in seconds) between two introspections. Only applies when boot is managed by ironic-inspector (i.e. `manage_boot==True`).

**ipmi\_address\_fields**

**Type**

list

**Default**

['redfish\_address', 'ilo\_address', 'drac\_host',  
'drac\_address', 'ibmc\_address']

Ironic driver\_info fields that are equivalent to ipmi\_address.

**rootwrap\_config**

**Type**

string

**Default**

/etc/ironic-inspector/rootwrap.conf

Path to the rootwrap configuration file to use for running commands as root



**api\_max\_limit**

**Type**  
integer

**Default**  
1000

**Minimum Value**  
1

Limit the number of elements an API list-call returns

**can\_manage\_boot**

**Type**  
boolean

**Default**  
True

Whether the current installation of ironic-inspector can manage PXE booting of nodes. If set to False, the API will reject introspection requests with manage\_boot missing or set to True.

**enable\_mdns**

**Type**  
boolean

**Default**  
False

Whether to enable publishing the ironic-inspector API endpoint via multicast DNS.

**standalone**

**Type**  
boolean

**Default**  
True

Whether to run ironic-inspector as a standalone service. Its EXPERIMENTAL to set to False.

**executor\_thread\_pool\_size**

**Type**  
integer

**Default**  
64

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

Table 1: 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**  
openstack

The default exchange under which topics are scoped. May be overridden by an exchange name specified in the `transport_url` option.

### **rpc\_ping\_enabled**

**Type**  
boolean

**Default**  
False

Add an endpoint to answer to ping calls. Endpoint is named `oslo_rpc_server_ping`

### **debug**

**Type**  
boolean

**Default**  
False

**Mutable**  
This option can be changed without restarting.

If set to true, the logging level will be set to DEBUG instead of the default INFO level.

**log\_config\_append****Type**

string

**Default**

&lt;None&gt;

**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 2: 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**

&lt;None&gt;

(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 3: Deprecated Variations

Group	Name
DEFAULT	logfile

**log\_dir****Type**

string

**Default**

&lt;None&gt;

(Optional) The base directory used for relative `log_file` paths. This option is ignored if `log_config_append` is set.

Table 4: Deprecated Variations

Group	Name
DEFAULT	logdir

### **watch\_log\_file**

**Type**

boolean

**Default**

False

Uses logging handler designed to watch file system. When log file is moved or removed this handler will open a new log file with specified path instantaneously. It makes sense only if `log_file` option is specified and Linux platform is used. This option is ignored if `log_config_append` is set.

**Warning**

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

**Reason**

This function is known to have bene broken for long time, and depends on the unmaintained library

### **use\_syslog**

**Type**

boolean

**Default**

False

Use syslog for logging. Existing syslog format is DEPRECATED and will be changed later to honor RFC5424. This option is ignored if `log_config_append` is set.

### **use\_journal**

**Type**

boolean

**Default**

False

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages. This option is ignored if `log_config_append` is set.

### **syslog\_log\_facility**

**Type**

string

**Default**

LOG\_USER

Syslog facility to receive log lines. This option is ignored if `log_config_append` is set.

**use\_json**

**Type**

boolean

**Default**

False

Use JSON formatting for logging. This option is ignored if `log_config_append` is set.

**use\_stderr**

**Type**

boolean

**Default**

False

Log output to standard error. This option is ignored if `log_config_append` is set.

**log\_color**

**Type**

boolean

**Default**

False

(Optional) Set the color key according to log levels. This option takes effect only when logging to `stderr` or `stdout` is used. This option is ignored if `log_config_append` is set.

**log\_rotate\_interval**

**Type**

integer

**Default**

1

The amount of time before the log files are rotated. This option is ignored unless `log_rotation_type` is set to `interval`.

**log\_rotate\_interval\_type**

**Type**

string

**Default**

days

**Valid Values**

Seconds, Minutes, Hours, Days, Weekday, Midnight

Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

### **max\_logfile\_count**

**Type**  
integer

**Default**  
30

Maximum number of rotated log files.

### **max\_logfile\_size\_mb**

**Type**  
integer

**Default**  
200

Log file maximum size in MB. This option is ignored if `log_rotation_type` is not set to size.

### **log\_rotation\_type**

**Type**  
string

**Default**  
none

**Valid Values**  
interval, size, none

Log rotation type.

#### **Possible values**

##### **interval**

Rotate logs at predefined time intervals.

##### **size**

Rotate logs once they reach a predefined size.

##### **none**

Do not rotate log files.

### **logging\_context\_format\_string**

**Type**  
string

**Default**  
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s  
[%s] %(request\_id)s %(user\_identity)s  
%(instance)s%(message)s

Format string to use for log messages with context. Used by `oslo_log.formatters.ContextFormatter`

### **logging\_default\_format\_string**

**Type**  
string

**Default**

```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s [-]
%(instance)s%(message)s
```

Format string to use for log messages when context is undefined. Used by `oslo_log.formatters.ContextFormatter`

**logging\_debug\_format\_suffix****Type**

string

**Default**

```
%(funcName)s %(pathname)s:%(lineno)d
```

Additional data to append to log message when logging level for the message is DEBUG. Used by `oslo_log.formatters.ContextFormatter`

**logging\_exception\_prefix****Type**

string

**Default**

```
%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s
%(instance)s
```

Prefix each line of exception output with this format. Used by `oslo_log.formatters.ContextFormatter`

**logging\_user\_identity\_format****Type**

string

**Default**

```
%(user)s %(project)s %(domain)s %(system_scope)s
%(user_domain)s %(project_domain)s
```

Defines the format string for `%(user_identity)s` that is used in `logging_context_format_string`. Used by `oslo_log.formatters.ContextFormatter`

**default\_log\_levels****Type**

list

**Default**

```
['sqlalchemy=WARNING', 'iso8601=WARNING',
'requests=WARNING', 'urllib3.connectionpool=WARNING',
'keystonemiddleware=WARNING', 'keystoneauth=WARNING',
'ironicclient=WARNING', 'amqp=WARNING', 'amqpplib=WARNING',
'stevedore=WARNING', 'oslo.messaging=WARNING',
'oslo_messaging=WARNING']
```

List of package logging levels in `logger=LEVEL` pairs. This option is ignored if `log_config_append` is set.

**publish\_errors**

**Type**

boolean

**Default**

False

Enables or disables publication of error events.

**instance\_format**

**Type**

string

**Default**

"[instance: %(uuid)s] "

The format for an instance that is passed with the log message.

**instance\_uuid\_format**

**Type**

string

**Default**

"[instance: %(uuid)s] "

The format for an instance UUID that is passed with the log message.

**rate\_limit\_interval**

**Type**

integer

**Default**

0

Interval, number of seconds, of log rate limiting.

**rate\_limit\_burst**

**Type**

integer

**Default**

0

Maximum number of logged messages per rate\_limit\_interval.

**rate\_limit\_except\_level**

**Type**

string

**Default**

CRITICAL

**Valid Values**

CRITICAL, ERROR, INFO, WARNING, DEBUG,



Log level name used by rate limiting. Logs with level greater or equal to `rate_limit_except_level` are not filtered. An empty string means that all levels are filtered.

### **fatal\_deprecations**

**Type**

boolean

**Default**

False

Enables or disables fatal status of deprecations.

### **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.

**capabilities****boot\_mode****Type**

boolean

**Default**

False

Whether to store the boot mode (BIOS or UEFI).

**cpu\_flags****Type**

dict

**Default**

```
{'vmx': 'cpu_vt', 'svm': 'cpu_vt', 'aes': 'cpu_aes', 'pse':  
'cpu_hugepages', 'pdpe1gb': 'cpu_hugepages_1g', 'smx':  
'cpu_txt'}
```

Mapping between a CPU flag and a capability to set if this flag is present.

**coordination****backend\_url****Type**

string

**Default**

memcached://localhost:11211

The backend URL to use for distributed coordination. EXPERIMENTAL.

### **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**  
[]

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', 'POST', 'PUT', 'HEAD', 'PATCH', 'DELETE', 'OPTIONS']

Indicate which methods can be used during the actual request.

#### **allow\_headers**

**Type**  
list

**Default**

```
['X-Auth-Token', 'X-OpenStack-Ironic-Inspector-API-Minimum-Version',  
'X-OpenStack-Ironic-Inspector-API-Maximum-Version',  
'X-OpenStack-Ironic-Inspector-API-Version']
```

Indicate which header field names may be used during the actual request.

**database**

**sqlite\_synchronous**

**Type**

boolean

**Default**

True

If True, SQLite uses synchronous mode.

**backend**

**Type**

string

**Default**

sqlalchemy

The back end to use for the database.

**connection**

**Type**

string

**Default**

<None>

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

**slave\_connection**

**Type**

string

**Default**

<None>

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

**asyncio\_connection**

**Type**

string

**Default**

<None>

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

### **asyncio\_slave\_connection**

**Type**  
string

**Default**  
<None>

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

### **mysql\_sql\_mode**

**Type**  
string

**Default**  
TRADITIONAL

The SQL mode to be used for MySQL sessions. This option, including the default, overrides any server-set SQL mode. To use whatever SQL mode is set by the server configuration, set this to no value. Example: `mysql_sql_mode=`

### **mysql\_wsrep\_sync\_wait**

**Type**  
integer

**Default**  
<None>

For Galera only, configure `wsrep_sync_wait` causality checks on new connections. Default is None, meaning dont configure any setting.

### **connection\_recycle\_time**

**Type**  
integer

**Default**  
3600

Connections which have been present in the connection pool longer than this number of seconds will be replaced with a new one the next time they are checked out from the pool.

### **max\_pool\_size**

**Type**  
integer

**Default**  
5

Maximum number of SQL connections to keep open in a pool. Setting a value of 0 indicates no limit.

### **max\_retries**

**Type**  
integer

**Default**

10

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

**retry\_interval**

**Type**

integer

**Default**

10

Interval between retries of opening a SQL connection.

**max\_overflow**

**Type**

integer

**Default**

50

If set, use this value for max\_overflow with SQLAlchemy.

**connection\_debug**

**Type**

integer

**Default**

0

**Minimum Value**

0

**Maximum Value**

100

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

**connection\_trace**

**Type**

boolean

**Default**

False

Add Python stack traces to SQL as comment strings.

**pool\_timeout**

**Type**

integer

**Default**

<None>

If set, use this value for pool\_timeout with SQLAlchemy.

### **use\_db\_reconnect**

**Type**

boolean

**Default**

False

Enable the experimental use of database reconnect on connection lost.

### **db\_retry\_interval**

**Type**

integer

**Default**

1

Seconds between retries of a database transaction.

### **db\_inc\_retry\_interval**

**Type**

boolean

**Default**

True

If True, increases the interval between retries of a database operation up to `db_max_retry_interval`.

### **db\_max\_retry\_interval**

**Type**

integer

**Default**

10

If `db_inc_retry_interval` is set, the maximum seconds between retries of a database operation.

### **db\_max\_retries**

**Type**

integer

**Default**

20

Maximum retries in case of connection error or deadlock error before error is raised. Set to -1 to specify an infinite retry count.

### **connection\_parameters**

**Type**

string

**Default**

''

Optional URL parameters to append onto the connection URL at connect time; specify as `param1=value1&param2=value2&`



## discovery

### enroll\_node\_driver

**Type**  
string

**Default**  
fake-hardware

The name of the Ironic driver used by the enroll hook when creating a new node in Ironic.

### enroll\_node\_fields

**Type**  
dict

**Default**  
{}

Additional fields to set on newly discovered nodes.

### enabled\_bmc\_address\_version

**Type**  
list

**Default**  
['4', '6']

IP version of BMC address that will be used when enrolling a new node in Ironic. Defaults to 4,6. Could be 4 (use v4 address only), 4,6 (v4 address have higher priority and if both addresses found v6 version is ignored), 6,4 (v6 is desired but fall back to v4 address for BMCs having v4 address, opposite to 4,6), 6 (use v6 address only and ignore v4 version).

## dnsmasq\_pxe\_filter

### dhcp\_hostsdire

**Type**  
string

**Default**  
/var/lib/ironic-inspector/dhcp-hostsdire

The MAC address cache directory, exposed to dnsmasq. This directory is expected to be in exclusive control of the driver.

### purge\_dhcp\_hostsdire

**Type**  
boolean

**Default**  
True

Purge the hostsdire upon driver initialization. Setting to false should only be performed when the deployment of inspector is such that there are multiple processes executing inside of the same host and namespace. In this case, the Operator is responsible for setting up a custom cleaning facility.

### **dnsmasq\_start\_command**

**Type**  
string

**Default**  
''

A (shell) command line to start the dnsmasq service upon filter initialization. Default: dont start.

### **dnsmasq\_stop\_command**

**Type**  
string

**Default**  
''

A (shell) command line to stop the dnsmasq service upon inspector (error) exit. Default: dont stop.

## **exception**

### **fatal\_exception\_format\_errors**

**Type**  
boolean

**Default**  
False

Used if there is a formatting error when generating an exception message (a programming error). If True, raise an exception; if False, use the unformatted message.

Table 5: Deprecated Variations

Group	Name
ironic_lib	fatal_exception_format_errors

## **extra hardware**

### **strict**

**Type**  
boolean

**Default**  
False

If True, refuse to parse extra data if at least one record is too short. Additionally, remove the incoming data even if parsing failed.

## **healthcheck**

### **enabled**

**Type**  
boolean

**Default**

False

Enable the health check endpoint at /healthcheck. Note that this is unauthenticated. More information is available at [https://docs.openstack.org/oslo.middleware/latest/reference/healthcheck\\_plugins.html](https://docs.openstack.org/oslo.middleware/latest/reference/healthcheck_plugins.html).

**path****Type**

string

**Default**

/healthcheck

The path to respond to healthcheck requests on.

**Warning**

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

**detailed****Type**

boolean

**Default**

False

Show more detailed information as part of the response. Security note: Enabling this option may expose sensitive details about the service being monitored. Be sure to verify that it will not violate your security policies.

**backends****Type**

list

**Default**

[]

Additional backends that can perform health checks and report that information back as part of a request.

**allowed\_source\_ranges****Type**

list

**Default**

[]

A list of network addresses to limit source ip allowed to access healthcheck information. Any request from ip outside of these network addresses are ignored.

**ignore\_proxied\_requests****Type**

boolean

**Default**

False

Ignore requests with proxy headers.

**disable\_by\_file\_path**

**Type**

string

**Default**

<None>

Check the presence of a file to determine if an application is running on a port. Used by DisableByFileHealthcheck plugin.

**disable\_by\_file\_paths**

**Type**

list

**Default**

[]

Check the presence of a file based on a port to determine if an application is running on a port. Expects a port:path list of strings. Used by DisableByFilesPortsHealthcheck plugin.

**enable\_by\_file\_paths**

**Type**

list

**Default**

[]

Check the presence of files. Used by EnableByFilesHealthcheck plugin.

**iptables**

**dnsmasq\_interface**

**Type**

string

**Default**

br-ctlplane

Interface on which dnsmasq listens, the default is for VMs.

**firewall\_chain**

**Type**

string

**Default**

ironic-inspector

iptables chain name to use.

**ethoib\_interfaces**

**Type**  
list

**Default**  
[]

List of Ethernet Over InfiniBand interfaces on the Inspector host which are used for physical access to the DHCP network. Multiple interfaces would be attached to a bond or bridge specified in `dnsmasq_interface`. The MACs of the InfiniBand nodes which are not in desired state are going to be blocked based on the list of neighbor MACs on these interfaces.

**ip\_version**

**Type**  
string

**Default**  
4

**Valid Values**  
4, 6

The IP version that will be used for iptables filter. Defaults to 4.

**Possible values**

**4**  
IPv4

**6**  
IPv6

**ironic****auth\_url**

**Type**  
unknown type

**Default**  
<None>

Authentication URL

**auth\_type**

**Type**  
unknown type

**Default**  
<None>

Authentication type to load

Table 6: Deprecated Variations

Group	Name
ironic	auth_plugin

**cafile****Type**

string

**Default**

&lt;None&gt;

PEM encoded Certificate Authority to use when verifying HTTPs connections.

**certfile****Type**

string

**Default**

&lt;None&gt;

PEM encoded client certificate cert file

**collect\_timing****Type**

boolean

**Default**

False

Collect per-API call timing information.

**connect\_retries****Type**

integer

**Default**

&lt;None&gt;

The maximum number of retries that should be attempted for connection errors.

**connect\_retry\_delay****Type**

floating point

**Default**

&lt;None&gt;

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.

**default\_domain\_id****Type**

unknown type

**Default**

&lt;None&gt;

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**default\_domain\_name****Type**

unknown type

**Default**

&lt;None&gt;

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**domain\_id****Type**

unknown type

**Default**

&lt;None&gt;

Domain ID to scope to

**domain\_name****Type**

unknown type

**Default**

&lt;None&gt;

Domain name to scope to

**endpoint\_override****Type**

string

**Default**

&lt;None&gt;

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.

**insecure****Type**

boolean

**Default**

False

Verify HTTPS connections.

**keyfile**

**Type**  
string

**Default**  
<None>

PEM encoded client certificate key file

**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.

**max\_retries**

**Type**  
integer

**Default**  
30

Maximum number of retries in case of conflict error (HTTP 409).

**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.

**password**

**Type**  
unknown type

**Default**  
<None>

Users password

**project\_domain\_id**

**Type**  
unknown type

**Default**  
<None>

Domain ID containing project



**project\_domain\_name****Type**

unknown type

**Default**

&lt;None&gt;

Domain name containing project

**project\_id****Type**

unknown type

**Default**

&lt;None&gt;

Project ID to scope to

Table 7: Deprecated Variations

Group	Name
ironic	tenant-id
ironic	tenant_id

**project\_name****Type**

unknown type

**Default**

&lt;None&gt;

Project name to scope to

Table 8: Deprecated Variations

Group	Name
ironic	tenant-name
ironic	tenant_name

**region\_name****Type**

string

**Default**

&lt;None&gt;

The default region\_name for endpoint URL discovery.

**retriable\_status\_codes****Type**

list

**Default**

<None>

List of retrievable HTTP status codes that should be retried. If not set default to [503]

**retry\_interval**

**Type**

integer

**Default**

2

Interval between retries in case of conflict error (HTTP 409).

**service\_name**

**Type**

string

**Default**

<None>

The default service\_name for endpoint URL discovery.

**service\_type**

**Type**

string

**Default**

baremetal

The default service\_type for endpoint URL discovery.

**split\_loggers**

**Type**

boolean

**Default**

False

Log requests to multiple loggers.

**status\_code\_retries**

**Type**

integer

**Default**

<None>

The maximum number of retries that should be attempted for retrievable HTTP status codes.

**status\_code\_retry\_delay**

**Type**

floating point

**Default**

<None>

Delay (in seconds) between two retries for retrievable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

**system\_scope**

**Type**

unknown type

**Default**

<None>

Scope for system operations

**tenant\_id**

**Type**

unknown type

**Default**

<None>

Tenant ID

**tenant\_name**

**Type**

unknown type

**Default**

<None>

Tenant Name

**timeout**

**Type**

integer

**Default**

<None>

Timeout value for http requests

**trust\_id**

**Type**

unknown type

**Default**

<None>

ID of the trust to use as a trustee use

**user\_domain\_id**

**Type**

unknown type

**Default**

<None>

Users domain id

**user\_domain\_name**

**Type**  
unknown type

**Default**  
<None>

Users domain name

**user\_id**

**Type**  
unknown type

**Default**  
<None>

User id

**username**

**Type**  
unknown type

**Default**  
<None>

Username

Table 9: Deprecated Variations

Group	Name
ironic	user-name
ironic	user_name

**valid\_interfaces**

**Type**  
list

**Default**  
['internal', 'public']

List of interfaces, in order of preference, for endpoint URL.

**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

**keystone\_authtoken****www\_authenticate\_uri****Type**

string

**Default**

&lt;None&gt;

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 you're 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 10: Deprecated Variations

Group	Name
keystone_authtoken	auth_uri

**auth\_uri****Type**

string

**Default**

&lt;None&gt;

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 you're 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**

&lt;None&gt;

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 11: Deprecated Variations

Group	Name
keystone_authtoken	memcache_servers

**token\_cache\_time**

**Type**  
integer

**Default**  
300

In order to prevent excessive effort spent validating tokens, the middleware caches previously-seen tokens for a configurable duration (in seconds). Set to -1 to disable caching completely.

**memcache\_security\_strategy**

**Type**  
string

**Default**  
None

**Valid Values**  
None, MAC, ENCRYPT

(Optional) If defined, indicate whether token data should be authenticated or authenticated and encrypted. If MAC, token data is authenticated (with HMAC) in the cache. If ENCRYPT, token data is encrypted and authenticated in the cache. If the value is not one of these options or empty, `auth_token` will raise an exception on initialization.

**memcache\_secret\_key**

**Type**  
string

**Default**  
<None>

(Optional, mandatory if `memcache_security_strategy` is defined) This string is used for key derivation.

**memcache\_pool\_dead\_retry**

**Type**  
integer

**Default**  
300

(Optional) Number of seconds memcached server is considered dead before it is tried again.

**memcache\_pool\_maxsize**

**Type**  
integer

**Default**  
10

(Optional) Maximum total number of open connections to every memcached server.



**memcache\_pool\_socket\_timeout**

**Type**  
integer

**Default**  
3

(Optional) Socket timeout in seconds for communicating with a memcached server.

**memcache\_pool\_unused\_timeout**

**Type**  
integer

**Default**  
60

(Optional) Number of seconds a connection to memcached is held unused in the pool before it is closed.

**memcache\_pool\_conn\_get\_timeout**

**Type**  
integer

**Default**  
10

(Optional) Number of seconds that an operation will wait to get a memcached client connection from the pool.

**memcache\_use\_advanced\_pool**

**Type**  
boolean

**Default**  
True

(Optional) Use the advanced (eventlet safe) memcached client pool.

**include\_service\_catalog**

**Type**  
boolean

**Default**  
True

(Optional) Indicate whether to set the X-Service-Catalog header. If False, middleware will not ask for service catalog on token validation and will not set the X-Service-Catalog header.

**enforce\_token\_bind**

**Type**  
string

**Default**  
permissive

Used to control the use and type of token binding. Can be set to: disabled to not check token binding. permissive (default) to validate binding information if the bind type is of a form known to the server and ignore it if not. strict like permissive but if the bind type is unknown the token will be rejected. required any form of token binding is needed to be allowed. Finally the name of a binding method that must be present in tokens.

### **service\_token\_roles**

**Type**

list

**Default**

['service']

A choice of roles that must be present in a service token. Service tokens are allowed to request that an expired token can be used and so this check should tightly control that only actual services should be sending this token. Roles here are applied as an ANY check so any role in this list must be present. For backwards compatibility reasons this currently only affects the allow\_expired check.

### **service\_token\_roles\_required**

**Type**

boolean

**Default**

False

For backwards compatibility reasons we must let valid service tokens pass that dont pass the service\_token\_roles check as valid. Setting this true will become the default in a future release and should be enabled if possible.

### **service\_type**

**Type**

string

**Default**

<None>

The name or type of the service as it appears in the service catalog. This is used to validate tokens that have restricted access rules.

### **auth\_type**

**Type**

unknown type

**Default**

<None>

Authentication type to load

Table 12: Deprecated Variations

Group	Name
keystone_authtoken	auth_plugin

### **auth\_section**

**Type**

unknown type

**Default**

<None>

Config Section from which to load plugin specific options

### **mdns**

#### **registration\_attempts**

**Type**

integer

**Default**

5

**Minimum Value**

1

Number of attempts to register a service. Currently has to be larger than 1 because of race conditions in the zeroconf library.

#### **lookup\_attempts**

**Type**

integer

**Default**

3

**Minimum Value**

1

Number of attempts to lookup a service.

#### **params**

**Type**

unknown type

**Default**

{}

Additional parameters to pass for the registered service.

#### **interfaces**

**Type**

list

**Default**

<None>

List of IP addresses of interfaces to use for mDNS. Defaults to all interfaces on the system.

### **oslo\_messaging\_kafka**

#### **kafka\_max\_fetch\_bytes**

**Type**  
integer

**Default**  
1048576

Max fetch bytes of Kafka consumer

#### **kafka\_consumer\_timeout**

**Type**  
floating point

**Default**  
1.0

Default timeout(s) for Kafka consumers

#### **consumer\_group**

**Type**  
string

**Default**  
oslo\_messaging\_consumer

Group id for Kafka consumer. Consumers in one group will coordinate message consumption

#### **producer\_batch\_timeout**

**Type**  
floating point

**Default**  
0.0

Upper bound on the delay for KafkaProducer batching in seconds

#### **producer\_batch\_size**

**Type**  
integer

**Default**  
16384

Size of batch for the producer async send

#### **compression\_codec**

**Type**  
string

**Default**  
none

**Valid Values**  
none, gzip, snappy, lz4, zstd

The compression codec for all data generated by the producer. If not set, compression will not be used. Note that the allowed values of this depend on the kafka version

**enable\_auto\_commit**

**Type**

boolean

**Default**

False

Enable asynchronous consumer commits

**max\_poll\_records**

**Type**

integer

**Default**

500

The maximum number of records returned in a poll call

**security\_protocol**

**Type**

string

**Default**

PLAINTEXT

**Valid Values**

PLAINTEXT, SASL\_PLAINTEXT, SSL, SASL\_SSL

Protocol used to communicate with brokers

**sasl\_mechanism**

**Type**

string

**Default**

PLAIN

Mechanism when security protocol is SASL

**ssl\_cafile**

**Type**

string

**Default**

''

CA certificate PEM file used to verify the server certificate

**ssl\_client\_cert\_file**

**Type**

string

**Default**

''

Client certificate PEM file used for authentication.

**ssl\_client\_key\_file**

**Type**

string

**Default**

''

Client key PEM file used for authentication.

**ssl\_client\_key\_password**

**Type**

string

**Default**

''

Client key password file used for authentication.

**oslo\_messaging\_notifications**

**driver**

**Type**

multi-valued

**Default**

''

The Drivers(s) to handle sending notifications. Possible values are messaging, messagingv2, routing, log, test, noop

**transport\_url**

**Type**

string

**Default**

<None>

A URL representing the messaging driver to use for notifications. If not set, we fall back to the same configuration used for RPC.

**topics**

**Type**

list

**Default**

['notifications']

AMQP topic used for OpenStack notifications.

## **retry**

**Type**  
integer

**Default**  
-1

The maximum number of attempts to re-send a notification message which failed to be delivered due to a recoverable error. 0 - No retry, -1 - indefinite

## **oslo\_messaging\_rabbit**

### **amqp\_durable\_queues**

**Type**  
boolean

**Default**  
False

Use durable queues in AMQP. If rabbit\_quorum\_queue is enabled, queues will be durable and this value will be ignored.

### **amqp\_auto\_delete**

**Type**  
boolean

**Default**  
False

Auto-delete queues in AMQP.

### **rpc\_conn\_pool\_size**

**Type**  
integer

**Default**  
30

**Minimum Value**  
1

Size of RPC connection pool.

### **conn\_pool\_min\_size**

**Type**  
integer

**Default**  
2

The pool size limit for connections expiration policy

### **conn\_pool\_ttl**

**Type**  
integer

**Default**

1200

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

**ssl**

**Type**

boolean

**Default**

False

Connect over SSL.

**ssl\_version**

**Type**

string

**Default**

''

SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1\_1, and TLSv1\_2 may be available on some distributions.

**ssl\_key\_file**

**Type**

string

**Default**

''

SSL key file (valid only if SSL enabled).

**ssl\_cert\_file**

**Type**

string

**Default**

''

SSL cert file (valid only if SSL enabled).

**ssl\_ca\_file**

**Type**

string

**Default**

''

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

**ssl\_enforce\_fips\_mode**

**Type**

boolean



**Default**

False

Global toggle for enforcing the OpenSSL FIPS mode. This feature requires Python support. This is available in Python 3.9 in all environments and may have been backported to older Python versions on select environments. If the Python executable used does not support OpenSSL FIPS mode, an exception will be raised.

**heartbeat\_in\_pthread****Type**

boolean

**Default**

False

(DEPRECATED) It is recommend not to use this option anymore. Run the health check heartbeat thread through a native python thread by default. If this option is equal to False then the health check heartbeat will inherit the execution model from the parent process. For example if the parent process has monkey patched the stdlib by using eventlet/greenlet then the heartbeat will be run through a green thread. This option should be set to True only for the wsgi services.

**Warning**

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

**Reason**

The option is related to Eventlet which will be removed. In addition this has never worked as expected with services using eventlet for core service framework.

**kombu\_reconnect\_delay****Type**

floating point

**Default**

1.0

**Minimum Value**

0.0

**Maximum Value**

4.5

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

**kombu\_compression****Type**

string

**Default**

&lt;None&gt;

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 13: Deprecated Variations

Group	Name
oslo_messaging_rabbit	kombu_reconnect_timeout

**kombu\_failover\_strategy**

**Type**  
string

**Default**  
round-robin

**Valid Values**  
round-robin, shuffle

Determines how the next RabbitMQ node is chosen in case the one we are currently connected to becomes unavailable. Takes effect only if more than one RabbitMQ node is provided in config.

**rabbit\_login\_method**

**Type**  
string

**Default**  
AMQPLAIN

**Valid Values**  
PLAIN, AMQPLAIN, EXTERNAL, RABBIT-CR-DEMO

The RabbitMQ login method.

**rabbit\_retry\_interval**

**Type**  
integer

**Default**  
1

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.

**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}`

**rabbit\_quorum\_queue****Type**

boolean

**Default**

False

Use quorum queues in RabbitMQ (`x-queue-type: quorum`). The quorum queue is a modern queue type for RabbitMQ implementing a durable, replicated FIFO queue based on the Raft consensus algorithm. It is available as of RabbitMQ 3.8.0. If set this option will conflict with the HA queues (`rabbit_ha_queues`) aka mirrored queues, in other words the HA queues should be disabled. Quorum queues are also durable by default so the `amqp_durable_queues` option is ignored when this option is enabled.

**rabbit\_transient\_quorum\_queue****Type**

boolean

**Default**

False

Use quorum queues for transients queues in RabbitMQ. Enabling this option will then make sure those queues are also using quorum kind of rabbit queues, which are HA by default.

**rabbit\_quorum\_delivery\_limit****Type**

integer

**Default**

0

Each time a message is redelivered to a consumer, a counter is incremented. Once the redelivery count exceeds the delivery limit the message gets dropped or dead-lettered (if a DLX exchange has been configured) Used only when rabbit\_quorum\_queue is enabled, Default 0 which means dont set a limit.

**rabbit\_quorum\_max\_memory\_length**

**Type**

integer

**Default**

0

By default all messages are maintained in memory if a quorum queue grows in length it can put memory pressure on a cluster. This option can limit the number of messages in the quorum queue. Used only when rabbit\_quorum\_queue is enabled, Default 0 which means dont set a limit.

Table 14: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_quorum_max_memory_length

**rabbit\_quorum\_max\_memory\_bytes**

**Type**

integer

**Default**

0

By default all messages are maintained in memory if a quorum queue grows in length it can put memory pressure on a cluster. This option can limit the number of memory bytes used by the quorum queue. Used only when rabbit\_quorum\_queue is enabled, Default 0 which means dont set a limit.

Table 15: Deprecated Variations

Group	Name
oslo_messaging_rabbit	rabbit_quorum_max_memory_bytes

**rabbit\_transient\_queues\_ttl**

**Type**

integer

**Default**

1800

**Minimum Value**

0

Positive integer representing duration in seconds for queue TTL (x-expires). Queues which are unused for the duration of the TTL are automatically deleted. The parameter affects only reply

and fanout queues. Setting 0 as value will disable the x-expires. If doing so, make sure you have a rabbitmq policy to delete the queues or your deployment will create an infinite number of queues over time. In case `rabbit_stream_fanout` is set to True, this option will control data retention policy (x-max-age) for messages in the fanout queue rather than the queue duration itself. So the oldest data in the stream queue will be discarded from it once reaching TTL. Setting to 0 will disable x-max-age for stream which makes stream grow indefinitely filling up the disk space.

#### `rabbit_qos_prefetch_count`

**Type**

integer

**Default**

0

Specifies the number of messages to prefetch. Setting to zero allows unlimited messages.

#### `heartbeat_timeout_threshold`

**Type**

integer

**Default**

60

Number of seconds after which the Rabbit broker is considered down if heartbeats keep-alive fails (0 disables heartbeat).

#### `heartbeat_rate`

**Type**

integer

**Default**

3

How often times during the `heartbeat_timeout_threshold` we check the heartbeat.

#### `direct_mandatory_flag`

**Type**

boolean

**Default**

True

(DEPRECATED) Enable/Disable the RabbitMQ mandatory flag for direct send. The direct send is used as reply, so the `MessageUndeliverable` exception is raised in case the client queue does not exist. `MessageUndeliverable` exception will be used to loop for a timeout to let 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 consumers when queue is down

### **use\_queue\_manager**

**Type**

boolean

**Default**

False

Should we use consistent queue names or random ones

### **hostname**

**Type**

string

**Default**

node1.example.com

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

Hostname used by queue manager. Defaults to the value returned by `socket.gethostname()`.

### **processname**

**Type**

string

**Default**

nova-api

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

Process name used by queue manager

### **rabbit\_stream\_fanout**

**Type**

boolean

**Default**

False

Use stream queues in RabbitMQ (x-queue-type: stream). Streams are a new persistent and replicated data structure (queue type) in RabbitMQ which models an append-only log with non-destructive consumer semantics. It is available as of RabbitMQ 3.9.0. If set this option will replace all fanout queues with only one stream queue.

## oslo\_policy

### enforce\_scope

**Type**

boolean

**Default**

True

This option controls whether or not to enforce scope when evaluating policies. If `True`, the scope of the token used in the request is compared to the `scope_types` of the policy being enforced. If the scopes do not match, an `InvalidScope` exception will be raised. If `False`, a message will be logged informing operators that policies are being invoked with mismatching scope.

**Warning**

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

**Reason**

This configuration was added temporarily to facilitate a smooth transition to the new RBAC. OpenStack will always enforce scope checks. This configuration option is deprecated and will be removed in the 2025.2 cycle.

### enforce\_new\_defaults

**Type**

boolean

**Default**

True

This option controls whether or not to use old deprecated defaults when evaluating policies. If `True`, the old deprecated defaults are not going to be evaluated. This means if any existing token is allowed for old defaults but is disallowed for new defaults, it will be disallowed. It is encouraged to enable this flag along with the `enforce_scope` flag so that you can get the benefits of new defaults and `scope_type` together. If `False`, the deprecated policy check string is logically OR'd with the new policy check string, allowing for a graceful upgrade experience between releases with new policies, which is the default behavior.

### policy\_file

**Type**

string

**Default**

policy.yaml

The relative or absolute path of a file that maps roles to permissions for a given service. Relative paths must be specified in relation to the configuration file setting this option.

### policy\_default\_rule

**Type**

string

**Default**

default

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

**policy\_dirs**

**Type**

multi-valued

**Default**

policy.d

Directories where policy configuration files are stored. They can be relative to any directory in the search path defined by the `config_dir` option, or absolute paths. The file defined by `policy_file` must exist for these directories to be searched. Missing or empty directories are ignored.

**remote\_content\_type**

**Type**

string

**Default**

application/x-www-form-urlencoded

**Valid Values**

application/x-www-form-urlencoded, application/json

Content Type to send and receive data for REST based policy check

**remote\_ssl\_verify\_server\_cert**

**Type**

boolean

**Default**

False

server identity verification for REST based policy check

**remote\_ssl\_ca\_cert\_file**

**Type**

string

**Default**

<None>

Absolute path to ca cert file for REST based policy check

**remote\_ssl\_client\_cert\_file**

**Type**

string

**Default**

<None>

Absolute path to client cert for REST based policy check



**remote\_ssl\_client\_key\_file****Type**

string

**Default**

&lt;None&gt;

Absolute path client key file REST based policy check

**remote\_timeout****Type**

floating point

**Default**

60

**Minimum Value**

0

Timeout in seconds for REST based policy check

**pci\_devices****alias****Type**

multi-valued

**Default**

''

An alias for PCI device identified by vendor\_id and product\_id fields. Format: {vendor\_id: 1234, product\_id: 5678, name: pci\_dev1}

**port\_physnet****cidr\_map****Type**

list

**Default**

10.10.10.0/24:physnet\_a,2001:db8::/64:physnet\_b

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

Mapping of IP subnet CIDR to physical network. When the physnet\_cidr\_map processing hook is enabled the physical\_network property of baremetal ports is populated based on this mapping.

**processing****add\_ports****Type**

string

**Default**

pxe

**Valid Values**

all, active, pxe, disabled

Which MAC addresses to add as ports during introspection. Possible values: all (all MAC addresses), active (MAC addresses of NIC with IP addresses), pxe (only MAC address of NIC node PXE booted from, falls back to active if PXE MAC is not supplied by the ramdisk).

**keep\_ports**

**Type**

string

**Default**

all

**Valid Values**

all, present, added

Which ports (already present on a node) to keep after introspection. Possible values: all (do not delete anything), present (keep ports which MACs were present in introspection data), added (keep only MACs that we added during introspection).

**overwrite\_existing**

**Type**

boolean

**Default**

True

Whether to overwrite existing values in node database. Disable this option to make introspection a non-destructive operation.

**default\_processing\_hooks**

**Type**

string

**Default**

ramdisk\_error,root\_disk\_selection,scheduler,  
validate\_interfaces,capabilities,pci\_devices

Comma-separated list of default hooks for processing pipeline. Hook scheduler updates the node with the minimum properties required by the Nova scheduler. Hook validate\_interfaces ensures that valid NIC data was provided by the ramdisk. Do not exclude these two unless you really know what youre doing.

**processing\_hooks**

**Type**

string

**Default**

`$default_processing_hooks`

Comma-separated list of enabled hooks for processing pipeline. The default for this is `$default_processing_hooks`, hooks can be added before or after the defaults like this: `pre-hook,$default_processing_hooks,posthook`.

#### **ramdisk\_logs\_dir**

**Type**  
string

**Default**  
<None>

If set, logs from ramdisk will be stored in this directory.

#### **always\_store\_ramdisk\_logs**

**Type**  
boolean

**Default**  
False

Whether to store ramdisk logs even if it did not return an error message (dependent upon `ramdisk_logs_dir` option being set).

#### **node\_not\_found\_hook**

**Type**  
string

**Default**  
<None>

The name of the hook to run when inspector receives inspection information from a node it isn't already aware of. This hook is ignored by default.

#### **store\_data**

**Type**  
string

**Default**  
none

The storage backend for storing introspection data. Possible values are: none, database and swift. If set to none, introspection data will not be stored.

#### **disk\_partitioning\_spacing**

**Type**  
boolean

**Default**  
True

Whether to leave 1 GiB of disk size untouched for partitioning. Only has effect when used with the IPA as a ramdisk, for older ramdisk `local_gb` is calculated on the ramdisk side.

### **ramdisk\_logs\_filename\_format**

**Type**

string

**Default**

{uuid}\_{dt:%Y%m%d-%H%M%S.%f}.tar.gz

File name template for storing ramdisk logs. The following replacements can be used: {uuid} - node UUID or unknown, {bmc} - node BMC address or unknown, {dt} - current UTC date and time, {mac} - PXE booting MAC or unknown.

### **power\_off**

**Type**

boolean

**Default**

True

Whether to power off a node after introspection. Nodes in active or rescue states which submit introspection data will be left on if the feature is enabled via the `permit_active_introspection` configuration option.

### **permit\_active\_introspection**

**Type**

boolean

**Default**

False

Whether to process nodes that are in running states.

### **update\_pxe\_enabled**

**Type**

boolean

**Default**

True

Whether to update the `pxe_enabled` value according to the introspection data. This option has no effect if `[processing]overwrite_existing` is set to False

### **pxe\_filter**

#### **driver**

**Type**

string

**Default**

iptables

PXE boot filter driver to use, possible filters are: `iptables`, `dnsmasq` and `noop`. Set `noop` to disable the firewall filtering.

**sync\_period****Type**  
integer**Default**  
15**Minimum Value**  
0

Amount of time in seconds, after which repeat periodic update of the filter.

**deny\_unknown\_macs****Type**  
boolean**Default**  
False

By default inspector will open the DHCP server for any node when introspection is active. Opening DHCP for unknown MAC addresses when introspection is active allow for users to add nodes with no ports to ironic and have ironic-inspector enroll ports based on node introspection results. NOTE: If this option is True, nodes must have at least one enrolled port prior to introspection.

**service\_catalog****auth\_url****Type**  
unknown type**Default**  
<None>

Authentication URL

**auth\_type****Type**  
unknown type**Default**  
<None>

Authentication type to load

Table 16: Deprecated Variations

Group	Name
service_catalog	auth_plugin

**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

**collect\_timing**

**Type**

boolean

**Default**

False

Collect per-API call timing information.

**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.

**default\_domain\_id**

**Type**

unknown type

**Default**

<None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**default\_domain\_name**

**Type**

unknown type

**Default**

<None>

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**domain\_id**

**Type**

unknown type

**Default**

<None>

Domain ID to scope to

**domain\_name**

**Type**

unknown type

**Default**

<None>

Domain name to scope to

**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.

**insecure**

**Type**

boolean

**Default**

False

Verify HTTPS connections.

**keyfile**

**Type**

string

**Default**

<None>

PEM encoded client certificate key file

**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`.

**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.

**password**

**Type**

unknown type

**Default**

<None>

Users password

**project\_domain\_id**

**Type**

unknown type

**Default**

<None>

Domain ID containing project

**project\_domain\_name**

**Type**

unknown type

**Default**

<None>

Domain name containing project

**project\_id**

**Type**

unknown type

**Default**

<None>

Project ID to scope to



Table 17: Deprecated Variations

Group	Name
service_catalog	tenant-id
service_catalog	tenant_id

**project\_name**

**Type**

unknown type

**Default**

<None>

Project name to scope to

Table 18: Deprecated Variations

Group	Name
service_catalog	tenant-name
service_catalog	tenant_name

**region\_name**

**Type**

string

**Default**

<None>

The default region\_name for endpoint URL discovery.

**retriable\_status\_codes**

**Type**

list

**Default**

<None>

List of retriable HTTP status codes that should be retried. If not set default to [503]

**service\_name**

**Type**

string

**Default**

<None>

The default service\_name for endpoint URL discovery.

**service\_type**

**Type**

string

**Default**

baremetal-introspection

The default service\_type for endpoint URL discovery.

**split\_loggers**

**Type**

boolean

**Default**

False

Log requests to multiple loggers.

**status\_code\_retries**

**Type**

integer

**Default**

<None>

The maximum number of retries that should be attempted for retrieable HTTP status codes.

**status\_code\_retry\_delay**

**Type**

floating point

**Default**

<None>

Delay (in seconds) between two retries for retrieable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

**system\_scope**

**Type**

unknown type

**Default**

<None>

Scope for system operations

**tenant\_id**

**Type**

unknown type

**Default**

<None>

Tenant ID

**tenant\_name**

**Type**

unknown type

**Default**

<None>

Tenant Name

**timeout**

**Type**

integer

**Default**

<None>

Timeout value for http requests

**trust\_id**

**Type**

unknown type

**Default**

<None>

ID of the trust to use as a trustee use

**user\_domain\_id**

**Type**

unknown type

**Default**

<None>

Users domain id

**user\_domain\_name**

**Type**

unknown type

**Default**

<None>

Users domain name

**user\_id**

**Type**

unknown type

**Default**

<None>

User id

**username**

**Type**

unknown type

**Default**

<None>

Username

Table 19: Deprecated Variations

Group	Name
service_catalog	user-name
service_catalog	user_name

### valid\_interfaces

**Type**

list

**Default**

['internal', 'public']

List of interfaces, in order of preference, for endpoint URL.

### 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

### ssl

#### ca\_file

**Type**

string

**Default**

<None>

CA certificate file to use to verify connecting clients.

#### cert\_file

**Type**

string

**Default**

<None>

Certificate file to use when starting the server securely.

#### key\_file

**Type**

string

**Default**

<None>

Private key file to use when starting the server securely.

**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.

**swift**

**auth\_url**

**Type**

unknown type

**Default**

<None>

Authentication URL

**auth\_type**

**Type**

unknown type

**Default**

<None>

Authentication type to load

Table 20: Deprecated Variations

Group	Name
swift	auth_plugin

**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

### **collect\_timing**

**Type**

boolean

**Default**

False

Collect per-API call timing information.

### **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.

### **container**

**Type**

string

**Default**

ironic-inspector

Default Swift container to use when creating objects.

### **default\_domain\_id**

**Type**

unknown type

**Default**

<None>

Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**default\_domain\_name****Type**

unknown type

**Default**

&lt;None&gt;

Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

**delete\_after****Type**

integer

**Default**

0

Number of seconds that the Swift object will last before being deleted. (set to 0 to never delete the object).

**domain\_id****Type**

unknown type

**Default**

&lt;None&gt;

Domain ID to scope to

**domain\_name****Type**

unknown type

**Default**

&lt;None&gt;

Domain name to scope to

**endpoint\_override****Type**

string

**Default**

&lt;None&gt;

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.

**insecure****Type**

boolean

**Default**

False

Verify HTTPS connections.

**keyfile**

**Type**

string

**Default**

<None>

PEM encoded client certificate key file

**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.

**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.

**password**

**Type**

unknown type

**Default**

<None>

Users password

**project\_domain\_id**

**Type**

unknown type

**Default**

<None>

Domain ID containing project

**project\_domain\_name**

**Type**

unknown type

**Default**

<None>



Domain name containing project

### **project\_id**

**Type**

unknown type

**Default**

<None>

Project ID to scope to

Table 21: Deprecated Variations

Group	Name
swift	tenant-id
swift	tenant_id

### **project\_name**

**Type**

unknown type

**Default**

<None>

Project name to scope to

Table 22: Deprecated Variations

Group	Name
swift	tenant-name
swift	tenant_name

### **region\_name**

**Type**

string

**Default**

<None>

The default region\_name for endpoint URL discovery.

### **retriable\_status\_codes**

**Type**

list

**Default**

<None>

List of retriable HTTP status codes that should be retried. If not set default to [503]

### **service\_name**

**Type**  
string

**Default**  
<None>

The default service\_name for endpoint URL discovery.

### **service\_type**

**Type**  
string

**Default**  
object-store

The default service\_type for endpoint URL discovery.

### **split\_loggers**

**Type**  
boolean

**Default**  
False

Log requests to multiple loggers.

### **status\_code\_retries**

**Type**  
integer

**Default**  
<None>

The maximum number of retries that should be attempted for retrieable HTTP status codes.

### **status\_code\_retry\_delay**

**Type**  
floating point

**Default**  
<None>

Delay (in seconds) between two retries for retrieable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

### **system\_scope**

**Type**  
unknown type

**Default**  
<None>

Scope for system operations

**tenant\_id**

**Type**

unknown type

**Default**

<None>

Tenant ID

**tenant\_name**

**Type**

unknown type

**Default**

<None>

Tenant Name

**timeout**

**Type**

integer

**Default**

<None>

Timeout value for http requests

**trust\_id**

**Type**

unknown type

**Default**

<None>

ID of the trust to use as a trustee use

**user\_domain\_id**

**Type**

unknown type

**Default**

<None>

Users domain id

**user\_domain\_name**

**Type**

unknown type

**Default**

<None>

Users domain name

**user\_id****Type**

unknown type

**Default**

&lt;None&gt;

User id

**username****Type**

unknown type

**Default**

&lt;None&gt;

Username

Table 23: Deprecated Variations

Group	Name
swift	user-name
swift	user_name

**valid\_interfaces****Type**

list

**Default**

['internal', 'public']

List of interfaces, in order of preference, for endpoint URL.

**version****Type**

string

**Default**

&lt;None&gt;

Minimum Major API version within a given Major API version for endpoint URL discovery. Mutually exclusive with `min_version` and `max_version`

### 3.3.2 Policies

**Warning**

JSON formatted policy files were deprecated in the Wallaby development cycle due to the Victoria deprecation by the `olso.policy` library. Use the [oslopolicy-convert-json-to-yaml](#) tool to convert the existing JSON to YAML formatted policy file in backward compatible way.

The following is an overview of all available policies in **ironic inspector**. For a sample configuration file, refer to *Ironic Inspector Policy*.

### **ironic\_inspector.api**

#### **is\_admin**

##### **Default**

role:admin or role:administrator or role:baremetal\_admin

Full read/write API access

#### **is\_observer**

##### **Default**

role:baremetal\_observer

Read-only API access

#### **public\_api**

##### **Default**

is\_public\_api:True

Internal flag for public API routes

#### **default**

##### **Default**

!

Default API access policy

#### **introspection**

##### **Default**

rule:public\_api

##### **Operations**

- GET /

Access the API root for available versions information

#### **introspection:version**

##### **Default**

rule:public\_api

##### **Operations**

- GET /{version}

Access the versioned API root for version information

#### **introspection:continue**

##### **Default**

rule:public\_api

##### **Operations**

- POST /continue

Ramdisk callback to continue introspection

### **introspection:status**

#### **Default**

(role:reader and system\_scope:all) or (role:admin) or (role:service)

#### **Operations**

- **GET** /introspection
- **GET** /introspection/{node\_id}

Get introspection status

### **introspection:start**

#### **Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

#### **Operations**

- **POST** /introspection/{node\_id}

Start introspection

### **introspection:abort**

#### **Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

#### **Operations**

- **POST** /introspection/{node\_id}/abort

Abort introspection

### **introspection:data**

#### **Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

#### **Operations**

- **GET** /introspection/{node\_id}/data

Get introspection data

### **introspection:reapply**

#### **Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

#### **Operations**

- **POST** /introspection/{node\_id}/data/unprocessed

Reapply introspection on stored data

### **introspection:rule:get**

**Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

**Operations**

- **GET** /rules
- **GET** /rules/{rule\_id}

Get introspection rule(s)

**introspection:rule:delete****Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

**Operations**

- **DELETE** /rules
- **DELETE** /rules/{rule\_id}

Delete introspection rule(s)

**introspection:rule:create****Default**

(role:admin and system\_scope:all) or (role:admin) or (role:service)

**Operations**

- **POST** /rules

Create introspection rule

## 3.4 User Guide

### 3.4.1 How Ironic Inspector Works

#### How Ironic Inspector Works

##### Workflow

Usual hardware introspection flow is as follows:

- Operator enrolls nodes into **Ironic** e.g. via **baremetal CLI** command. Power management credentials should be provided to Ironic at this step.
- Nodes are put in the correct state for introspection as described in *node states*.
- Operator sends nodes on introspection using **ironic-inspector** API or CLI (see *usage*).
- On receiving node UUID **ironic-inspector**:
  - validates node power credentials, current power and provisioning states,
  - allows access to PXE boot service for the nodes,
  - issues reboot command for the nodes, so that they boot the ramdisk.

- The ramdisk collects the required information and posts it back to **ironic-inspector**.
- On receiving data from the ramdisk, **ironic-inspector**:
  - validates received data,
  - finds the node in Ironic database using its BMC address (MAC address in case of SSH driver),
  - fills missing node properties with received data and creates missing ports.

### Note

**ironic-inspector** is responsible to create Ironic ports for some or all NICs found on the node. **ironic-inspector** is also capable of deleting ports that should not be present. There are two important configuration options that affect this behavior: `add_ports` and `keep_ports` (please refer to *the sample configuration file* for a detailed explanation).

Default values as of **ironic-inspector** 1.1.0 are `add_ports=pxe`, `keep_ports=all`, which means that only one port will be added, which is associated with NIC the ramdisk PXE booted from. No ports will be deleted. This setting ensures that deploying on introspected nodes will succeed despite [Ironic bug 1405131](#).

Ironic inspection feature by default requires different settings: `add_ports=all`, `keep_ports=present`, which means that ports will be created for all detected NICs, and all other ports will be deleted. Refer to the [Ironic inspection documentation](#) for details.

Ironic inspector can also be configured to not create any ports. This is done by setting `add_ports=disabled`. If setting `add_ports` to disabled the `keep_ports` option should be also set to `all`. This will ensure no manually added ports will be deleted.

- Separate API (see [usage](#) and [API reference](#)) can be used to query introspection results for a given node.
- Nodes are put in the correct state for deploying as described in [node states](#).

Starting DHCP server and configuring PXE boot environment is not part of this package and should be done separately.

## State machine diagram

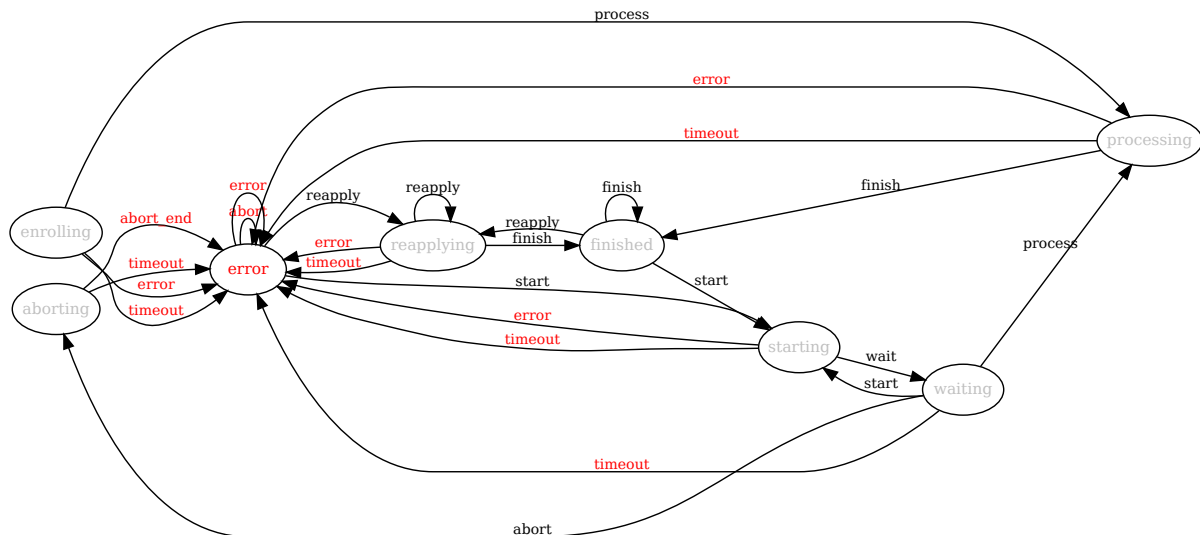
The diagram below shows the introspection states that an **ironic-inspector** FSM goes through during the node introspection, discovery and reprocessing. The diagram also shows events that trigger state transitions.

### 3.4.2 How to use Ironic Inspector

#### Usage

Refer to the [API reference](#) for information on the HTTP API. Refer to the [client documentation](#) for information on how to use CLI and Python library.





## Using from Ironic API

Ironic Kilo introduced support for hardware introspection under name of inspection. **ironic-inspector** introspection is supported for some generic drivers, please refer to [Ironic inspection documentation](#) for details.

## Node States

- The nodes should be moved to **MANAGEABLE** provision state before introspection (requires *python-ironicclient* of version 0.5.0 or newer):

```
baremetal node manage <node>
```

- The introspection can be triggered by using the following command:

```
baremetal node inspect <node>
```

- After successful introspection and before deploying nodes should be made available to Nova, by moving them to **AVAILABLE** state:

```
baremetal node provide <node>
```

### Note

Due to how Nova interacts with Ironic driver, you should wait 1 minute before Nova becomes aware of available nodes after issuing this command. Use `nova hypervisor-stats` command output to check it.

## Introspection Rules

Inspector supports a simple JSON-based DSL to define rules to run during introspection. Inspector provides an API to manage such rules, and will run them automatically after running all processing hooks.

A rule consists of conditions to check, and actions to run. If conditions evaluate to true on the introspection data, then actions are run on a node.

Please refer to the command below to import introspection rule:

```
baremetal introspection rule import <json file>
```

Available conditions and actions are defined by plugins, and can be extended, see *How To Contribute* for details. See the [API reference](#) for specific calls to define introspection rules.

### Conditions

A condition is represented by an object with fields:

`op` the type of comparison operation, default available operators include:

- `eq`, `le`, `ge`, `ne`, `lt`, `gt` - basic comparison operators;
- `in-net` - checks that an IP address is in a given network;
- `matches` - requires a full match against a given regular expression;
- `contains` - requires a value to contain a given regular expression;
- `is-empty` - checks that field is an empty string, list, dict or None value.

`field` a [JSON path](#) to the field in the introspection data to use in comparison.

Starting with the Mitaka release, you can also apply conditions to ironic node field. Prefix field with schema (`data://` or `node://`) to distinguish between values from introspection data and node. Both schemes use JSON path:

```
{"field": "node://property.path", "op": "eq", "value": "val"}  
{"field": "data://introspection.path", "op": "eq", "value": "val"}
```

if scheme (node or data) is missing, condition compares data with introspection data.

`invert` boolean value, whether to invert the result of the comparison.

`multiple` how to treat situations where the `field` query returns multiple results (e.g. the field contains a list), available options are:

- `any` (the default) require any to match,
- `all` require all to match,
- `first` require the first to match.

All other fields are passed to the condition plugin, e.g. numeric comparison operations require a `value` field to compare against.

### Scope

By default, introspection rules are applied to all nodes being inspected. In order for the rule to be applied only to specific nodes, a matching scope variable must be set to both the rule and the node. To set the scope for a rule include field `"scope"` in JSON file before importing. For example:

```
cat <json file>
{
  "description": "...",
  "actions": [...],
  "conditions": [...],
  "scope": "SCOPE"
}
```

Set the property `inspection_scope` on the node you want the rule to be applied to:

```
baremetal node set --property inspection_scope="SCOPE" <node>
```

Now, when inspecting, the rule will be applied only to nodes with matching scope value. It will also ignore nodes that do not have `inspection_scope` property set. Note that if a rule has no scope set, it will be applied to all nodes, regardless if they have `inspection_scope` set or not.

## Actions

An action is represented by an object with fields:

`action` type of action. Possible values are defined by plugins.

All other fields are passed to the action plugin.

Default available actions include:

- `fail` fail introspection. Requires a `message` parameter for the failure message.
- `set-attribute` sets an attribute on an Ironic node. Requires a `path` field, which is the path to the attribute as used by ironic (e.g. `/properties/something`), and a `value` to set.
- `set-capability` sets a capability on an Ironic node. Requires `name` and `value` fields, which are the name and the value for a new capability accordingly. Existing value for this same capability is replaced.
- `extend-attribute` the same as `set-attribute`, but treats existing value as a list and appends value to it. If optional `unique` parameter is set to `True`, nothing will be added if given value is already in a list.
- `add-trait` adds a trait to an Ironic node. Requires a `name` field with the name of the trait to add.
- `remove-trait` removes a trait from an Ironic node. Requires a `name` field with the name of the trait to remove.

Starting from Mitaka release, `value` field in actions supports fetching data from introspection, using `python string formatting notation`:

```
{"action": "set-attribute", "path": "/driver_info/ipmi_address",
 "value": "{data[inventory][bmc_address]}"}}
```

Note that any value referenced in this way will be converted to a string.

If `value` is a dict or list, strings nested at any level within the structure will be formatted as well:

```
{"action": "set-attribute", "path": "/properties/root_device",
 "value": {"serial": "{data[root_device][serial]}"}]}
```

### Plugins

**ironic-inspector** heavily relies on plugins for data processing. Even the standard functionality is largely based on plugins. Set `processing_hooks` option in the configuration file to change the set of plugins to be run on introspection data. Note that order **does** matter in this option, especially for hooks that have dependencies on other hooks.

These are plugins that are enabled by default and should not be disabled, unless you understand what you're doing:

#### **scheduler**

validates and updates basic hardware scheduling properties: CPU number and architecture, memory and disk size.

#### Note

Diskless nodes have the disk size property `local_gb == 0`. Always use node driver `root_device` hints to prevent unexpected HW failures passing silently.

#### **validate\_interfaces**

validates network interfaces information. Creates new ports, optionally deletes ports that were not present in the introspection data. Also sets the `pxe_enabled` flag for the PXE-booting port and unsets it for all the other ports to avoid **nova** picking a random port to boot the node.

#### Note

When the `pxe_filter` is configured to only open the DHCP server for known MAC addresses, i.e the `[pxe_filter]deny_unknown_macs` configuration option is enabled, it is not possible to rely on the `validate_interfaces` processing plug-in to create the PXE-booting port in **ironic**. Nodes must have at least one enrolled port prior to introspection in this case.

The following plugins are enabled by default, but can be disabled if not needed:

#### **ramdisk\_error**

reports error, if `error` field is set by the ramdisk, also optionally stores logs from `logs` field, see the [API reference](#) for details.

#### **capabilities**

detect node capabilities: CPU, boot mode, etc. See *Capabilities Detection* for more details.

#### **pci\_devices**

gathers the list of all PCI devices returned by the ramdisk and compares to those defined in `alias` field(s) from `pci_devices` section of configuration file. The recognized PCI devices and their count are then stored in node properties. This information can be later used in **nova** flavors for node scheduling.

Here are some plugins that can be additionally enabled:

#### **example**

example plugin logging its input and output.

#### **raid\_device**

gathers block devices from ramdisk and exposes root device in multiple runs.

**extra\_hardware**

stores the value of the data key returned by the ramdisk as a JSON encoded string in a Swift object. The plugin will also attempt to convert the data into a format usable by introspection rules. If this is successful then the new format will be stored in the extra key. The data key is then deleted from the introspection data, as unless converted its assumed unusable by introspection rules.

**lldp\_basic**

Processes LLDP data returned from inspection, parses TLVs from the Basic Management (802.1AB), 802.1Q, and 802.3 sets and stores the processed data back in the Ironic inspector database. To enable LLDP in the inventory from IPA, `ipa-collect-lldp=1` should be passed as a kernel parameter to the IPA ramdisk.

**local\_link\_connection**

Processes LLDP data returned from inspection, specifically looking for the port ID and chassis ID. If found, it configures the local link connection information on the Ironic ports with that data. To enable LLDP in the inventory from IPA, `ipa-collect-lldp=1` should be passed as a kernel parameter to the IPA ramdisk. In order to avoid processing the raw LLDP data twice, the `lldp_basic` plugin should also be installed and run prior to this plugin.

**physnet\_cidr\_map**

Configures the `physical_network` property of the nodes Ironic port when the IP address is in a configured CIDR mapping. CIDR to physical network mappings is set in configuration using the `[port_physnet]/cidr_map` option, for example:

```
[port_physnet]
cidr_map = 10.10.10.0/24:physnet_a, 2001:db8::/64:physnet_b
```

**accelerators**

Processes PCI data returned from inspection and compares with the accelerator inventory, it will update accelerator device information to the properties field of the ironic node if any accelerator device is found, for example:

```
{'local_gb': '1115', 'cpus': '40', 'cpu_arch': 'x86_64', 'memory_mb':
↪'32768',
  'capabilities': 'boot_mode:bios,cpu_vt:true,cpu_aes:true,cpu_
↪hugepages:true,cpu_hugepages_1g:true,cpu_txt:true',
  'accel': [{ 'vendor_id': '10de', 'device_id': '1eb8', 'type': 'GPU',
              'pci_address': '0000:82:00.0',
              'device_info': 'NVIDIA Corporation Tesla T4' }]}
```

Refer to *How To Contribute* for information on how to write your own plugin.

**Discovery**

Starting from Mitaka, **ironic-inspector** is able to register new nodes in Ironic.

The existing `node-not-found-hook` handles what happens if **ironic-inspector** receives inspection data from a node it can not identify. This can happen if a node is manually booted without registering it with Ironic first.

For discovery, the configuration file option `node_not_found_hook` should be set to load the hook called `enroll`. This hook will enroll the unidentified node into Ironic using the `fake-hardware` hardware type.

This is a configurable option: set `enroll_node_driver` in the **ironic-inspector** configuration file to the hardware type you want. You can also configure arbitrary fields to set on discovery, for example:

```
[discovery]
enroll_node_driver = ipmi
enroll_node_fields = management_interface:noop,resource_class:baremetal
```

The `enroll` hook will also set the `ipmi_address` property on the new node, if its available in the introspection data we received, see [ramdisk callback](#).

Once the `enroll` hook is finished, **ironic-inspector** will process the introspection data in the same way it would for an identified node. It runs the processing *plugins*, and after that it runs introspection rules, which would allow for more customisable node configuration, see *rules*.

A rule to set a nodes Ironic driver to `ipmi` and populate the required `driver_info` for that driver would look like:

```
[{
  "description": "Set IPMI driver_info if no credentials",
  "actions": [
    {"action": "set-attribute", "path": "driver", "value": "ipmi"},
    {"action": "set-attribute", "path": "driver_info/ipmi_username",
     "value": "username"},
    {"action": "set-attribute", "path": "driver_info/ipmi_password",
     "value": "password"}
  ],
  "conditions": [
    {"op": "is-empty", "field": "node://driver_info.ipmi_password"},
    {"op": "is-empty", "field": "node://driver_info.ipmi_username"}
  ]
}, {
  "description": "Set deploy info if not already set on node",
  "actions": [
    {"action": "set-attribute", "path": "driver_info/deploy_kernel",
     "value": "<glance uuid>"},
    {"action": "set-attribute", "path": "driver_info/deploy_ramdisk",
     "value": "<glance uuid>"}
  ],
  "conditions": [
    {"op": "is-empty", "field": "node://driver_info.deploy_ramdisk"},
    {"op": "is-empty", "field": "node://driver_info.deploy_kernel"}
  ]
}]
```

All nodes discovered and enrolled via the `enroll` hook, will contain an `auto_discovered` flag in the introspection data, this flag makes it possible to distinguish between manually enrolled nodes and auto-discovered nodes in the introspection rules using the rule condition eq:

```
{
  "description": "Enroll auto-discovered nodes with ipmi hardware type",
  "actions": [
    {"action": "set-attribute", "path": "driver", "value": "ipmi"}
  ]
}
```

(continues on next page)

(continued from previous page)

```
],  
  "conditions": [  
    {"op": "eq", "field": "data://auto_discovered", "value": true}  
  ]  
}
```

### Reapplying introspection on stored data

To allow correcting mistakes in introspection rules the API provides an entry point that triggers the introspection over stored data. The data to use for processing is kept in Swift separately from the data already processed. Reapplying introspection overwrites processed data in the store. Updating the introspection data through the endpoint isnt supported yet. Following preconditions are checked before reapplying introspection:

- no data is being sent along with the request
- Swift store is configured and enabled
- introspection data is stored in Swift for the node UUID
- node record is kept in database for the UUID
- introspection is not ongoing for the node UUID

Should the preconditions fail an immediate response is given to the user:

- 400 if the request contained data or in case Swift store is not enabled in configuration
- 404 in case Ironic doesnt keep track of the node UUID
- 409 if an introspection is already ongoing for the node

If the preconditions are met a background task is executed to carry out the processing and a 202 Accepted response is returned to the endpoint user. As requested, these steps are performed in the background task:

- preprocessing hooks
- post processing hooks, storing result in Swift
- introspection rules

These steps are avoided, based on the feature requirements:

- `node_not_found_hook` is skipped
- power operations
- roll-back actions done by hooks

Limitations:

- theres no way to update the unprocessed data atm.
- the unprocessed data is never cleaned from the store
- check for stored data presence is performed in background; missing data situation still results in a 202 response

### Capabilities Detection

Starting with the Newton release, **Ironic Inspector** can optionally discover several node capabilities. A recent (Newton or newer) IPA image is required for it to work.

#### Boot mode

The current boot mode (BIOS or UEFI) can be detected and recorded as `boot_mode` capability in Ironic. It will make some drivers to change their behaviour to account for this capability. Set the `capabilities.boot_mode` configuration option to `True` to enable.

#### CPU capabilities

Several CPU flags are detected by default and recorded as following capabilities:

- `cpu_aes` AES instructions.
- `cpu_vt` virtualization support.
- `cpu_txt` TXT support.
- `cpu_hugepages` huge pages (2 MiB) support.
- `cpu_hugepages_1g` huge pages (1 GiB) support.

It is possible to define your own rules for detecting CPU capabilities. Set the `capabilities.cpu_flags` configuration option to a mapping between a CPU flag and a capability, for example:

```
cpu_flags = aes:cpu_aes,svm:cpu_vt,vmx:cpu_vt
```

See the default value of this option for a more detail example.

#### InfiniBand support

Starting with the Ocata release, **Ironic Inspector** supports detection of InfiniBand network interfaces. A recent (Ocata or newer) IPA image is required for that to work. When an InfiniBand network interface is discovered, the **Ironic Inspector** adds a `client-id` attribute to the `extra` attribute in the ironic port. The **Ironic Inspector** should be configured with `iptables.ethoib_interfaces` to indicate the Ethernet Over InfiniBand (EoIB) which are used for physical access to the DHCP network. For example if **Ironic Inspector** DHCP server is using `br-inspector` and the `br-inspector` has EoIB port e.g. `eth0`, the `iptables.ethoib_interfaces` should be set to `eth0`. The `iptables.ethoib_interfaces` allows to map the baremetal GUID to its EoIB MAC based on the `neighs` files. This is needed for blocking DHCP traffic of the nodes (MACs) which are not part of the introspection.

The format of the `/sys/class/net/<ethoib>/eth/neighs` file:

```
# EMAC=<ethernet mac of the ethoib> IMAC=<qp number:lid:GUID>
# For example:
IMAC=97:fe:80:00:00:00:00:00:00:7c:fe:90:03:00:29:26:52
qp number=97:fe
lid=80:00:00:00:00:00:00:00
GUID=7c:fe:90:03:00:29:26:52
```

Example of content:



```
EMAC=02:00:02:97:00:01 IMAC=97:fe:80:00:00:00:00:00:00:7c:fe:90:03:00:29:26:52
EMAC=02:00:00:61:00:02 IMAC=61:fe:80:00:00:00:00:00:00:7c:fe:90:03:00:29:24:4f
```

### 3.4.3 HTTP API Reference

- Bare Metal Introspection API Reference.

### 3.4.4 Troubleshooting

#### Troubleshooting

#### Errors when starting introspection

- *Invalid provision state available*

In Kilo release with *python-ironicclient* 0.5.0 or newer Ironic defaults to reporting provision state `AVAILABLE` for newly enrolled nodes. **ironic-inspector** will refuse to conduct introspection in this state, as such nodes are supposed to be used by Nova for scheduling. See *node states* for instructions on how to put nodes into the correct state.

#### Introspection times out

There may be 3 reasons why introspection can time out after some time (defaulting to 60 minutes, altered by `timeout` configuration option):

1. Fatal failure in processing chain before node was found in the local cache. See *Troubleshooting data processing* for the hints.
2. Failure to load the ramdisk on the target node. See *Troubleshooting PXE boot* for the hints.
3. Failure during ramdisk run. See *Troubleshooting ramdisk run* for the hints.

#### Troubleshooting data processing

In this case **ironic-inspector** logs should give a good idea what went wrong. E.g. for RDO or Fedora the following command will output the full log:

```
sudo journalctl -u openstack-ironic-inspector
```

(use `openstack-ironic-discoverd` for version < 2.0.0).

#### Note

Service name and specific command might be different for other Linux distributions (and for old version of **ironic-inspector**).

If `ramdisk_error` plugin is enabled and `ramdisk_logs_dir` configuration option is set, **ironic-inspector** will store logs received from the ramdisk to the `ramdisk_logs_dir` directory. This depends, however, on the ramdisk implementation.

A local cache miss during data processing would leave a message like:

```
ERROR ironic_python_agent.inspector [-] inspectorerror 400: {"error":{"message
↳:"The following failures happened during running pre-processing hooks:\
↳nLook up error: Could not find a node for attributes {'bmc_address': u'10.x.
↳y.z', 'mac': [u'00:aa:bb:cc:dd:ee', u'00:aa:bb:cc:dd:ef']}}"}}
```

One potential explanation for such an error is a misconfiguration in the BMC where a channel with the wrong IP address is active (and hence detected and reported back by the Ironic Python Agent), but can then not be matched to the IP address Ironic has in its cache for this node.

## Troubleshooting PXE boot

PXE booting most often becomes a problem for bare metal environments with several physical networks. If the hardware vendor provides a remote console (e.g. iDRAC for DELL), use it to connect to the machine and see what is going on. You may need to restart introspection.

Another source of information is DHCP and TFTP server logs. Their location depends on how the servers were installed and run. For RDO or Fedora use:

```
$ sudo journalctl -u openstack-ironic-inspector-dnsmasq
```

(use `openstack-ironic-discoverd-dnsmasq` for version < 2.0.0).

The last resort is `tcpdump` utility. Use something like

```
$ sudo tcpdump -i any port 67 or port 68 or port 69
```

to watch both DHCP and TFTP traffic going through your machine. Replace any with a specific network interface to check that DHCP and TFTP requests really reach it.

If you see node not attempting PXE boot or attempting PXE boot on the wrong network, reboot the machine into BIOS settings and make sure that only one relevant NIC is allowed to PXE boot.

If you see node attempting PXE boot using the correct NIC but failing, make sure that:

1. network switches configuration does not prevent PXE boot requests from propagating,
2. there is no additional firewall rules preventing access to port 67 on the machine where *ironic-inspector* and its DHCP server are installed.

If you see node receiving DHCP address and then failing to get kernel and/or ramdisk or to boot them, make sure that:

1. TFTP server is running and accessible (use `tftp` utility to verify),
2. no firewall rules prevent access to TFTP port,
3. SELinux is configured properly to allow external TFTP access,

If SELinux is neither permissive nor disabled, you should config `tftp_home_dir` in SELinux by executing the command

```
$ sudo setsebool -P tftp_home_dir 1
```

See [the man page](#) for more details.

4. DHCP server is correctly set to point to the TFTP server,
5. `pxelinux.cfg/default` within TFTP root contains correct reference to the kernel and ramdisk.

**Note**

If using iPXE instead of PXE, check the HTTP server logs and the iPXE configuration instead.

**Troubleshooting ramdisk run**

First, check if the ramdisk logs were stored locally as described in the *Troubleshooting data processing* section. If not, ensure that the ramdisk actually booted as described in the *Troubleshooting PXE boot* section.

Finally, you can try connecting to the IPA ramdisk. If you have any remote console access to the machine, you can check the logs as they appear on the screen. Otherwise, you can rebuild the IPA image with your SSH key to be able to log into it. Use the `dynamic-login` or `devuser` element for a DIB-based build or put an `authorized_keys` file in `/usr/share/oem/` for a CoreOS-based one.

**Troubleshooting DNS issues on Ubuntu**

Ubuntu uses local DNS caching, so tries localhost for DNS results first before calling out to an external DNS server. When DNSmasq is installed and configured for use with ironic-inspector, it can cause problems by interfering with the local DNS cache. To fix this issue ensure that `/etc/resolv.conf` points to your external DNS servers and not to `127.0.0.1`.

On Ubuntu 14.04 this can be done by editing your `/etc/resolvconf/resolv.conf.d/head` and adding your nameservers there. This will ensure they will come up first when `/etc/resolv.conf` is regenerated.

**Troubleshooting DnsmasqFilter**

When introspection fails and the following error is in `ironic-inspector.log`

```
ERROR ironic_inspector.node_cache [-] [node: 651da5a3-4ecb-4214-a87d-
↳139cc7778c05
state starting] Processing the error event because of an exception
<class 'ironic_inspector.pxe_filter.base.InvalidFilterDriverState'>:
The PXE filter driver DnsmasqFilter, state=uninitialized: my fsm encountered
↳an
exception: Can not transition from state 'uninitialized' on event 'sync'
(no defined transition) raised by ironic_inspector.introspect._do_introspect:
ironic_inspector.pxe_filter.base.InvalidFilterDriverState: The PXE filter
↳driver
DnsmasqFilter, state=uninitialized: my fsm encountered an exception:
Can not transition from state 'uninitialized' on event 'sync'
(no defined transition)
```

restart `ironic-inspector`.

**Running Inspector in a VirtualBox environment**

By default VirtualBox does not expose a DMI table to the guest. This prevents ironic-inspector from being able to discover the properties of the a node. In order to run ironic-inspector on a VirtualBox guest the host must be configured to expose DMI data inside the guest. To do this run the following command on the VirtualBox host:

```
VBoxManage setextradata {NodeName} "VBoxInternal/Devices/pcbios/0/Config/
↳DmiExposeMemoryTable" 1
```

#### Note

Replace *{NodeName}* with the name of the guest you wish to expose the DMI table on. This command will need to be run once per host to enable this functionality.

## HTTP API

See <https://docs.openstack.org/api-ref/baremetal-introspection/>

## 3.5 Administrator Guide

### 3.5.1 How to upgrade Ironic Inspector

#### Upgrade Guide

The [release notes](#) should always be read carefully when upgrading the ironic-inspector service. Starting with the Mitaka series, specific upgrade steps and considerations are well-documented in the release notes.

Upgrades are only supported one series at a time, or within a series. Only offline (with downtime) upgrades are currently supported.

When upgrading ironic-inspector, the following steps should always be taken:

- Update ironic-inspector code, without restarting the service yet.
- Stop the ironic-inspector service.
- Run database migrations:

```
ironic-inspector-dbsync --config-file <PATH-TO-INSPECTOR.CONF> upgrade
```

- Start the ironic-inspector service.
- Upgrade the ironic-python-agent image used for introspection.

#### Note

There is no implicit upgrade order between ironic and ironic-inspector, unless the [release notes](#) say otherwise.

### Migrating introspection data

Starting with Stein release, ironic-inspector supports two introspection data storage backends: `swift` and `database`. If you decide to change the backend, you can use the provided command to migrate the data:

```
ironic-inspector-migrate-data --from swift --to database --config-file /etc/
↳ironic-inspector/inspector.conf
```

**Note**

Configuration for **both** backends is expected to be present in the configuration file for this command to succeed.

### 3.5.2 Dnsmasq PXE filter driver

#### dnsmasq PXE filter

An inspection PXE DHCP stack is often implemented by the **dnsmasq** service. The **dnsmasq** PXE filter implementation relies on directly configuring the **dnsmasq** DHCP service to provide a caching PXE traffic filter of node MAC addresses.

#### How it works

The filter works by populating the **dnsmasq** DHCP hosts directory with a configuration file per MAC address. Each file is either enabling or disabling, thru the `ignore` directive, the DHCP service for a particular MAC address:

```
$ cat /etc/dnsmasq.d/de-ad-be-ef-de-ad
de:ad:be:ef:de:ad,ignore
$
```

The filename is used to keep track of all MAC addresses in the cache, avoiding file parsing. The content of the file determines the MAC address access policy.

Thanks to the `inotify` facility, **dnsmasq** is notified once a new file is *created* or an existing file is *modified* in the DHCP hosts directory. Thus, to allow a MAC address, the filter removes the `ignore` directive:

```
$ cat /etc/dnsmasq.d/de-ad-be-ef-de-ad
de:ad:be:ef:de:ad
$
```

The hosts directory content establishes a *cached* MAC addresses filter that is kept synchronized with the **ironic** port list.

**Note**

The **dnsmasq** `inotify` facility implementation doesn't react to a file being removed or truncated.

#### Configuration

The `inotify` facility was [introduced](#) to **dnsmasq** in the version 2.73. This filter driver has been checked by **ironic-inspector** CI with **dnsmasq** versions  $\geq 2.76$ .

To enable the **dnsmasq** PXE filter, update the PXE filter driver name in the **ironic-inspector** configuration file:

```
[pxe_filter]
driver = dnsmasq
```

The DHCP hosts directory can be specified to override the default `/var/lib/ironic-inspector/dhcp-hostsdir`:

```
[dnsmasq_pxe_filter]
dhcp_hostsdir = /etc/ironic-inspector/dhcp-hostsdir
```

The filter design relies on the hosts directory being in exclusive **ironic-inspector** control. The hosts directory should be considered a *private cache* directory of **ironic-inspector** that **dnsmasq** polls configuration updates from, through the `inotify` facility. The directory has to be writable by **ironic-inspector** and readable by **dnsmasq**.

It is also possible to override the default (empty) **dnsmasq** start and stop commands to, for instance, directly control the **dnsmasq** service:

```
[dnsmasq_pxe_filter]
dnsmasq_start_command = dnsmasq --conf-file /etc/ironic-inspector/dnsmasq.conf
dnsmasq_stop_command = kill $(cat /var/run/dnsmasq.pid)
```

### Note

The commands support shell expansion. The default empty start command means the **dnsmasq** service won't be started upon the filter initialization. Conversely, the default empty stop command means the service won't be stopped upon an (error) exit.

### Note

These commands are executed through the `rootwrap` facility, so overriding may require a filter file to be created in the `rootwrap.d` directory. A sample configuration to use with the `systemctl` facility might be:

```
sudo cat > /etc/ironic-inspector/rootwrap.d/ironic-inspector-dnsmasq-
↪systemctl.filters <<EOF
[Filters]
# ironic_inspector/pxe_filter/dnsmasq.py
systemctl: CommandFilter, systemctl, root, restart, dnsmasq
systemctl: CommandFilter, systemctl, root, stop, dnsmasq
EOF
```

## Caveats

The initial synchronization will put some load on the **dnsmasq** service starting based on the amount of ports **ironic** keeps. The start-up can take up to a minute of full CPU load for huge amounts of MACs (tens of thousands). Subsequent filter synchronizations will only cause the **dnsmasq** to parse the modified files. Typically those are the bare metal nodes being added or phased out from the compute service, meaning dozens of file updates per sync call.

The **ironic-inspector** takes over the control of the DHCP hosts directory to implement its filter cache. Files are generated dynamically so should not be edited by hand. To minimize the interference between the deployment and introspection, **ironic-inspector** has to start the **dnsmasq** service only after the initial synchronization. Conversely, the **dnsmasq** service is stopped upon (unexpected) **ironic-inspector** exit.

To avoid accumulating stale DHCP host files over time, the driver cleans up the DHCP hosts directory before the initial synchronization during the start-up.

Although the filter driver tries its best to always stop the **dnsmasq** service, it is recommended that the operator configures the **dnsmasq** service in such a way that it terminates upon **ironic-inspector** (unexpected) exit to prevent a stale deny list from being used by the **dnsmasq** service.





## CONTRIBUTOR DOCS

### 4.1 How To Contribute

#### 4.1.1 Basics

- Our source code is hosted on [OpenStack GitHub](#), but please do not send pull requests there.
- Please follow usual OpenStack [Gerrit Workflow](#) to submit a patch.
- Update change log in README.rst on any significant change.
- It goes without saying that any code change should be accompanied by unit tests.
- Note the branch you're proposing changes to. `master` is the current focus of development, use `stable/VERSION` for proposing an urgent fix, where `VERSION` is the current stable series. E.g. at the moment of writing the stable branch is `stable/1.0`.
- Please file an RFE in [StoryBoard](#) for any significant code change and a regular story for any significant bug fix.

#### 4.1.2 Development Environment

First of all, install `tox` utility. It's likely to be in your distribution repositories under name of `python-tox`. Alternatively, you can install it from PyPI.

Next checkout and create environments:

```
git clone https://github.com/openstack/ironic-inspector.git
cd ironic-inspector
tox
```

Repeat `tox` command each time you need to run tests. If you don't have Python interpreter of one of supported versions (currently 3.6 and 3.7), use `-e` flag to select only some environments, e.g.

```
tox -e py36
```

#### Note

This command also runs tests for database migrations. By default the `sqlite` backend is used. For testing with `mysql` or `postgresql`, you need to set up a db named `openstack_citest` with user `openstack_citest` and password `openstack_citest` on localhost. Use the script `tools/test_setup.sh` to set the database up the same way as done in the OpenStack CI environment.

### Note

Users of Fedora <= 23 will need to run `sudo dnf releasever=24 update python-virtualenv` to run unit tests

To run the functional tests, use:

```
tox -e func
```

Once you have added new state or transition into inspection state machine, you should regenerate *State machine diagram* with:

```
tox -e genstates
```

Run the service with:

```
./tox/py36/bin/ironic-inspector --config-file example.conf
```

Of course you may have to modify `example.conf` to match your OpenStack environment. See the [install guide](#) for information on generating or downloading an example configuration file.

You can develop and test **ironic-inspector** using DevStack - see [Deploying Ironic Inspector with DevStack](#) for the current status.

### 4.1.3 Deploying Ironic Inspector with DevStack

DevStack provides a way to quickly build a full OpenStack development environment with requested components. There is a plugin for installing **ironic-inspector** in DevStack. Installing **ironic-inspector** requires a machine running Ubuntu 14.04 (or later) or Fedora 23 (or later). Make sure this machine is fully up to date and has the latest packages installed before beginning this process.

Download DevStack:

```
git clone https://git.openstack.org/openstack-dev/devstack.git
cd devstack
```

Create `local.conf` file with minimal settings required to enable both the **ironic** and the **ironic-inspector**. You can start with the [Example local.conf](#) and extend it as needed.

#### Example local.conf

```
[[local|localrc]]
# Credentials
ADMIN_PASSWORD=password
DATABASE_PASSWORD=password
RABBIT_PASSWORD=password
SERVICE_PASSWORD=password
SERVICE_TOKEN=password
SWIFT_HASH=password
SWIFT_TEMPURL_KEY=password

# Enable Ironic plugin
```

(continues on next page)

(continued from previous page)

```
enable_plugin ironic https://opendev.org/openstack/ironic
enable_plugin ironic-inspector https://opendev.org/openstack/ironic-inspector

# Disable nova novnc service, ironic does not support it anyway.
disable_service n-novnc

# Enable Swift for the direct deploy interface.
enable_service s-proxy
enable_service s-object
enable_service s-container
enable_service s-account

# Disable Horizon
disable_service horizon

# Disable Cinder
disable_service cinder c-sch c-api c-vol

# Swift temp URL's are required for the direct deploy interface
SWIFT_ENABLE_TEMPURLS=True

# Create 3 virtual machines to pose as Ironic's baremetal nodes.
IRONIC_VM_COUNT=3
IRONIC_BAREMETAL_BASIC_OPS=True
DEFAULT_INSTANCE_TYPE=baremetal

# Enable additional hardware types, if needed.
#IRONIC_ENABLED_HARDWARE_TYPES=ipmi,fake-hardware
# Don't forget that many hardware types require enabling of additional
# interfaces, most often power and management:
#IRONIC_ENABLED_MANAGEMENT_INTERFACES=ipmitool,fake
#IRONIC_ENABLED_POWER_INTERFACES=ipmitool,fake
# The 'ipmi' hardware type's default deploy interface is 'iscsi'.
# This would change the default to 'direct':
#IRONIC_DEFAULT_DEPLOY_INTERFACE=direct

# Enable inspection via ironic-inspector
IRONIC_ENABLED_INSPECT_INTERFACES=inspector,no-inspect
# Make it the default for all hardware types:
IRONIC_DEFAULT_INSPECT_INTERFACE=inspector

# Change this to alter the default driver for nodes created by devstack.
# This driver should be in the enabled list above.
IRONIC_DEPLOY_DRIVER=ipmi

# The parameters below represent the minimum possible values to create
# functional nodes.
IRONIC_VM_SPECS_RAM=2048
IRONIC_VM_SPECS_DISK=10
```

(continues on next page)

(continued from previous page)

```
# Size of the ephemeral partition in GB. Use 0 for no ephemeral partition.
IRONIC_VM_EPHEMERAL_DISK=0

# To build your own IPA ramdisk from source, set this to True
IRONIC_BUILD_DEPLOY_RAMDISK=False
IRONIC_INSPECTOR_BUILD_RAMDISK=False
VIRT_DRIVER=ironic

# By default, DevStack creates a 10.0.0.0/24 network for instances.
# If this overlaps with the hosts network, you may adjust with the
# following.
NETWORK_GATEWAY=10.1.0.1
FIXED_RANGE=10.1.0.0/24
FIXED_NETWORK_SIZE=256

# Log all output to files
LOGFILE=/opt/stack/devstack.log
LOGDIR=/opt/stack/logs
IRONIC_VM_LOG_DIR=/opt/stack/ironic-bm-logs
```

## Notes

- Set `IRONIC_INSPECTOR_BUILD_RAMDISK` to `True` if you want to build ramdisk. Default value is `False` and ramdisk will be downloaded instead of building.
- 1024 MiB of RAM is a minimum required for the default build of IPA based on CoreOS. If you plan to use another operating system and build IPA with diskimage-builder 2048 MiB is recommended.
- Network configuration is pretty sensitive, better not to touch it without deep understanding.
- This configuration disables **horizon**, **heat**, **cinder** and **tempest**, adjust it if you need these services.

Start the install:

```
./stack.sh
```

## Usage

After installation is complete, you can source `openrc` in your shell, and then use the OpenStack CLI to manage your DevStack:

```
source openrc admin demo
```

Show DevStack screens:

```
screen -x stack
```

To exit screen, hit `CTRL-a d`.

List baremetal nodes:

```
baremetal node list
```

Bring the node to manageable state:

```
baremetal node manage <NodeID>
```

Inspect the node:

```
baremetal node inspect <NodeID>
```

#### Note

The deploy driver used must support the inspect interface. See also the [Ironic Python Agent](#).

A node can also be inspected using the following command. However, this will not affect the provision state of the node:

```
baremetal introspection start <NodeID>
```

Check inspection status:

```
baremetal introspection status <NodeID>
```

Optionally, get the inspection data:

```
baremetal introspection data save <NodeID>
```

### 4.1.4 Writing a Plugin

- **ironic-inspector** allows you to hook code into the data processing chain after introspection. Inherit `ProcessingHook` class defined in `ironic_inspector.plugins.base module` and overwrite any or both of the following methods:

#### **before\_processing(introspection\_data, \*\*)**

called before any data processing, providing the raw data. Each plugin in the chain can modify the data, so order in which plugins are loaded matters here. Returns nothing.

#### **before\_update(introspection\_data, node\_info, \*\*)**

called after node is found and ports are created, but before data is updated on a node. Please refer to the docstring for details and examples.

You can optionally define the following attribute:

#### **dependencies**

a list of entry point names of the hooks this hook depends on. These hooks are expected to be enabled before the current hook.

Make your plugin a setuptools entry point under `ironic_inspector.hooks.processing namespace` and enable it in the configuration file (`processing.processing_hooks` option).

- **ironic-inspector** allows plugins to override the action when node is not found in node cache. Write a callable with the following signature:

### **(introspection\_data, \*\*)**

called when node is not found in cache, providing the processed data. Should return a `NodeInfo` class instance.

Make your plugin a `setuptools` entry point under `ironic_inspector.hooks.node_not_found` namespace and enable it in the configuration file (`processing.node_not_found_hook` option).

- **ironic-inspector** allows more condition types to be added for *Introspection Rules*. Inherit `RuleConditionPlugin` class defined in *ironic\_inspector.plugins.base module* and overwrite at least the following method:

### **check(node\_info, field, params, \*\*)**

called to check that condition holds for a given field. Field value is provided as `field` argument, `params` is a dictionary defined at the time of condition creation. Returns boolean value.

The following methods and attributes may also be overridden:

### **validate(params, \*\*)**

called to validate parameters provided during condition creating. Default implementation requires keys listed in `REQUIRED_PARAMS` (and only them).

### **REQUIRED\_PARAMS**

contains set of required parameters used in the default implementation of `validate` method, defaults to `value` parameter.

### **ALLOW\_NONE**

if its set to `True`, missing fields will be passed as `None` values instead of failing the condition. Defaults to `False`.

Make your plugin a `setuptools` entry point under `ironic_inspector.rules.conditions` namespace.

- **ironic-inspector** allows more action types to be added for *Introspection Rules*. Inherit `RuleActionPlugin` class defined in *ironic\_inspector.plugins.base module* and overwrite at least the following method:

### **apply(node\_info, params, \*\*)**

called to apply the action.

The following methods and attributes may also be overridden:

### **validate(params, \*\*)**

called to validate parameters provided during actions creating. Default implementation requires keys listed in `REQUIRED_PARAMS` (and only them).

### **REQUIRED\_PARAMS**

contains set of required parameters used in the default implementation of `validate` method, defaults to no parameters.

Make your plugin a `setuptools` entry point under `ironic_inspector.rules.conditions` namespace.

#### Note

\*\* argument is needed so that we can add optional arguments without breaking out-of-tree plugins. Please make sure to include and ignore it.

### 4.1.5 Making changes to the database

In order to make a change to the ironic-inspector database you must update the database models found in *ironic\_inspector.db package* and then create a migration to reflect that change.

There are two ways to create a migration which are described below, both of these generate a new migration file. In this file there is only one function:

- **upgrade - The function to run when**  
`ironic-inspector-dbsync upgrade` is run, and should be populated with code to bring the database up to its new state from the state it was in after the last migration.

For further information on creating a migration, refer to [Create a Migration Script](#) from the alembic documentation.

#### Autogenerate

This is the simplest way to create a migration. Alembic will compare the models to an up to date database, and then attempt to write a migration based on the differences. This should generate correct migrations in most cases however there are some cases when it can not detect some changes and may require manual modification, see [What does Autogenerate Detect \(and what does it not detect?\)](#) from the alembic documentation.

```
ironic-inspector-dbsync upgrade
ironic-inspector-dbsync revision -m "A short description" --autogenerate
```

#### Manual

This will generate an empty migration file, with the correct revision information already included. However the upgrade function is left empty and must be manually populated in order to perform the correct actions on the database:

```
ironic-inspector-dbsync revision -m "A short description"
```

### 4.1.6 Implementing PXE Filter Drivers

#### Background

**inspector** in-band introspection PXE-boots the Ironic Python Agent live image, to inspect the baremetal server. **ironic** also PXE-boots IPA to perform tasks on a node, such as deploying an image. **ironic** uses **neutron** to provide DHCP, however **neutron** does not provide DHCP for unknown MAC addresses so **inspector** has to use its own DHCP/TFTP stack for discovery and inspection.

When **ironic** and **inspector** are operating in the same L2 network, there is a potential for the two DHCPs to race, which could result in a node being deployed by **ironic** being PXE booted by **inspector**.

To prevent DHCP races between the **inspector** DHCP and **ironic** DHCP, **inspector** has to be able to filter which nodes can get a DHCP lease from the **inspector** DHCP server. These filters can then be used to prevent nodes enrolled in **ironic** inventory from being PXE-booted unless they are explicitly moved into the inspected state.

### Filter Interface

The contract between **inspector** and a PXE filter driver is described in the *FilterDriver* interface. The methods a driver has to implement are:

- *init\_filter()* called on the service start to initialize internal driver state
- *sync()* called both periodically and when a node starts or finishes introspection to allow or deny its ports MAC addresses in the driver
- *tear\_down\_filter()* called on service exit to reset the internal driver state

The driver-specific configuration is suggested to be parsed during instantiation. There's also a convenience generic interface implementation *BaseFilter* that provides base locking and initialization implementation. If required, a driver can opt-out from the periodic synchronization by overriding the *get\_periodic\_sync\_task()*.

### 4.1.7 Python API

#### ironic\_inspector

#### ironic\_inspector package

#### Subpackages

#### ironic\_inspector.cmd package

#### Submodules

#### ironic\_inspector.cmd.all module

The Ironic Inspector service.

```
ironic_inspector.cmd.all.main(args=['-b', 'latex', 'doc/source', 'doc/build/pdf'])
```

#### ironic\_inspector.cmd.conductor module

The Ironic Inspector Conductor service.

```
ironic_inspector.cmd.conductor.main(args=['-b', 'latex', 'doc/source', 'doc/build/pdf'])
```

#### ironic\_inspector.cmd.dbsync module

```
ironic_inspector.cmd.dbsync.add_alembic_command(subparsers, name)
```

```
ironic_inspector.cmd.dbsync.add_command_parsers(subparsers)
```

```
ironic_inspector.cmd.dbsync.do_alembic_command(config, cmd, *args, **kwargs)
```

```
ironic_inspector.cmd.dbsync.do_revision(config, cmd, *args, **kwargs)
```

```
ironic_inspector.cmd.dbsync.main(args=['-b', 'latex', 'doc/source', 'doc/build/pdf'])
```

```
ironic_inspector.cmd.dbsync.with_revision(config, cmd, *args, **kwargs)
```



### **ironic\_inspector.cmd.migration module**

Migrate introspected data between Swift and database.

**class** `ironic_inspector.cmd.migration.MigrationTool`

Bases: `object`

**main()**

`ironic_inspector.cmd.migration.main()`

### **ironic\_inspector.cmd.status module**

**class** `ironic_inspector.cmd.status.Checks`

Bases: `UpgradeCommands`

Upgrade checks for the ironic-status upgrade check command

Upgrade checks should be added as separate methods in this class and added to `_upgrade_checks` tuple.

`ironic_inspector.cmd.status.main()`

### **ironic\_inspector.cmd.wsgi module**

WSGI script for Ironic Inspector API, installed by pbr.

`ironic_inspector.cmd.wsgi.initialize_wsgi_app()`

## **Module contents**

### **ironic\_inspector.common package**

#### **Submodules**

### **ironic\_inspector.common.auth\_basic module**

**class** `ironic_inspector.common.auth_basic.BasicAuthMiddleware`(*app, auth\_file*)

Bases: `object`

Middleware which performs HTTP basic authentication on requests

**format\_exception**(*e*)

`ironic_inspector.common.auth_basic.auth_entry`(*entry, password*)

Compare a password with a single user auth file entry

**Param**

entry: Line from auth user file to use for authentication

**Param**

password: Password encoded as bytes

**Returns**

A dictionary of WSGI environment values to append to the request

**Raises**

Unauthorized, if the entry doesnt match supplied password or if the entry is crypted with a method other than bcrypt

`ironic_inspector.common.auth_basic.authenticate(auth_file, username, password)`

Finds username and password match in Apache style user auth file

The user auth file format is expected to comply with Apache documentation[1] however the bcrypt password digest is the *only* digest format supported.

[1] [https://httpd.apache.org/docs/current/misc/password\\_encryptions.html](https://httpd.apache.org/docs/current/misc/password_encryptions.html)

**Param**

auth\_file: Path to user auth file

**Param**

username: Username to authenticate

**Param**

password: Password encoded as bytes

**Returns**

A dictionary of WSGI environment values to append to the request

**Raises**

Unauthorized, if no file entries match supplied username/password

`ironic_inspector.common.auth_basic.parse_entry(entry)`

Extrace the username and crypted password from a user auth file entry

**Param**

entry: Line from auth user file to use for authentication

**Returns**

a tuple of username and crypted password

**Raises**

ConfigInvalid if the password is not in the supported bcrypt format

`ironic_inspector.common.auth_basic.parse_header(env)`

Parse WSGI environment for Authorization header of type Basic

**Param**

env: WSGI environment to get header from

**Returns**

Token portion of the header value

**Raises**

Unauthorized, if header is missing or if the type is not Basic

`ironic_inspector.common.auth_basic.parse_token(token)`

Parse the token portion of the Authentication header value

**Param**

token: Token value from basic authorization header

**Returns**

tuple of username, password

**Raises**

Unauthorized, if username and password could not be parsed for any reason

`ironic_inspector.common.auth_basic.unauthorized(message=None)`

Raise an Unauthorized exception to prompt for basic authentication

**Param**

message: Optional message for exception

**Raises**

Unauthorized with WWW-Authenticate header set

`ironic_inspector.common.auth_basic.validate_auth_file(auth_file)`

Read the auth user file and validate its correctness

**Param**

auth\_file: Path to user auth file

**Raises**

ConfigInvalid on validation error

**ironic\_inspector.common.context module**

**class** `ironic_inspector.common.context.RequestContext(is_public_api=False, **kwargs)`

Bases: `RequestContext`

Extends security contexts from the oslo.context library.

**classmethod** `from_dict(values, **kwargs)`

Construct a context object from a provided dictionary.

**classmethod** `from_environ(environ, **kwargs)`

Load a context object from a request environment.

If keyword arguments are provided then they override the values in the request environment.

**Parameters**

**environ** (*dict*) The environment dictionary associated with a request.

**to\_policy\_values()**

A dictionary of context attributes to enforce policy with.

oslo.policy enforcement requires a dictionary of attributes representing the current logged in user on which it applies policy enforcement. This dictionary defines a standard list of attributes that should be available for enforcement across services.

It is expected that services will often have to override this method with either deprecated values or additional attributes used by that service specific policy.

**ironic\_inspector.common.coordination module**

**class** `ironic_inspector.common.coordination.Coordinator(prefix=None)`

Bases: `object`

Tooz coordination wrapper.

**get\_lock(uuid)**

Get lock for node uuid.

**get\_members()**

Get members in the service group.

**group\_name = b'ironic\_inspector.service\_group'**

**join\_group()**

Join service group.

**leave\_group()**

Leave service group

**lock\_prefix = 'ironic\_inspector.'**

**run\_elect\_coordinator()**

Trigger a new leader election.

**start(heartbeat=True)**

Start coordinator.

#### Parameters

**heartbeat** Whether spawns a new thread to keep heartbeating with the tooz backend. Unless there is periodic task to do heartbeat manually, it should be always set to True.

**stop()**

Disconnect from coordination backend and stop heartbeat.

`ironic_inspector.common.coordination.get_coordinator(prefix=None)`

### **ironic\_inspector.common.device\_hints module**

`ironic_inspector.common.device_hints.find_devices_by_hints(devices, root_device_hints)`

Find all devices that match the root device hints.

Try to find devices that match the root device hints. In order for a device to be matched it needs to satisfy all the given hints.

#### Parameters

- **devices**

**A list of dictionaries representing the devices** containing one or more of the following keys:

**name**

(String) The device name, e.g /dev/sda

**size**

(Integer) Size of the device in *bytes*

**model**

(String) Device model

**vendor**

(String) Device vendor name

**serial**

(String) Device serial number

**wwn**

(String) Unique storage identifier

**wwn\_with\_extension**

(String): Unique storage identifier with the vendor extension appended

**wwn\_vendor\_extension**

(String): United vendor storage identifier

**rotational**

(Boolean) Whether its a rotational device or not. Useful to distinguish HDDs (rotational) and SSDs (not rotational).

**hctl**

(String): The SCSI address: Host, channel, target and lun. For example: 1:0:0:0.

**by\_path**

(String): The alternative device name, e.g. /dev/disk/by-path/pci-0000:00

- **root\_device\_hints** A dictionary with the root device hints.

**Raises**

ValueError, if some information is invalid.

**Returns**

A generator with all matching devices as dictionaries.

`ironic_inspector.common.device_hints.match_root_device_hints(devices, root_device_hints)`

Try to find a device that matches the root device hints.

Try to find a device that matches the root device hints. In order for a device to be matched it needs to satisfy all the given hints.

**Parameters**

- **devices**

**A list of dictionaries representing the devices** containing one or more of the following keys:

**name**

(String) The device name, e.g /dev/sda

**size**

(Integer) Size of the device in *bytes*

**model**

(String) Device model

**vendor**

(String) Device vendor name

**serial**

(String) Device serial number

**wwn**

(String) Unique storage identifier

**wwn\_with\_extension**

(String): Unique storage identifier with the vendor extension appended

**wwn\_vendor\_extension**

(String): United vendor storage identifier

**rotational**

(Boolean) Whether its a rotational device or not. Useful to distinguish HDDs (rotational) and SSDs (not rotational).

**hctl**

(String): The SCSI address: Host, channel, target and lun. For example: 1:0:0:0.

**by\_path**

(String): The alternative device name, e.g. /dev/disk/by-path/pci-0000:00

- **root\_device\_hints** A dictionary with the root device hints.

**Raises**

ValueError, if some information is invalid.

**Returns**

The first device to match all the hints or None.

`ironic_inspector.common.device_hints.parse_root_device_hints(root_device)`

Parse the `root_device` property of a node.

Parses and validates the `root_device` property of a node. These are hints for how a nodes root device is created. The size hint should be a positive integer. The rotational hint should be a Boolean value.

**Parameters**

**root\_device** the `root_device` dictionary from the nodes property.

**Returns**

a dictionary with the root device hints parsed or None if there are no hints.

**Raises**

ValueError, if some information is invalid.

## **ironic\_inspector.common.exception module**

IroniC base exception handling.

Includes decorator for re-raising IroniC-type exceptions.

SHOULD include dedicated exception logging.

**exception** `ironic_inspector.common.exception.BadRequest(message=None, **kwargs)`

Bases: *IroniCException*

**code = 400**

**exception** `ironic_inspector.common.exception.ConfigInvalid`(*message=None*,  
*\*\*kwargs*)

Bases: *IronicException*

**exception** `ironic_inspector.common.exception.IronicException`(*message=None*,  
*\*\*kwargs*)

Bases: `Exception`

Base Ironic Exception

To correctly use this class, inherit from it and define a `_msg_fmt` property. That `_msg_fmt` will get `printfd` with the keyword arguments provided to the constructor.

If you need to access the message from an exception you should use `str(exc)`

**code = 500**

**headers = {}**

**safe = False**

**exception** `ironic_inspector.common.exception.ServiceLookupFailure`(*message=None*,  
*\*\*kwargs*)

Bases: *IronicException*

**exception** `ironic_inspector.common.exception.ServiceRegistrationFailure`(*message=None*,  
*\*\*kwargs*)

Bases: *IronicException*

**exception** `ironic_inspector.common.exception.Unauthorized`(*message=None*, *\*\*kwargs*)

Bases: *IronicException*

**code = 401**

**headers = {'WWW-Authenticate': 'Basic realm="Baremetal API"}**

## **ironic\_inspector.common.ironic module**

**exception** `ironic_inspector.common.ironic.NotFound`(*node\_ident*, *code=404*, *\*args*,  
*\*\*kwargs*)

Bases: *Error*

Node not found in Ironic.

`ironic_inspector.common.ironic.call_with_retries`(*func*, *\*args*, *\*\*kwargs*)

Call an ironic client function retrying all errors.

If an ironic client exception is raised, try calling the `func` again, at most 5 times, waiting 1 sec between each call. If on the 5th attempt the `func` raises again, the exception is propagated to the caller.

`ironic_inspector.common.ironic.capabilities_to_dict`(*caps*)

Convert the Nodes capabilities into a dictionary.

`ironic_inspector.common.ironic.check_provision_state(node)`

Sanity checks the provision state of the node.

**Parameters**

**node** An API client returned node object describing the baremetal node according to ironics node data model.

**Returns**

None if no action is to be taken, True if the power node state should not be modified.

**Raises**

Error on an invalid state being detected.

`ironic_inspector.common.ironic.dict_to_capabilities(caps_dict)`

Convert a dictionary into a string with the capabilities syntax.

`ironic_inspector.common.ironic.get_client(token=None)`

Get an ironic client connection.

`ironic_inspector.common.ironic.get_ipmi_address(node)`

Get the BMC address defined in node.driver\_info dictionary

Possible names of BMC address value examined in order of list [ipmi\_address] + CONF.ipmi\_address\_fields. The value could be an IP address or a hostname. DNS lookup performed for the first non empty value.

The first valid BMC address value returned along with its v4 and v6 IP addresses.

**Parameters**

**node** Node object with defined driver\_info dictionary

**Returns**

tuple (ipmi\_address, ipv4\_address, ipv6\_address)

`ironic_inspector.common.ironic.get_node(node_id, ironic=None, **kwargs)`

Get a node from Ironic.

**Parameters**

- **node\_id** node UUID or name.
- **ironic** ironic client instance.
- **kwargs** arguments to pass to Ironic client.

**Raises**

Error on failure

`ironic_inspector.common.ironic.lookup_node(macs=None, bmc_addresses=None, introspection_data=None, ironic=None)`

Lookup a node in the ironic database.

`ironic_inspector.common.ironic.lookup_node_by_bmc_addresses(addresses, introspection_data=None, ironic=None, fail=False)`

Find a node by its BMC address.



`ironic_inspector.common.ironic.lookup_node_by_macs`(*macs*, *introspection\_data=None*,  
*ironic=None*, *fail=False*)

Find a node by its MACs.

`ironic_inspector.common.ironic.reset_ironic_session`()

Reset the global session variable.

Mostly useful for unit tests.

### **ironic\_inspector.common.keystone module**

`ironic_inspector.common.keystone.add_auth_options`(*options*, *service\_type*)

`ironic_inspector.common.keystone.get_adapter`(*group*, *\*\*adapter\_kwargs*)

`ironic_inspector.common.keystone.get_endpoint`(*group*, *\*\*kwargs*)

`ironic_inspector.common.keystone.get_session`(*group*)

`ironic_inspector.common.keystone.register_auth_opts`(*group*, *service\_type*)

### **ironic\_inspector.common.lldp\_parsers module**

Names and mapping functions used to map LLDP TLVs to name/value pairs

**class** `ironic_inspector.common.lldp_parsers.LLDPBasicMgmtParser`(*nv=None*)

Bases: `LLDPParser`

Class to handle parsing of 802.1AB Basic Management set

This class will also handle 802.1Q and 802.3 OUI TLVs.

**add\_capabilities**(*struct*, *name*, *data*)

Handle LLDP\_TLV\_SYS\_CAPABILITIES

**add\_mgmt\_address**(*struct*, *name*, *data*)

Handle LLDP\_TLV\_MGMT\_ADDRESS

There can be multiple Mgmt Address TLVs, store in list.

**handle\_org\_specific\_tlv**(*struct*, *name*, *data*)

Handle Organizationally Unique ID TLVs

This class supports 802.1Q and 802.3 OUI TLVs.

See <http://www.ieee802.org/1/pages/802.1Q-2014.html>, Annex D and <http://standards.ieee.org/about/get/802/802.3.html>

**class** `ironic_inspector.common.lldp_parsers.LLDPParser`(*node\_info*, *nv=None*)

Bases: `object`

Base class to handle parsing of LLDP TLVs

Each class that inherits from this base class must provide a parser map. Parser maps are used to associate a LLDP TLV with a function handler and arguments necessary to parse the TLV and generate one or more name/value pairs. Each LLDP TLV maps to a tuple with the following fields:

function - handler function to generate name/value pairs

`construct` - name of construct definition for TLV

`name` - user-friendly name of TLV. For TLVs that generate only one name/value pair this is the name used

`len_check` - boolean indicating if length check should be done on construct

Its valid to have a function handler of None, this is for TLVs that are not mapped to a name/value pair(e.g.LLDP\_TLV\_TTL).

**add\_dot1\_link\_aggregation**(*struct, name, data*)

Add name/value pairs for TLV Dot1\_LinkAggregationId

This is in base class since it can be used by both dot1 and dot3.

**add\_nested\_value**(*struct, name, data*)

Add a single nested name/value pair to the dict

**add\_single\_value**(*struct, name, data*)

Add a single name/value pair to the nv dict

**append\_value**(*name, value*)

Add value to a list mapped to name

**parse\_tlv**(*tlv\_type, data*)

Parse TLVs from mapping table

This functions takes the TLV type and the raw data for this TLV and gets a tuple from the `parser_map`. The `construct` field in the tuple contains the construct lib definition of the TLV which can be parsed to access individual fields. Once the TLV is parsed, the handler function for each TLV will store the individual fields as name/value pairs in `nv_dict`.

If the handler function does not exist, then no name/value pairs will be added to `nv_dict`, but since the TLV was handled, `True` will be returned.

**Param**

`tlv_type` - type identifier for TLV

**Param**

`data` - raw TLV value

**Returns**

True if TLV in `parser_map` and data is valid, otherwise False.

**set\_value**(*name, value*)

Set name value pair in dictionary

The value for a name should not be changed if it exists.

**class** `ironic_inspector.common.lldp_parsers.LLDPdot1Parser`(*node\_info, nv=None*)

Bases: `LLDPParser`

Class to handle parsing of 802.1Q TLVs

**add\_dot1\_port\_protocol\_vlan**(*struct, name, data*)

Handle dot1\_PORT\_PROTOCOL\_VLANID

**add\_dot1\_protocol\_identities**(*struct, name, data*)

Handle dot1\_PROTOCOL\_IDENTITY

There can be multiple protocol ids TLVs, store in list

**add\_dot1\_vlans**(*struct, name, data*)

Handle dot1\_VLAN\_NAME

There can be multiple vlan TLVs, add dictionary entry with id/vlan to list.

**class** `ironic_inspector.common.lldp_parsers.LLDPdot3Parser`(*node\_info, nv=None*)

Bases: `LLDPParser`

Class to handle parsing of 802.3 TLVs

**add\_dot3\_macphy\_config**(*struct, name, data*)

Handle dot3\_MACPHY\_CONFIG\_STATUS

### **ironic\_inspector.common.lldp\_tlvs module**

Link Layer Discovery Protocol TLVs

`ironic_inspector.common.lldp_tlvs.bytes_to_int`(*obj*)

Convert bytes to an integer

**Param**

*obj* - array of bytes

`ironic_inspector.common.lldp_tlvs.get_autoneg_cap`(*pmd*)

Get autonegotiated capability strings

This returns a list of capability strings from the Physical Media Dependent (PMD) capability bits.

**Parameters**

**pmd** PMD bits

**Returns**

Sorted list containing capability strings

`ironic_inspector.common.lldp_tlvs.mapping_for_enum`(*mapping*)

Return tuple used for keys as a dict

**Param**

*mapping* - dict with tuple as keys

`ironic_inspector.common.lldp_tlvs.mapping_for_switch`(*mapping*)

Return dict from values

**Param**

*mapping* - dict with tuple as keys

### **ironic\_inspector.common.locking module**

**class** `ironic_inspector.common.locking.BaseLock`

Bases: `object`

**abstract acquire**(*blocking=True*)

Acquire lock.

**abstract is\_locked**()

Return lock status

**abstract release**()

Release lock.

**class** `ironic_inspector.common.locking.InternalLock`(*uuid*)

Bases: `BaseLock`

Locking mechanism based on `threading.Semaphore`.

**acquire**(*blocking=True*)

Acquire lock.

**is\_locked**()

Return lock status

**release**()

Release lock.

**class** `ironic_inspector.common.locking.ToozLock`(*lock*)

Bases: `BaseLock`

Wrapper on tooz locks.

**acquire**(*blocking=True*)

Acquire lock.

**is\_locked**()

Return lock status

**release**()

Release lock.

`ironic_inspector.common.locking.get_lock`(*uuid*)

## **ironic\_inspector.common.mdns module**

Multicast DNS implementation for API discovery.

This implementation follows RFC 6763 as clarified by the API SIG guideline <https://review.opendev.org/651222>.

**class** `ironic_inspector.common.mdns.Zeroconf`

Bases: `object`

Multicast DNS implementation client and server.

Uses threading internally, so there is no start method. It starts automatically on creation.

### **Warning**

The underlying library does not yet support IPv6.

**close()**

Shut down mDNS and unregister services.

**Note**

If another server is running for the same services, it will re-register them immediately.

**get\_endpoint**(*service\_type*, *skip\_loopback=True*, *skip\_link\_local=False*)

Get an endpoint and its properties from mDNS.

If the requested endpoint is already in the built-in server cache, and its TTL is not exceeded, the cached value is returned.

**Parameters**

- **service\_type** OpenStack service type.
- **skip\_loopback** Whether to ignore loopback addresses.
- **skip\_link\_local** Whether to ignore link local V6 addresses.

**Returns**

tuple (endpoint URL, properties as a dict).

**Raises**

[\*ServiceLookupFailure\*](#) if the service cannot be found.

**register\_service**(*service\_type*, *endpoint*, *params=None*)

Register a service.

This call announces the new services via multicast and instructs the built-in server to respond to queries about it.

**Parameters**

- **service\_type** OpenStack service type, e.g. baremetal.
- **endpoint** full endpoint to reach the service.
- **params** optional properties as a dictionary.

**Raises**

[\*ServiceRegistrationFailure\*](#) if the service cannot be registered, e.g. because of conflicts.

**ironic\_inspector.common.rpc module****ironic\_inspector.common.rpc.get\_client**(*topic=None*)

Get a RPC client instance.

**Parameters**

**topic** The topic of the message will be delivered to. This argument is ignored if CONF.standalone is True.

**ironic\_inspector.common.rpc.get\_server**(*endpoints*)

Get a RPC server instance.

**ironic\_inspector.common.rpc.init**()

### `ironic_inspector.common.rpc_service` module

`class ironic_inspector.common.rpc_service.RPCService(host)`

Bases: `Service`

`start()`

Start a service.

`stop()`

Stop a service.

#### Parameters

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

### `ironic_inspector.common.service_utils` module

`ironic_inspector.common.service_utils.prepare_service(args=None)`

### `ironic_inspector.common.swift` module

`class ironic_inspector.common.swift.SwiftAPI`

Bases: `object`

API for communicating with Swift.

`create_object(object, data, container=None, headers=None)`

Uploads a given string to Swift.

#### Parameters

- **object** The name of the object in Swift
- **data** string data to put in the object
- **container** The name of the container for the object. Defaults to the value set in the configuration options.
- **headers** the headers for the object to pass to Swift

#### Returns

The Swift UUID of the object

#### Raises

`utils.Error`, if any operation with Swift fails.

`get_object(object, container=None)`

Downloads a given object from Swift.

#### Parameters

- **object** The name of the object in Swift
- **container** The name of the container for the object. Defaults to the value set in the configuration options.

#### Returns

Swift object

### Raises

utils.Error, if the Swift operation fails.

`ironic_inspector.common.swift.get_introspection_data(uuid, suffix=None)`

Downloads introspection data from Swift.

### Parameters

- **uuid** UUID of the Ironic node that the data came from
- **suffix** optional suffix to add to the underlying swift object name

### Returns

Swift object with the introspection data

`ironic_inspector.common.swift.reset_swift_session()`

Reset the global session variable.

Mostly useful for unit tests.

`ironic_inspector.common.swift.store_introspection_data(data, uuid, suffix=None)`

Uploads introspection data to Swift.

### Parameters

- **data** data to store in Swift
- **uuid** UUID of the Ironic node that the data came from
- **suffix** optional suffix to add to the underlying swift object name

### Returns

name of the Swift object that the data is stored in

## Module contents

**ironic\_inspector.conductor package**

### Submodules

**ironic\_inspector.conductor.manager module**

**class** `ironic_inspector.conductor.manager.ConductorManager`

Bases: object

ironic inspector conductor manager

**RPC\_API\_VERSION** = '1.3'

**del\_host()**

Shutdown the ironic inspector conductor service.

**do\_abort(\*\*kwargs)**

**do\_continue(\*\*kwargs)**

**do\_introspection(\*\*kwargs)**

**do\_reapply(\*\*kwargs)**

**init\_host()**

Initialize Worker host

Init db connection, load and validate processing hooks, runs periodic tasks.

:returns None

**target = <Target version=1.3>**

`ironic_inspector.conductor.manager.periodic_clean_up()`

`ironic_inspector.conductor.manager.periodic_leader_election(conductor)`

`ironic_inspector.conductor.manager.sync_with_ironic(conductor)`

## Module contents

### ironic\_inspector.conf package

#### Submodules

#### ironic\_inspector.conf.accelerators module

`ironic_inspector.conf.accelerators.list_opts()`

`ironic_inspector.conf.accelerators.register_opts(conf)`

#### ironic\_inspector.conf.capabilities module

`ironic_inspector.conf.capabilities.list_opts()`

`ironic_inspector.conf.capabilities.register_opts(conf)`

#### ironic\_inspector.conf.coordination module

`ironic_inspector.conf.coordination.list_opts()`

`ironic_inspector.conf.coordination.register_opts(conf)`

#### ironic\_inspector.conf.default module

**class** `ironic_inspector.conf.default.Octal`(*min=None, max=None, type\_name='integer value', choices=None*)

Bases: Integer

`ironic_inspector.conf.default.list_opts()`

`ironic_inspector.conf.default.register_opts(conf)`

#### ironic\_inspector.conf.discovery module

`ironic_inspector.conf.discovery.list_opts()`

`ironic_inspector.conf.discovery.register_opts(conf)`



### **ironic\_inspector.conf.dnsmasq\_pxe\_filter module**

`ironic_inspector.conf.dnsmasq_pxe_filter.list_opts()`

`ironic_inspector.conf.dnsmasq_pxe_filter.register_opts(conf)`

### **ironic\_inspector.conf.exception module**

Ironic base exception handling.

Includes decorator for re-raising Ironic-type exceptions.

SHOULD include dedicated exception logging.

`ironic_inspector.conf.exception.register_opts(conf)`

### **ironic\_inspector.conf.extra\_hardware module**

`ironic_inspector.conf.extra_hardware.list_opts()`

`ironic_inspector.conf.extra_hardware.register_opts(conf)`

### **ironic\_inspector.conf.healthcheck module**

`ironic_inspector.conf.healthcheck.list_opts()`

`ironic_inspector.conf.healthcheck.register_opts(conf)`

### **ironic\_inspector.conf.iptables module**

`ironic_inspector.conf.iptables.list_opts()`

`ironic_inspector.conf.iptables.register_opts(conf)`

### **ironic\_inspector.conf.ironic module**

`ironic_inspector.conf.ironic.list_opts()`

`ironic_inspector.conf.ironic.register_opts(conf)`

### **ironic\_inspector.conf.mdns module**

`ironic_inspector.conf.mdns.register_opts(conf)`

### **ironic\_inspector.conf.opts module**

`ironic_inspector.conf.opts.list_opts()`

`ironic_inspector.conf.opts.parse_args(args, default_config_files=None)`

`ironic_inspector.conf.opts.set_config_defaults()`

Return a list of oslo.config options available in Inspector code.

`ironic_inspector.conf.opts.set_cors_middleware_defaults()`

Update default configuration options for oslo.middleware.

### **ironic\_inspector.conf.pci\_devices module**

`ironic_inspector.conf.pci_devices.list_opts()`

`ironic_inspector.conf.pci_devices.register_opts(conf)`

### **ironic\_inspector.conf.port\_physnet module**

`ironic_inspector.conf.port_physnet.list_opts()`

`ironic_inspector.conf.port_physnet.register_opts(conf)`

### **ironic\_inspector.conf.processing module**

`ironic_inspector.conf.processing.list_opts()`

`ironic_inspector.conf.processing.register_opts(conf)`

### **ironic\_inspector.conf.pxe\_filter module**

`ironic_inspector.conf.pxe_filter.list_opts()`

`ironic_inspector.conf.pxe_filter.register_opts(conf)`

### **ironic\_inspector.conf.service\_catalog module**

`ironic_inspector.conf.service_catalog.list_opts()`

`ironic_inspector.conf.service_catalog.register_opts(conf)`

### **ironic\_inspector.conf.swift module**

`ironic_inspector.conf.swift.list_opts()`

`ironic_inspector.conf.swift.register_opts(conf)`

## **Module contents**

### **ironic\_inspector.db package**

#### **Submodules**

#### **ironic\_inspector.db.api module**

DB models API for inspection data and shared database code.

`ironic_inspector.db.api.add_node(uuid, state, started_at=None, finished_at=None, error=None, manage_boot=None)`

Add new node

Before creating new node with certain uuid clean ups all existing node info.

#### **Parameters**

- **uuid** node uuid

- **state** initial node state
- **started\_at** node caching datetime
- **finished\_at** introspection finished datetime
- **error** introspection error
- **manage\_boot** whether to manage boot for this node

**Returns**

created node object

```
ironic_inspector.db.api.create_node(uuid, state, started_at=None, finished_at=None,
                                     error=None, manage_boot=None)
```

Create new node

**Parameters**

- **uuid** node uuid
- **state** initial node state
- **started\_at** node caching datetime
- **finished\_at** introspection finished datetime
- **error** introspection error

**Returns**

created node object

```
ironic_inspector.db.api.create_rule(uuid, conditions, actions, description=None,
                                     scope=None)
```

Create new rule

**Parameters**

- **uuid** rule uuid
- **conditions** list of (field, op, multiple, invert, params) tuple, which represents condition object
- **actions** list of (action, params) pair, which represents action object
- **description** rule description
- **scope** rule scope

**Returns**

created rule

```
ironic_inspector.db.api.delete_all_rules()
```

Delete all rules

**Returns**

None

```
ironic_inspector.db.api.delete_attributes(uuid)
```

Delete all attributes

**Parameters**

**uuid** the UUID of the node whose attributes you wish to delete

**Returns**

None

`ironic_inspector.db.api.delete_node(uuid)`

Delete node and its attributes

**Parameters**

**uuid** node uuid

**Returns**

None

`ironic_inspector.db.api.delete_nodes(finished_until=None)`

Delete all nodes

**Parameters**

**finished\_until** datetime object, delete nodes are introspected before finished\_until time

**Returns**

None

`ironic_inspector.db.api.delete_options(**filters)`

Delete all options

**Parameters**

**filters** deletion filter criteria

**Returns**

None

`ironic_inspector.db.api.delete_rule(uuid)`

Delete the rule by uuid

**Parameters**

**uuid** rule uuid

**Raises**

RuleNotFoundError in case rule not found

**Returns**

None

`ironic_inspector.db.api.get_active_nodes(started_before=None)`

Get list of nodes on introspection

**Parameters**

**started\_before** datetime object, returns nodes, started before provided time

**Returns**

list of nodes, could be empty

`ironic_inspector.db.api.get_attributes(order_by=None, **fields)`

Get all attributes

**Parameters**

- **order\_by** ordering criterion
- **fields** filter criteria fields

**Returns**

list of attributes

`ironic_inspector.db.api.get_introspection_data(node_id, processed=True)`

Get introspection data for this node.

**Parameters**

- **node\_id** node UUID.
- **processed** Specify the type of introspected data, set to False indicates retrieving the unprocessed data.

**Returns**

A dictionary representation of introspected data

`ironic_inspector.db.api.get_node(uuid, **fields)`

Get all cached nodes

**Parameters**

- **uuid** node uuid
- **fields** fields are used as filtering criterion

**Returns**

get node object

**Raises**

NodeNotFoundInDBError in case node not found or node version differ from passed in fields.

`ironic_inspector.db.api.get_nodes()`

Get list of cached nodes

**Returns**

list of nodes, could be empty

`ironic_inspector.db.api.get_options(**fields)`

Get all options

**Parameters**

**fields** filter criteria fields

**Returns**

list of options

`ironic_inspector.db.api.get_rule(uuid)`

Get rule by uuid

**Parameters**

**uuid** rule uuid

**Returns**

rule object

`ironic_inspector.db.api.get_rules(**fields)`

List all rules.

`ironic_inspector.db.api.get_rules_actions(**fields)`

Get all rule actions

**Parameters**

**fields** field filter criteria

**Returns**

list of actions

`ironic_inspector.db.api.get_rules_conditions(**fields)`

Get all rule conditions

**Parameters**

**fields** field filter criteria

**Returns**

list of conditions

`ironic_inspector.db.api.get_writer_session()`

Help method to get writer session.

**Returns**

The writer session.

`ironic_inspector.db.api.init()`

Initialize the database.

Method called on service start up, initialize transaction context manager and try to create db session.

`ironic_inspector.db.api.list_nodes_by_attributes(attributes)`

Get list of nodes with certain attributes

**Parameters**

**attributes** list of attributes as (name, value) pair

**Returns**

list of nodes, could be empty

`ironic_inspector.db.api.list_nodes_options_by_uuid(uuid)`

Get list of node options

**Parameters**

**uuid** node uuid

**Returns**

list of node options, could be empty

`ironic_inspector.db.api.model_query(model, *args, **kwargs)`

Query helper for simpler session usage.

**Parameters**

**session** if present, the session to use

`ironic_inspector.db.api.session_for_read()`

Create read session within context manager

`ironic_inspector.db.api.session_for_write()`

Create write session within context manager

`ironic_inspector.db.api.set_attribute(node_uuid, name, values)`

Set lookup attributes for node

**Parameters**

- **node\_uuid** node uuid
- **name** option name
- **values** list of attribute values

**Returns**

None

`ironic_inspector.db.api.set_option(node_uuid, name, value)`

Set option for node

**Parameters**

- **node\_uuid** node uuid
- **name** option name
- **value** option value

**Returns**

None

`ironic_inspector.db.api.store_introspection_data(node_id, introspection_data,  
processed=True)`

Store introspection data for this node.

**Parameters**

- **node\_id** node UUID.
- **introspection\_data** A dictionary of introspection data
- **processed** Specify the type of introspected data, set to False indicates the data is unprocessed.

`ironic_inspector.db.api.update_node(uuid, **values)`

Update node by uuid

Updates node fields with provided values, also bump node version.

**Parameters**

- **uuid** node uuid
- **values** node fields with values to be updated

**Raises**

`NodeNotFoundInDBError` in case node not found or node version differ from passed in values.

## ironic\_inspector.db.migration module

`ironic_inspector.db.migration.create_schema`(*config=None, engine=None*)

Create database schema from models description.

Can be used for initial installation instead of upgrade(head).

`ironic_inspector.db.migration.downgrade`(*revision, config=None*)

Used for downgrading database.

### Parameters

**version** (*string*) Desired database version

`ironic_inspector.db.migration.revision`(*message=None, autogenerate=False, config=None*)

Creates template for migration.

### Parameters

- **message** (*string*) Text that will be used for migration title
- **autogenerate** (*bool*) If True - generates diff based on current database state

`ironic_inspector.db.migration.stamp`(*revision, config=None*)

Stamps database with provided revision.

Dont run any migrations.

### Parameters

**revision** (*string*) Should match one from repository or head - to stamp database with most recent revision

`ironic_inspector.db.migration.upgrade`(*revision, config=None*)

Used for upgrading database.

### Parameters

**version** (*string*) Desired database version

`ironic_inspector.db.migration.version`(*config=None, engine=None*)

Current database version.

### Returns

Database version

### Return type

string

## ironic\_inspector.db.model module

SQLAlchemy models for inspection data and shared database code.

**class** `ironic_inspector.db.model.Attribute`(*\*\*kwargs*)

Bases: Base

**name**

**node\_uuid**

**uuid**



**value**

**class** ironic\_inspector.db.model.**IntrospectionData**(\*\*kwargs)

Bases: Base

**data**

**processed**

**uuid**

**class** ironic\_inspector.db.model.**ModelBase**

Bases: ModelBase

**class** ironic\_inspector.db.model.**Node**(\*\*kwargs)

Bases: Base

**error**

**finished\_at**

**manage\_boot**

**started\_at**

**state**

**uuid**

**version\_id**

**class** ironic\_inspector.db.model.**Option**(\*\*kwargs)

Bases: Base

**name**

**uuid**

**value**

**class** ironic\_inspector.db.model.**Rule**(\*\*kwargs)

Bases: Base

**actions**

**conditions**

**created\_at**

**description**

**disabled**

**scope**

**uuid**

```
class ironic_inspector.db.model.RuleAction(**kwargs)
```

Bases: Base

**action**

**as\_dict()**

**id**

**params**

**rule**

```
class ironic_inspector.db.model.RuleCondition(**kwargs)
```

Bases: Base

**as\_dict()**

**field**

**id**

**invert**

**multiple**

**op**

**params**

**rule**

## Module contents

### ironic\_inspector.plugins package

#### Submodules

#### ironic\_inspector.plugins.accel\_device module

Gather and distinguish Accelerator PCI devices from inventory.

```
class ironic_inspector.plugins.accel_device.AccelDevicesHook
```

Bases: *ProcessingHook*

Processing hook for distinguishing accelerator devices.

```
before_update(introspection_data, node_info, **kwargs)
```

Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

#### Parameters

- **introspection\_data** processed data from the ramdisk.
- **node\_info** NodeInfo instance.

- **kwargs** used for extensibility without breaking existing hooks.

**Returns**

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

**ironic\_inspector.plugins.base module**

Base code for plugins support.

**class** `ironic_inspector.plugins.base.ProcessingHook`

Bases: `object`

Abstract base class for introspection data processing hooks.

**before\_processing**(*introspection\_data*, **\*\*kwargs**)

Hook to run before any other data processing.

This hook is run even before sanity checks.

**Parameters**

- **introspection\_data** raw information sent by the ramdisk, may be modified by the hook.
- **kwargs** used for extensibility without breaking existing hooks

**Returns**

nothing.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)

Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

**Parameters**

- **introspection\_data** processed data from the ramdisk.
- **node\_info** `NodeInfo` instance.
- **kwargs** used for extensibility without breaking existing hooks.

**Returns**

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

**dependencies** = []

An ordered list of hooks that must be enabled before this one.

The items here should be entry point names, not classes.

**class** `ironic_inspector.plugins.base.RuleActionPlugin`

Bases: `WithValidation`

Abstract base class for rule action plugins.

**FORMATTED\_PARAMS = []**

List of params will be formatted with python format.

**abstract apply**(*node\_info*, *params*, *\*\*kwargs*)

Run action on successful rule match.

**Parameters**

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

utils.Error on failure

**class** `ironic_inspector.plugins.base.RuleConditionPlugin`

Bases: *WithValidation*

Abstract base class for rule condition plugins.

**ALLOW\_NONE = False**

Whether this condition accepts None when field is not found.

**REQUIRED\_PARAMS = {'value'}**

Set with names of required parameters.

**abstract check**(*node\_info*, *field*, *params*, *\*\*kwargs*)

Check if condition holds for a given field.

**Parameters**

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

**ValueError** on unacceptable field value

**Returns**

True if check succeeded, otherwise False

**class** `ironic_inspector.plugins.base.WithValidation`

Bases: object

**OPTIONAL\_PARAMS = {}**

Set with names of optional parameters.

**REQUIRED\_PARAMS = {}**

Set with names of required parameters.

**validate**(*params*, *\*\*kwargs*)

Validate params passed during creation.

Default implementation checks for presence of fields from REQUIRED\_PARAMS and fails for unexpected fields (not from REQUIRED\_PARAMS + OPTIONAL\_PARAMS).

#### Parameters

- **params** params as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

ValueError on validation failure

`ironic_inspector.plugins.base.introspection_data_manager()`

`ironic_inspector.plugins.base.missing_entrypoints_callback(names)`

Raise MissingHookError with comma-separated list of missing hooks

`ironic_inspector.plugins.base.node_not_found_hook_manager(*args)`

`ironic_inspector.plugins.base.processing_hooks_manager(*args)`

Create a Stevedore extension manager for processing hooks.

#### Parameters

**args** arguments to pass to the hooks constructor.

`ironic_inspector.plugins.base.reset()`

Reset cached managers.

`ironic_inspector.plugins.base.rule_actions_manager()`

Create a Stevedore extension manager for actions in rules.

`ironic_inspector.plugins.base.rule_conditions_manager()`

Create a Stevedore extension manager for conditions in rules.

`ironic_inspector.plugins.base.validate_processing_hooks()`

Validate the enabled processing hooks.

#### Raises

MissingHookError on missing or failed to load hooks

#### Raises

RuntimeError on validation failure

#### Returns

the list of hooks passed validation

## ironic\_inspector.plugins.base\_physnet module

**class** `ironic_inspector.plugins.base_physnet.BasePhysnetHook`

Bases: *ProcessingHook*

Base class for plugins that assign a physical network to ports.

The mechanism for mapping a port to a physical network should be provided by a subclass via the `get_physnet()` method.

**before\_update**(*introspection\_data*, *node\_info*, *\*\*kwargs*)

Process introspection data and patch port physical network.

**abstract get\_physnet**(*port*, *iface\_name*, *introspection\_data*)

Return a physical network to apply to a port.

Subclasses should implement this method to determine how to map a port to a physical network.

#### Parameters

- **port** The ironic port to patch.
- **iface\_name** Name of the interface.
- **introspection\_data** Introspection data.

#### Returns

The physical network to set, or None.

### ironic\_inspector.plugins.capabilities module

Gather capabilities from inventory.

**class** `ironic_inspector.plugins.capabilities.CapabilitiesHook`

Bases: *ProcessingHook*

Processing hook for detecting capabilities.

**before\_update**(*introspection\_data*, *node\_info*, *\*\*kwargs*)

Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

#### Parameters

- **introspection\_data** processed data from the ramdisk.
- **node\_info** NodeInfo instance.
- **kwargs** used for extensibility without breaking existing hooks.

#### Returns

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

### ironic\_inspector.plugins.discovery module

Enroll node not found hook hook.

`ironic_inspector.plugins.discovery.enroll_node_not_found_hook`(*introspection\_data*, *\*\*kwargs*)

## ironic\_inspector.plugins.example module

Example plugin.

**class** `ironic_inspector.plugins.example.ExampleProcessingHook`

Bases: *ProcessingHook*

**before\_processing**(*introspection\_data*, **\*\*kwargs**)

Hook to run before any other data processing.

This hook is run even before sanity checks.

### Parameters

- **introspection\_data** raw information sent by the ramdisk, may be modified by the hook.
- **kwargs** used for extensibility without breaking existing hooks

### Returns

nothing.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)

Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

### Parameters

- **introspection\_data** processed data from the ramdisk.
- **node\_info** NodeInfo instance.
- **kwargs** used for extensibility without breaking existing hooks.

### Returns

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

**class** `ironic_inspector.plugins.example.ExampleRuleAction`

Bases: *RuleActionPlugin*

**apply**(*node\_info*, *params*, **\*\*kwargs**)

Run action on successful rule match.

### Parameters

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

### Raises

`utils.Error` on failure

`ironic_inspector.plugins.example.example_not_found_hook`(*introspection\_data*, **\*\*kwargs**)

### `ironic_inspector.plugins.extraHardware` module

Plugin to store extra hardware information in Swift.

Stores the value of the data key returned by the ramdisk as a JSON encoded string in a Swift object. The object is named `extraHardware-<node uuid>` and is stored in the inspector container.

**class** `ironic_inspector.plugins.extraHardware.ExtraHardwareHook`

Bases: `ProcessingHook`

Processing hook for saving extra hardware information in Swift.

**before\_update**(*introspection\_data*, *node\_info*, *\*\*kwargs*)

Stores the data key from `introspection_data` in Swift.

If the data key exists, updates Ironic extra column `hardware_swift_object` key to the name of the Swift object, and stores the data in the inspector container in Swift.

Otherwise, it does nothing.

### `ironic_inspector.plugins.introspection_data` module

Backends for storing introspection data.

**class** `ironic_inspector.plugins.introspection_data.BaseStorageBackend`

Bases: `object`

**abstract get**(*node\_uuid*, *processed=True*, *get\_json=False*)

Get introspected data from storage backend.

#### Parameters

- **node\_uuid** node UUID.
- **processed** Specify whether the data to be retrieved is processed or not.
- **get\_json** Specify whether return the introspection data in json format, string value is returned if False.

#### Returns

the introspection data.

#### Raises

`IntrospectionDataStoreDisabled` if storage backend is disabled.

**abstract save**(*node\_uuid*, *data*, *processed=True*)

Save introspected data to storage backend.

#### Parameters

- **node\_uuid** node UUID.
- **data** the introspected data to be saved, in dict format.
- **processed** Specify whether the data to be saved is processed or not.

#### Raises

`IntrospectionDataStoreDisabled` if storage backend is disabled.



**class** `ironic_inspector.plugins.introspection_data.DatabaseStore`

Bases: `object`

**get**(*node\_uuid*, *processed=True*, *get\_json=False*)

**save**(*node\_uuid*, *data*, *processed=True*)

**class** `ironic_inspector.plugins.introspection_data.NoStore`

Bases: `BaseStorageBackend`

**get**(*node\_uuid*, *processed=True*, *get\_json=False*)

Get introspected data from storage backend.

#### Parameters

- **node\_uuid** node UUID.
- **processed** Specify whether the data to be retrieved is processed or not.
- **get\_json** Specify whether return the introspection data in json format, string value is returned if False.

#### Returns

the introspection data.

#### Raises

`IntrospectionDataStoreDisabled` if storage backend is disabled.

**save**(*node\_uuid*, *data*, *processed=True*)

Save introspected data to storage backend.

#### Parameters

- **node\_uuid** node UUID.
- **data** the introspected data to be saved, in dict format.
- **processed** Specify whether the data to be saved is processed or not.

#### Raises

`IntrospectionDataStoreDisabled` if storage backend is disabled.

**class** `ironic_inspector.plugins.introspection_data.SwiftStore`

Bases: `object`

**get**(*node\_uuid*, *processed=True*, *get\_json=False*)

**save**(*node\_uuid*, *data*, *processed=True*)

## `ironic_inspector.plugins.lldp_basic` module

LLDP Processing Hook for basic TLVs

**class** `ironic_inspector.plugins.lldp_basic.LLDPBasicProcessingHook`

Bases: `ProcessingHook`

Process mandatory and optional LLDP packet fields

Loop through raw LLDP TLVs and parse those from the basic management, 802.1, and 802.3 TLV sets. Store parsed data back to the ironic-inspector database.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)  
Process LLDP data and update all\_interfaces with processed data

### **ironic\_inspector.plugins.local\_link\_connection module**

Generic LLDP Processing Hook

#### **class**

**ironic\_inspector.plugins.local\_link\_connection.GenericLocalLinkConnectionHook**

Bases: *ProcessingHook*

Process mandatory LLDP packet fields

Non-vendor specific LLDP packet fields processed for each NIC found for a baremetal node, port ID and chassis ID. These fields if found and if valid will be saved into the local link connection info port id and switch id fields on the Ironic port that represents that NIC.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)  
Process LLDP data and patch Ironic port local link connection

### **ironic\_inspector.plugins.pci\_devices module**

Gather and distinguish PCI devices from inventory.

**class** **ironic\_inspector.plugins.pci\_devices.PciDevicesHook**

Bases: *ProcessingHook*

Processing hook for counting and distinguishing various PCI devices.

That information can be later used by nova for node scheduling.

**aliases** = {}

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)  
Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

#### **Parameters**

- **introspection\_data** processed data from the ramdisk.
- **node\_info** NodeInfo instance.
- **kwargs** used for extensibility without breaking existing hooks.

#### **Returns**

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

### **ironic\_inspector.plugins.physnet\_cidr\_map module**

Port Physical Network Hook

**class** `ironic_inspector.plugins.physnet_cidr_map.PhysnetCidrMapHook`

Bases: *BasePhysnetHook*

Process port physical network

Set the `physical_network` field of baremetal ports based on a cidr to physical network mapping in the configuration.

**get\_physnet**(*port, iface\_name, introspection\_data*)

Return a physical network to apply to a port.

#### Parameters

- **port** The ironic port to patch.
- **iface\_name** Name of the interface.
- **introspection\_data** Introspection data.

#### Returns

The physical network to set, or None.

### **ironic\_inspector.plugins.raid\_device module**

Gather root device hint from recognized block devices.

**class** `ironic_inspector.plugins.raid_device.RaidDeviceDetection`

Bases: *ProcessingHook*

Processing hook for learning the root device after RAID creation.

The plugin can figure out the root device in 2 runs. First, it saves the discovered block device serials in `node.extra`. The second run will check the difference between the recently discovered block devices and the previously saved ones. After saving the root device in `node.properties`, it will delete the temporarily saved block device serials in `node.extra`.

This way, it helps to figure out the root device hint in cases when otherwise Ironic doesn't have enough information to do so. Such a usecase is DRAC RAID configuration where the BMC doesn't provide any useful information about the created RAID disks. Using this plugin immediately before and after creating the root RAID device will solve the issue of root device hints.

In cases where there's no RAID volume on the node, the standard plugin will fail due to the missing `local_gb` value. This plugin fakes the missing value, until it's corrected during later runs. Note, that for this to work the plugin needs to take precedence over the standard plugin.

**before\_processing**(*introspection\_data, \*\*kwargs*)

Adds fake `local_gb` value if it's missing from `introspection_data`.

**before\_update**(*introspection\_data, node\_info, \*\*kwargs*)

Hook to run before Ironic node update.

This hook is run after node is found and ports are created, just before the node is updated with the data.

#### Parameters

- **introspection\_data** processed data from the ramdisk.
- **node\_info** NodeInfo instance.

- **kwargs** used for extensibility without breaking existing hooks.

**Returns**

nothing.

[RFC 6902] - <http://tools.ietf.org/html/rfc6902>

## ironic\_inspector.plugins.rules module

Standard plugins for rules API.

**class** `ironic_inspector.plugins.rules.AddTraitAction`

Bases: `RuleActionPlugin`

**REQUIRED\_PARAMS** = {'name'}

Set with names of required parameters.

**apply**(*node\_info*, *params*, **\*\*kwargs**)

Run action on successful rule match.

**Parameters**

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

`utils.Error` on failure

**class** `ironic_inspector.plugins.rules.ContainsCondition`

Bases: `ReCondition`

**check**(*node\_info*, *field*, *params*, **\*\*kwargs**)

Check if condition holds for a given field.

**Parameters**

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

**ValueError** on unacceptable field value

**Returns**

True if check succeeded, otherwise False

**class** `ironic_inspector.plugins.rules.EmptyCondition`

Bases: `RuleConditionPlugin`

**ALLOW\_NONE** = True

Whether this condition accepts None when field is not found.

**REQUIRED\_PARAMS** = {}

Set with names of required parameters.

**check**(*node\_info*, *field*, *params*, *\*\*kwargs*)

Check if condition holds for a given field.

#### Parameters

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

**ValueError** on unacceptable field value

#### Returns

True if check succeeded, otherwise False

**class** `ironic_inspector.plugins.rules.EqCondition`

Bases: *SimpleCondition*

**op**(*b*, /)

Same as `a == b`.

**class** `ironic_inspector.plugins.rules.ExtendAttributeAction`

Bases: *RuleActionPlugin*

**FORMATTED\_PARAMS** = ['value']

List of params will be formatted with python format.

**OPTIONAL\_PARAMS** = {'unique'}

Set with names of optional parameters.

**REQUIRED\_PARAMS** = {'path', 'value'}

Set with names of required parameters.

**apply**(*node\_info*, *params*, *\*\*kwargs*)

Run action on successful rule match.

#### Parameters

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

`utils.Error` on failure

**class** `ironic_inspector.plugins.rules.FailAction`

Bases: *RuleActionPlugin*

**REQUIRED\_PARAMS** = {'message'}

Set with names of required parameters.

**apply**(*node\_info*, *params*, *\*\*kwargs*)

Run action on successful rule match.

**Parameters**

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

utils.Error on failure

**class** `ironic_inspector.plugins.rules.GeCondition`

Bases: *SimpleCondition*

**op**(*b*, /)

Same as  $a \geq b$ .

**class** `ironic_inspector.plugins.rules.GtCondition`

Bases: *SimpleCondition*

**op**(*b*, /)

Same as  $a > b$ .

**class** `ironic_inspector.plugins.rules.LeCondition`

Bases: *SimpleCondition*

**op**(*b*, /)

Same as  $a \leq b$ .

**class** `ironic_inspector.plugins.rules.LtCondition`

Bases: *SimpleCondition*

**op**(*b*, /)

Same as  $a < b$ .

**class** `ironic_inspector.plugins.rules.MatchesCondition`

Bases: *ReCondition*

**check**(*node\_info*, *field*, *params*, *\*\*kwargs*)

Check if condition holds for a given field.

**Parameters**

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

**ValueError** on unacceptable field value

**Returns**

True if check succeeded, otherwise False

**class** `ironic_inspector.plugins.rules.NeCondition`

Bases: [SimpleCondition](#)

**op**(*b, /*)

Same as `a != b`.

**class** `ironic_inspector.plugins.rules.NetCondition`

Bases: [RuleConditionPlugin](#)

**check**(*node\_info, field, params, \*\*kwargs*)

Check if condition holds for a given field.

#### Parameters

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

**ValueError** on unacceptable field value

#### Returns

True if check succeeded, otherwise False

**validate**(*params, \*\*kwargs*)

Validate params passed during creation.

Default implementation checks for presence of fields from `REQUIRED_PARAMS` and fails for unexpected fields (not from `REQUIRED_PARAMS + OPTIONAL_PARAMS`).

#### Parameters

- **params** params as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

**ValueError** on validation failure

**class** `ironic_inspector.plugins.rules.ReCondition`

Bases: [RuleConditionPlugin](#)

**validate**(*params, \*\*kwargs*)

Validate params passed during creation.

Default implementation checks for presence of fields from `REQUIRED_PARAMS` and fails for unexpected fields (not from `REQUIRED_PARAMS + OPTIONAL_PARAMS`).

#### Parameters

- **params** params as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

**ValueError** on validation failure

**class** `ironic_inspector.plugins.rules.RemoveTraitAction`

Bases: `RuleActionPlugin`

**REQUIRED\_PARAMS** = {'name'}

Set with names of required parameters.

**apply**(`node_info`, `params`, `**kwargs`)

Run action on successful rule match.

**Parameters**

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

`utils.Error` on failure

**class** `ironic_inspector.plugins.rules.SetAttributeAction`

Bases: `RuleActionPlugin`

**FORMATTED\_PARAMS** = ['value']

List of params will be formatted with python format.

**OPTIONAL\_PARAMS** = {'reset\_interfaces', 'value'}

Set with names of optional parameters.

**REQUIRED\_PARAMS** = {'path'}

Set with names of required parameters.

**apply**(`node_info`, `params`, `**kwargs`)

Run action on successful rule match.

**Parameters**

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

`utils.Error` on failure

**validate**(`params`, `**kwargs`)

Validate params passed during creation.

Default implementation checks for presence of fields from `REQUIRED_PARAMS` and fails for unexpected fields (not from `REQUIRED_PARAMS` + `OPTIONAL_PARAMS`).

**Parameters**

- **params** params as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

**Raises**

`ValueError` on validation failure



**class** `ironic_inspector.plugins.rules.SetCapabilityAction`

Bases: [RuleActionPlugin](#)

**FORMATTED\_PARAMS** = ['value']

List of params will be formatted with python format.

**OPTIONAL\_PARAMS** = {'value'}

Set with names of optional parameters.

**REQUIRED\_PARAMS** = {'name'}

Set with names of required parameters.

**apply**(*node\_info*, *params*, **\*\*kwargs**)

Run action on successful rule match.

#### Parameters

- **node\_info** NodeInfo object
- **params** parameters as a dictionary
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

`utils.Error` on failure

**class** `ironic_inspector.plugins.rules.SimpleCondition`

Bases: [RuleConditionPlugin](#)

**check**(*node\_info*, *field*, *params*, **\*\*kwargs**)

Check if condition holds for a given field.

#### Parameters

- **node\_info** NodeInfo object
- **field** field value
- **params** parameters as a dictionary, changing it here will change what will be stored in database
- **kwargs** used for extensibility without breaking existing plugins

#### Raises

**ValueError** on unacceptable field value

#### Returns

True if check succeeded, otherwise False

**op** = None

`ironic_inspector.plugins.rules.coerce`(*value*, *expected*)

## `ironic_inspector.plugins.standard` module

Standard set of plugins.

**class** `ironic_inspector.plugins.standard.RamdiskErrorHook`

Bases: [\*ProcessingHook\*](#)

Hook to process error send from the ramdisk.

**before\_processing**(*introspection\_data*, **\*\*kwargs**)

Hook to run before any other data processing.

This hook is run even before sanity checks.

**Parameters**

- **introspection\_data** raw information sent by the ramdisk, may be modified by the hook.
- **kwargs** used for extensibility without breaking existing hooks

**Returns**

nothing.

**class** `ironic_inspector.plugins.standard.RootDiskSelectionHook`

Bases: [\*ProcessingHook\*](#)

Smarter root disk selection using Ironic root device hints.

This hook must always go before SchedulerHook, otherwise root\_disk field might not be updated.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)

Process root disk information.

**class** `ironic_inspector.plugins.standard.SchedulerHook`

Bases: [\*ProcessingHook\*](#)

Nova scheduler required properties.

**KEYS** = ('cpu', 'cpu\_arch', 'memory\_mb')

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)

Update node with scheduler properties.

**class** `ironic_inspector.plugins.standard.ValidateInterfacesHook`

Bases: [\*ProcessingHook\*](#)

Hook to validate network interfaces.

**before\_processing**(*introspection\_data*, **\*\*kwargs**)

Validate information about network interfaces.

**before\_update**(*introspection\_data*, *node\_info*, **\*\*kwargs**)

Create new ports and drop ports that are not present in the data.

## Module contents

[ironic\\_inspector.pxe\\_filter package](#)

[Submodules](#)

## ironic\_inspector.pxe\_filter.base module

Base code for PXE boot filtering.

**class** `ironic_inspector.pxe_filter.base.BaseFilter`

Bases: *FilterDriver*

The generic PXE boot filtering interface implementation.

This driver doesn't do anything but provides a basic synchronization and initialization logic for some drivers to reuse. Subclasses have to provide a custom `sync()` method.

**fsm** = `<automaton.machines.FiniteMachine object>`

**fsm\_reset\_on\_error()**

Reset the filter driver upon generic exception.

The context is `self.fsm`. The `automaton.exceptions.NotFound` error is cast to the `InvalidFilterDriverState` error. Other exceptions trigger `self.reset()`

**Raises**

`InvalidFilterDriverState`

**Returns**

nothing.

**get\_periodic\_sync\_task()**

Get periodic sync task for the filter.

The periodic task returned is casting the `InvalidFilterDriverState` to the `periodics.NeverAgain` exception to quit looping.

**Raises**

`periodics.NeverAgain`

**Returns**

a periodic task to be run in the background.

**init\_filter()**

Base driver initialization logic. Locked.

**Raises**

`InvalidFilterDriverState`

**Returns**

nothing.

**reset()**

Reset internal driver state.

This method is called by the `fsm_context` manager upon exception as well as by the `tear_down_filter` method. A subclass might wish to override as necessary, though must not lock the driver. The overriding subclass should up-call.

**Returns**

nothing.

**property state**

Current driver state.

**sync**(*ironic*)

Base driver sync logic. Locked.

**Parameters**

**ironic** obligatory ironic client instance

**Returns**

nothing.

**tear\_down\_filter**()

Base driver tear down logic. Locked.

**Returns**

nothing.

**class** `ironic_inspector.pxe_filter.base.Events`

Bases: `object`

PXE filter driver transitions.

**initialize** = 'initialize'

**reset** = 'reset'

**sync** = 'sync'

**exception** `ironic_inspector.pxe_filter.base.InvalidFilterDriverState`

Bases: `RuntimeError`

The fsm of the filter driver raised an error.

**class** `ironic_inspector.pxe_filter.base.NoopFilter`

Bases: `BaseFilter`

A trivial PXE boot filter.

**get\_periodic\_sync\_task**()

Get periodic sync task for the filter.

The periodic task returned is casting the `InvalidFilterDriverState` to the `periodics.NeverAgain` exception to quit looping.

**Raises**

`periodics.NeverAgain`

**Returns**

a periodic task to be run in the background.

**class** `ironic_inspector.pxe_filter.base.States`

Bases: `object`

PXE filter driver states.

**initialized** = 'initialized'

**uninitialized** = 'uninitialized'

`ironic_inspector.pxe_filter.base.driver()`

Get the driver for the PXE filter.

**Returns**

the singleton PXE filter driver object.

`ironic_inspector.pxe_filter.base.get_active_macs(ironic)`

`ironic_inspector.pxe_filter.base.get_inactive_macs(ironic)`

`ironic_inspector.pxe_filter.base.get_ironic_macs(ironic)`

`ironic_inspector.pxe_filter.base.locked_driver_event(event)`

Call driver method having processed the fsm event.

### **ironic\_inspector.pxe\_filter.dnsmasq module**

**class** `ironic_inspector.pxe_filter.dnsmasq.DnsmasqFilter`

Bases: *BaseFilter*

The dnsmasq PXE filter driver.

A pxe filter driver implementation that controls access to dnsmasq through amending its configuration.

**init\_filter()**

Performs an initial sync with ironic and starts dnsmasq.

The initial `_sync()` call reduces the chances dnsmasq might lose some inotify deny list events by prefetching the list before dnsmasq is started.

**Raises**

OSError, IOError.

**Returns**

None.

**reset()**

Stop dnsmasq and upcall reset.

**sync(ironic)**

Sync dnsmasq configuration with current Ironic&Inspector state.

Polls all ironic ports. Those being inspected, the active ones, are added to the allow list while the rest are added to the deny list in the dnsmasq configuration.

**Parameters**

**ironic** an ironic client instance.

**Raises**

OSError, IOError.

**Returns**

None.

## ironic\_inspector.pxe\_filter.interface module

The code of the PXE boot filtering interface.

**class** `ironic_inspector.pxe_filter.interface.FilterDriver`

Bases: `object`

The PXE boot filtering interface.

**abstract** `get_periodic_sync_task()`

Get periodic sync task for the filter.

**Returns**

a periodic task to be run in the background.

**abstract** `init_filter()`

Initialize the internal driver state.

This method should be idempotent and may perform system-wide filter state changes. Can be synchronous.

**Returns**

nothing.

**abstract** `sync(ironic)`

Synchronize the filter with ironic and inspector.

To be called both periodically and as needed by inspector. The filter should tear down its internal state if the sync method raises in order to propagate filtering exception between periodic and on-demand sync call. To this end, a driver should raise from the sync call if its internal state isnt properly initialized.

**Parameters**

**ironic** an ironic client instance.

**Returns**

nothing.

**abstract** `tear_down_filter()`

Reset the filter.

This method should be idempotent and may perform system-wide filter state changes. Can be synchronous.

**Returns**

nothing.

## ironic\_inspector.pxe\_filter.iptables module

**class** `ironic_inspector.pxe_filter.iptables.IptablesFilter`

Bases: `BaseFilter`

A PXE boot filtering interface implementation.

**init\_filter()**

Base driver initialization logic. Locked.

**Raises**

`InvalidFilterDriverState`

**Returns**

nothing.

**reset()**

Reset internal driver state.

This method is called by the `fsm_context` manager upon exception as well as by the `tear_down_filter` method. A subclass might wish to override as necessary, though must not lock the driver. The overriding subclass should up-call.

**Returns**

nothing.

**sync(*ironic*)**

Sync firewall filter rules for introspection.

Gives access to PXE boot port for any machine, except for those, whose MAC is registered in Ironic and is not on introspection right now.

This function is called from both introspection initialization code and from periodic task. This function is supposed to be resistant to unexpected iptables state.

`init()` function must be called once before any call to this function. This function is using `eventlet` semaphore to serialize access from different green threads.

**Parameters**

**ironic** an ironic client instance.

**Returns**

nothing.

## Module contents

### Submodules

#### **ironic\_inspector.api\_tools module**

Generic Rest Api tools.

`ironic_inspector.api_tools.limit_field(value)`

Fetch the pagination limit field from `flask.request.args`.

**Returns**

the limit

`ironic_inspector.api_tools.marker_field(value)`

Fetch the pagination marker field from `flask.request.args`.

**Returns**

an uuid

`ironic_inspector.api_tools.raises_coercion_exceptions(fn)`

Convert coercion function exceptions to `utils.Error`.

**Raises**

`utils.Error` when the coercion function raises an `AssertionError` or a `ValueError`

`ironic_inspector.api_tools.request_field(field_name)`

Decorate a function that coerces the specified field.

**Parameters**

**field\_name** name of the field to fetch

**Returns**

a decorator

`ironic_inspector.api_tools.state_field(value)`

Fetch the pagination state field from `flask.request.args`.

**Returns**

list of the state(s)

### **ironic\_inspector.introspect module**

Handling introspection request.

`ironic_inspector.introspect.abort(node_id, token=None)`

Abort running introspection.

**Parameters**

- **node\_id** node UUID or name
- **token** authentication token

**Raises**

Error

`ironic_inspector.introspect.introspect(node_id, manage_boot=True, token=None)`

Initiate hardware properties introspection for a given node.

**Parameters**

- **node\_id** node UUID or name
- **manage\_boot** whether to manage boot for this node
- **token** authentication token

**Raises**

Error

### **ironic\_inspector.introspection\_state module**

Introspection state.

**class** `ironic_inspector.introspection_state.Events`

Bases: `object`

Events that change introspection state.

**abort** = 'abort'

**abort\_end** = 'abort\_end'



```
classmethod all()
    Return a list of all events.

error = 'error'

finish = 'finish'

process = 'process'

reapply = 'reapply'

start = 'start'

timeout = 'timeout'

wait = 'wait'
```

```
class ironic_inspector.introspection_state.States
```

```
    Bases: object

    States of an introspection.

    aborting = 'aborting'

    classmethod all()
        Return a list of all states.

    enrolling = 'enrolling'

    error = 'error'

    finished = 'finished'

    processing = 'processing'

    reapplying = 'reapplying'

    starting = 'starting'

    waiting = 'waiting'
```

### **ironic\_inspector.main module**

```
ironic_inspector.main.add_version_headers(res)
```

```
ironic_inspector.main.api(path, is_public_api=False, rule=None, verb_to_rule_map=None,
                          **flask_kwargs)
```

Decorator to wrap api methods.

Performs flask routing, exception conversion, generation of oslo context for request and API access policy enforcement.

#### **Parameters**

- **path** flask app route path
- **is\_public\_api** whether this API path should be treated as public, with minimal access enforcement

- **rule** API access policy rule to enforce. If rule is None, the default policy rule will be enforced, which is deny all if not overridden in policy config file.
- **verb\_to\_rule\_map** if both rule and this are given, defines mapping between http verbs (uppercase) and strings to format the rule string with
- **kwargs** all the rest kwargs are passed to flask app.route

`ironic_inspector.main.api_continue()`

`ironic_inspector.main.api_introspection(node_id)`

`ironic_inspector.main.api_introspection_abort(node_id)`

`ironic_inspector.main.api_introspection_data(node_id)`

`ironic_inspector.main.api_introspection_reapply(node_id)`

`ironic_inspector.main.api_introspection_statuses()`

`ironic_inspector.main.api_introspection_unprocessed_data(node_id)`

`ironic_inspector.main.api_root()`

`ironic_inspector.main.api_rule(uuid)`

`ironic_inspector.main.api_rules()`

`ironic_inspector.main.check_api_version()`

`ironic_inspector.main.convert_exceptions(func)`

`ironic_inspector.main.create_link_object(urls)`

`ironic_inspector.main.error_response(exc, code=500)`

`ironic_inspector.main.generate_introspection_status(node)`

Return a dict representing current node status.

**Parameters**

**node** a NodeInfo instance

**Returns**

dictionary

`ironic_inspector.main.generate_resource_data(resources)`

`ironic_inspector.main.get_app()`

Get the flask instance.

`ironic_inspector.main.get_client_compat()`

`ironic_inspector.main.get_random_topic()`

`ironic_inspector.main.handle_404(error)`

`ironic_inspector.main.rule_repr(rule, short)`

`ironic_inspector.main.start_coordinator()`

Create a coordinator instance for non-standalone case.

`ironic_inspector.main.version_root(version)`

## `ironic_inspector.node_cache` module

Cache for nodes currently under introspection.

**class** `ironic_inspector.node_cache.NodeInfo`(*uuid, version\_id=None, state=None, started\_at=None, finished\_at=None, error=None, node=None, ports=None, ironic=None, manage\_boot=True*)

Bases: `object`

Record about a node in the cache.

This class optionally allows to acquire a lock on a node. Note that the class instance itself is NOT thread-safe, you need to create a new instance for every thread.

**acquire\_lock**(*blocking=True*)

Acquire a lock on the associated node.

Exits with success if a lock is already acquired using this `NodeInfo` object.

### Parameters

**blocking** if `True`, wait for lock to be acquired, otherwise return immediately.

### Returns

boolean value, whether lock was acquired successfully

**add\_attribute**(*name, value*)

Store look up attribute for a node in the database.

### Parameters

- **name** attribute name
- **value** attribute value or list of possible values

**add\_trait**(*trait, ironic=None*)

Add a trait to the node.

### Parameters

- **trait** trait to add
- **ironic** Ironic client to use instead of `self.ironic`

### property attributes

Node look up attributes as a dict.

**commit**()

Commit current node status into the database.

**create\_ports**(*ports, ironic=None*)

Create one or several ports for this node.

### Parameters

- **ports** List of ports with all their attributes e.g [{mac: xx, ip: xx, client\_id: None}, {mac: xx, ip: None, client\_id: None}] It also support the old style of list of macs. A warning is issued if port already exists on a node.
- **ironic** Ironic client to use instead of self.ironic

**delete\_port**(*port, ironic=None*)

Delete port.

**Parameters**

- **port** port object or its MAC
- **ironic** Ironic client to use instead of self.ironic

**finished**(*event, error=None*)

Record status for this node and process a terminal transition.

Also deletes look up attributes from the cache.

**Parameters**

- **event** the event to process
- **error** error message

**classmethod from\_row**(*row, ironic=None, node=None*)

Construct NodeInfo from a database row.

**fsm\_event**(*event, strict=False*)

Update node\_info.state based on a fsm.process\_event(event) call.

An AutomatonException triggers an error event. If strict, node\_info.finished(istate.Events.error, error=str(exc)) is called with the AutomatonException instance and a EventError raised.

**Parameters**

**event** an event to process by the fsm

**Strict**

whether to fail the introspection upon an invalid event

**Raises**

NodeStateInvalidEvent

**get\_by\_path**(*path*)

Get field value by ironic-style path (e.g. /extra/foo).

**Parameters**

**path** path to a field

**Returns**

field value

**Raises**

KeyError if field was not found

**invalidate\_cache**()

Clear all cached info, so that its reloaded next time.

**property ironic**

Ironic client instance.

**property manage\_boot**

Whether to manage boot for this node.

**node(*ironic=None*)**

Get Ironic node object associated with the cached node record.

**property options**

Node introspection options as a dict.

**patch(*patches, ironic=None, \*\*kwargs*)**

Apply JSON patches to a node.

Refreshes cached node instance.

**Parameters**

- **patches** JSON patches to apply
- **ironic** Ironic client to use instead of self.ironic
- **kwargs** Arguments to pass to ironicclient.

**Raises**

openstacksdk exceptions

**patch\_port(*port, patches, ironic=None*)**

Apply JSON patches to a port.

**Parameters**

- **port** port object or its MAC
- **patches** JSON patches to apply
- **ironic** Ironic client to use instead of self.ironic

**ports(*ironic=None*)**

Get Ironic port objects associated with the cached node record.

This value is cached as well, use `invalidate_cache()` to clean.

**Returns**

dict MAC -> port object

**release\_lock()**

Release a lock on a node.

Does nothing if lock was not acquired using this NodeInfo object.

**remove\_trait(*trait, ironic=None*)**

Remove a trait from the node.

**Parameters**

- **trait** trait to add
- **ironic** Ironic client to use instead of self.ironic

**replace\_field**(*path, func, \*\*kwargs*)

Replace a field on ironic node.

**Parameters**

- **path** path to a field as used by the ironic client
- **func** function accepting an old value and returning a new one
- **kwargs** if default value is passed here, it will be used when no existing value is found.

**Raises**

KeyError if value is not found and default is not set

**Raises**

everything that patch() may raise

**set\_option**(*name, value*)

Set an option for a node.

**property state**

State of the node\_info object.

**update\_capabilities**(*ironic=None, \*\*caps*)

Update capabilities on a node.

**Parameters**

- **caps** capabilities to update
- **ironic** Ironic client to use instead of self.ironic

**update\_properties**(*ironic=None, \*\*props*)

Update properties on a node.

**Parameters**

- **props** properties to update
- **ironic** Ironic client to use instead of self.ironic

**property version\_id**

Deprecated - Get the version id

`ironic_inspector.node_cache.active_macs()`

List all MACs that are on introspection right now.

`ironic_inspector.node_cache.add_node(uuid, state, manage_boot=True, **attributes)`

Store information about a node under introspection.

All existing information about this node is dropped. Empty values are skipped.

**Parameters**

- **uuid** Ironic node UUID
- **state** The initial state of the node
- **manage\_boot** whether to manage boot for this node

- **attributes** attributes known about this node (like macs, BMC etc); also ironic client instance may be passed under ironic

**Returns**

NodeInfo

`ironic_inspector.node_cache.clean_up()`

Clean up the cache.

Finish introspection for timed out nodes.

**Returns**

list of timed out node UUIDs

`ironic_inspector.node_cache.create_node(driver, ironic=None, **attributes)`

Create ironic node and cache it.

- Create new node in ironic.
- Cache it in inspector.
- Sets node\_info state to enrolling.

**Parameters**

- **driver** driver for Ironic node.
- **ironic** ironic client instance.
- **attributes** dict, additional keyword arguments to pass to the ironic client on node creation.

**Returns**

NodeInfo, or None in case error happened.

`ironic_inspector.node_cache.delete_nodes_not_in_list(uuids)`

Delete nodes which dont exist in Ironic node UUIDs.

**Parameters**

**uuids** Ironic node UUIDs

`ironic_inspector.node_cache.find_node(**attributes)`

Find node in cache.

Looks up a node based on attributes in a best-match fashion. This function acquires a lock on a node.

**Parameters**

**attributes** attributes known about this node (like macs, BMC etc) also ironic client instance may be passed under ironic

**Returns**

structure NodeInfo with attributes uuid and created\_at

**Raises**

Error if node is not found or multiple nodes match the attributes

`ironic_inspector.node_cache.fsm_event_after(event, strict=False)`

Trigger an fsm event after the function execution.

It is assumed the first function arg of the decorated function is always a NodeInfo instance.

### **Parameters**

- **event** the event to process after the function call
- **strict** make an invalid fsm event trigger an error event

`ironic_inspector.node_cache.fsm_event_before(event, strict=False)`

Trigger an fsm event before the function execution.

It is assumed the first function arg of the decorated function is always a NodeInfo instance.

### **Parameters**

- **event** the event to process before the function call
- **strict** make an invalid fsm event trigger an error event

`ironic_inspector.node_cache.fsm_transition(event, reentrant=True, **exc_kwargs)`

Decorate a function to perform a (non-)reentrant transition.

If True, reentrant transition will be performed at the end of a function call. If False, the transition will be performed before the function call. The function is decorated with the `triggers_fsm_error_transition` decorator as well.

### **Parameters**

- **event** the event to bind the transition to.
- **reentrant** whether the transition is reentrant.
- **exc\_kwargs** passed on to the `triggers_fsm_error_transition` decorator

`ironic_inspector.node_cache.get_introspection_data(node_id, processed=True)`

Get introspection data for this node.

### **Parameters**

- **node\_id** node UUID.
- **processed** Specify the type of introspected data, set to False indicates retrieving the unprocessed data.

### **Returns**

A dictionary representation of introspected data

`ironic_inspector.node_cache.get_node(node_id, ironic=None)`

Get node from cache.

### **Parameters**

- **node\_id** node UUID or name.
- **ironic** optional ironic client instance

### **Returns**

structure NodeInfo.



`ironic_inspector.node_cache.get_node_list(ironic=None, marker=None, limit=None, state=None)`

Get node list from the cache.

The list of the nodes is ordered based on the (started\_at, uuid) attribute pair, newer items first.

#### Parameters

- **ironic** optional ironic client instance
- **marker** pagination marker (an UUID or None)
- **limit** pagination limit; None for default CONF.api\_max\_limit
- **state** list of states for the filter; None for no state filter

#### Returns

a list of NodeInfo instances.

`ironic_inspector.node_cache.introspection_active()`

Check if introspection is active for at least one node.

`ironic_inspector.node_cache.record_node(ironic=None, bmc_addresses=None, macs=None)`

Create a cache record for a known active node.

#### Parameters

- **ironic** ironic client instance.
- **bmc\_addresses** list of BMC addresses.
- **macs** list of MAC addresses.

#### Returns

NodeInfo

`ironic_inspector.node_cache.release_lock(func)`

Decorate a node\_info-function to release the node\_info lock.

Assumes the first parameter of the function func is always a NodeInfo instance.

`ironic_inspector.node_cache.start_introspection(uuid, **kwargs)`

Start the introspection of a node.

If a node\_info record exists in the DB, a start transition is used rather than dropping the record in order to check for the start transition validity in particular node state.

#### Parameters

- **uuid** Ironic node UUID
- **kwargs** passed on to add\_node()

#### Raises

NodeStateInvalidEvent in case the start transition is invalid in the current node state

#### Raises

NodeStateRaceCondition if a mismatch was detected between the node\_info cache and the DB

#### Returns

NodeInfo

`ironic_inspector.node_cache.store_introspection_data`(*node\_id*, *introspection\_data*,  
*processed=True*)

Store introspection data for this node.

### Parameters

- **node\_id** node UUID.
- **introspection\_data** A dictionary of introspection data
- **processed** Specify the type of introspected data, set to False indicates the data is unprocessed.

`ironic_inspector.node_cache.triggers_fsm_error_transition`(*errors*=(*<class*  
*'Exception'>*, ),  
*no\_errors*=(*<class*  
*'ironic\_inspector.utils.NodeStateInvalidEvent'*,  
*<class*  
*'ironic\_inspector.utils.NodeStateRaceCondition'*))

Trigger an fsm error transition upon certain errors.

It is assumed the first function arg of the decorated function is always a NodeInfo instance.

### Parameters

- **errors** a tuple of exceptions upon which an error event is triggered. Re-raised.
- **no\_errors** a tuple of exceptions that wont trigger the error event.

## ironic\_inspector.policy module

`ironic_inspector.policy.authorize`(*rule*, *target*, *creds*, *\*args*, *\*\*kwargs*)

A shortcut for `policy.Enforcer.authorize()`

Checks authorization of a rule against the target and credentials, and raises an exception if the rule is not defined. *args* and *kwargs* are passed directly to `oslo.policy.Enforcer.authorize` Always returns True if `CONF.auth_strategy != keystone`.

### Parameters

- **rule** name of a registered `oslo.policy` rule
- **target** dict-like structure to check rule against
- **creds** dict of policy values from request

### Returns

True if request is authorized against given policy, False otherwise

### Raises

`oslo_policy.policy.PolicyNotRegistered` if supplied policy is not registered in `oslo_policy`

`ironic_inspector.policy.get_enforcer`()

Provides access to the single instance of Policy enforcer.

`ironic_inspector.policy.get_oslo_policy_enforcer()`

Get the enforcer instance to generate policy files.

This method is for use by oslopolicy CLI scripts. Those scripts need the output-file and namespace options, but having those in `sys.argv` means loading the inspector config options will fail as those are not expected to be present. So we pass in an arg list with those stripped out.

`ironic_inspector.policy.init_enforcer(policy_file=None, rules=None, default_rule=None, use_conf=True)`

Synchronously initializes the policy enforcer

#### Parameters

- **policy\_file** Custom policy file to use, if none is specified, `CONF.oslo_policy.policy_file` will be used.
- **rules** Default dictionary / Rules to use. It will be considered just in the first instantiation.
- **default\_rule** Default rule to use, `CONF.oslo_policy.policy_default_rule` will be used if none is specified.
- **use\_conf** Whether to load rules from config file.

`ironic_inspector.policy.list_policies()`

Get list of all policies defined in code.

Used to register them all at runtime, and by oslo-config-generator to generate sample policy files.

## ironic\_inspector.process module

Handling introspection data from the ramdisk.

`ironic_inspector.process.get_introspection_data(uuid, processed=True, get_json=False)`

Get introspection data from the storage backend.

#### Parameters

- **uuid** node UUID
- **processed** Indicates the type of introspection data to be read, set True to request processed introspection data.
- **get\_json** Specify whether return the introspection data in json format, string value is returned if False.

#### Raises

`utils.Error`

`ironic_inspector.process.process(introspection_data)`

Process data from the ramdisk.

This function heavily relies on the hooks to do the actual data processing.

`ironic_inspector.process.reapply(node_uuid, data=None)`

Re-apply introspection steps.

Re-apply preprocessing, postprocessing and introspection rules on stored data.

#### Parameters

- **node\_uuid** node UUID
- **data** unprocessed introspection data to be reapplied

**Raises**

utils.Error

`ironic_inspector.process.store_introspection_data(node_uuid, data, processed=True)`

Store introspection data to the storage backend.

**Parameters**

- **node\_uuid** node UUID
- **data** Introspection data to be saved
- **processed** The type of introspection data, set to True means the introspection data is processed, otherwise unprocessed.

**Raises**

utils.Error

### **ironic\_inspector.rules module**

Support for introspection rules.

**class** `ironic_inspector.rules.IntrospectionRule`(*uuid, conditions, actions, description, scope=None*)

Bases: object

High-level class representing an introspection rule.

**apply\_actions**(*node\_info, data=None*)

Run actions on a node.

**Parameters**

- **node\_info** NodeInfo instance
- **data** introspection data

**as\_dict**(*short=False*)

**check\_conditions**(*node\_info, data*)

Check if conditions are true for a given node.

**Parameters**

- **node\_info** a NodeInfo object
- **data** introspection data

**Returns**

True if conditions match, otherwise False

**check\_scope**(*node\_info*)

Check if nodes scope falls under rule.\_scope and rule is applicable

**Parameters**

**node\_info** a NodeInfo object

**Returns**

True if conditions match, otherwise False

**property description**

`ironic_inspector.rules.actions_schema()`

`ironic_inspector.rules.apply(node_info, data)`

Apply rules to a node.

`ironic_inspector.rules.conditions_schema()`

`ironic_inspector.rules.create(conditions_json, actions_json, uuid=None, description=None, scope=None)`

Create a new rule in database.

**Parameters**

- **conditions\_json** list of dicts with the following keys: \* op - operator \* field - JSON path to field to compare Other keys are stored as is.
- **actions\_json** list of dicts with the following keys: \* action - action type Other keys are stored as is.
- **uuid** rule UUID, will be generated if empty
- **description** human-readable rule description
- **scope** if scope on node and rule matches, rule applies; if its empty, rule applies to all nodes.

**Returns**

new IntrospectionRule object

**Raises**

`utils.Error` on failure

`ironic_inspector.rules.delete(uuid)`

Delete a rule by its UUID.

`ironic_inspector.rules.delete_all()`

Delete all rules.

`ironic_inspector.rules.get(uuid)`

Get a rule by its UUID.

`ironic_inspector.rules.get_all()`

List all rules.

**ironic\_inspector.utils module**

`class ironic_inspector.utils.DeferredBasicAuthMiddleware(app, auth_file)`

Bases: `object`

Middleware which sets X-Identity-Status header based on authentication

**exception** `ironic_inspector.utils.Error(msg, code=400, log_level='error', **kwargs)`

Bases: `Exception`

Inspector exception.

**exception** `ironic_inspector.utils.IntrospectionDataNotFound(msg, code=404, **kwargs)`

Bases: `NotFoundInCacheError`

Introspection data not found.

**exception** `ironic_inspector.utils.IntrospectionDataStoreDisabled(msg, code=400, log_level='error', **kwargs)`

Bases: `Error`

Introspection data store is disabled.

**exception** `ironic_inspector.utils.NoAvailableConductor(msg, **kwargs)`

Bases: `Error`

No available conductor in the service group.

**exception** `ironic_inspector.utils.NodeNotFoundInDBError(**kwargs)`

Bases: `Error`

The node was not found in the database.

**exception** `ironic_inspector.utils.NodeStateInvalidEvent(msg, code=400, log_level='error', **kwargs)`

Bases: `Error`

Invalid event attempted.

**exception** `ironic_inspector.utils.NodeStateRaceCondition(*args, **kwargs)`

Bases: `Error`

State mismatch between the DB and a node\_info.

**exception** `ironic_inspector.utils.NotFoundInCacheError(msg, code=404, **kwargs)`

Bases: `Error`

Exception when node was not found in cache during processing.

**class** `ironic_inspector.utils.ProcessingLoggerAdapter(logger, extra=None)`

Bases: `KeywordArgumentAdapter`

**process**(`msg, kwargs`)

Process the logging message and keyword arguments passed in to a logging call to insert contextual information. You can either manipulate the message itself, the keyword args or both. Return the message and kwargs modified (or not) to suit your needs.

Normally, you'll only need to override this one method in a `LoggerAdapter` subclass for your specific needs.

**exception** `ironic_inspector.utils.RuleNotFoundError(uuid, *args, **kwargs)`

Bases: `Error`

The requested rule was not found.

**exception** `ironic_inspector.utils.RuleUUIDExistError(uuid, *args, **kwargs)`

Bases: `Error`

Rule requested already exists in the database.

`ironic_inspector.utils.add_auth_middleware(app)`

Add authentication middleware to Flask application.

**Parameters**

**app** application.

`ironic_inspector.utils.add_basic_auth_middleware(app)`

Add HTTP Basic authentication middleware to Flask application.

**Parameters**

**app** application.

`ironic_inspector.utils.add_cors_middleware(app)`

Create a CORS wrapper

Attach ironic-inspector-specific defaults that must be included in all CORS responses.

**Parameters**

**app** application

`ironic_inspector.utils.add_healthcheck_middleware(app)`

Add healthcheck middleware

**Parameters**

**app** application

`ironic_inspector.utils.check_auth(request, rule=None, target=None)`

Check authentication on request.

**Parameters**

- **request** Flask request
- **rule** policy rule to check the request against
- **target** dict-like structure to check rule against

**Raises**

`utils.Error` if access is denied

`ironic_inspector.utils.execute(*cmd, use_standard_locale=False, log_stdout=True, **kwargs)`

Convenience wrapper around `oslos execute()` method.

Executes and logs results from a system command. See docs for `oslo_concurrency.processutils.execute` for usage.

**Parameters**

- **cmd** positional arguments to pass to `processutils.execute()`
- **use\_standard\_locale** Defaults to False. If set to True, execute command with standard locale added to environment variables.
- **log\_stdout** Defaults to True. If set to True, logs the output.

- **kwargs** keyword arguments to pass to `processutils.execute()`

**Returns**

(`stdout`, `stderr`) from process execution

**Raises**

`UnknownArgumentError` on receiving unknown arguments

**Raises**

`ProcessExecutionError`

**Raises**

`OSError`

`ironic_inspector.utils.executor()`

Return the current futures executor.

`ironic_inspector.utils.getProcessingLogger(name)`

`ironic_inspector.utils.get_inventory(data, node_info=None)`

Get and validate the hardware inventory from introspection data.

`ironic_inspector.utils.get_ipmi_address_from_data(introspection_data)`

`ironic_inspector.utils.get_ipmi_v6address_from_data(introspection_data)`

`ironic_inspector.utils.get_pxe_mac(introspection_data)`

`ironic_inspector.utils.get_route_source(dest, ignore_link_local=True)`

Get the IP address to send packages to destination.

`ironic_inspector.utils.get_valid_macs(data)`

Get a list of valid MACs from the introspection data.

`ironic_inspector.utils.iso_timestamp(timestamp=None, tz=datetime.timezone.utc)`

Return an ISO8601-formatted timestamp (tz: UTC) or None.

**Parameters**

- **timestamp** such as `time.time()` or `None`
- **tz** timezone

**Returns**

an ISO8601-formatted timestamp, or `None`

`ironic_inspector.utils.processing_logger_prefix(data=None, node_info=None)`

Calculate prefix for logging.

Tries to use: \* node UUID, `node._state` \* node PXE MAC, \* node BMC address

**Parameters**

- **data** introspection data
- **node\_info** `NodeInfo` or ironic node object

**Returns**

logging prefix as a string

`ironic_inspector.utils.unlink_without_raise(path)`



## ironic\_inspector.version module

## ironic\_inspector.wsgi\_service module

### class ironic\_inspector.wsgi\_service.WSGIService

Bases: Service

Provides ability to launch API from wsgi app.

#### reset()

Reset server greenpool size to default.

#### Returns

None

#### start()

Start serving this service using loaded configuration.

#### Returns

None

#### stop()

Stop serving this API.

#### Returns

None

#### wait()

Wait for the service to stop serving this API.

#### Returns

None

## Module contents

### 4.1.8 Ironic Inspector CI

Its important to understand the role of each job in the CI. To facilitate that, we have created the documentation below.

#### Jobs description

The description of each jobs that runs in the CI when you submit a patch for *openstack/ironic-inspector* is shown in the following table.

#### Note

All jobs are configured to use a pre-build tinyipa ramdisk, a wholedisk image that is downloaded from a Swift temporary url, *pxe* boot and *ipmi* driver.

Table 1: Table. OpenStack Ironic Inspector CI jobs description

Job name	Description
ironic-inspector-grenade	Deploys Ironic and Ironic Inspector in DevStack and runs upgrade for all enabled services.
ironic-inspector-tempest	Deploys Ironic and Ironic Inspector in DevStack. Runs tempest tests that match the regex <i>InspectorBasicTest</i> and deploys 1 virtual baremetal.
ironic-inspector-tempest-discovery	Deploys Ironic and Ironic Inspector in DevStack. Runs tempest tests that match the regex <i>InspectorDiscoveryTest</i> and deploys 1 virtual baremetal.
ironic-inspector-tempest-python3	Deploys Ironic and Ironic Inspector in DevStack under Python3. Runs tempest tests that match the regex <i>Inspector</i> and deploys 1 virtual baremetal.
openstack-tox-functional-py36	Run tox-based functional tests for Ironic Inspector under Python3.6
bifrost-integration-tinyipa-ubuntu-xenial	Tests the integration between Ironic Inspector and Bifrost.
ironic-inspector-tox-bandit	Runs bandit security tests in a tox environment to find known issues in the Ironic Inspector code.

## INDICES AND TABLES

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### i

- `ironic_inspector`, 181
- `ironic_inspector.api_tools`, 163
- `ironic_inspector.cmd`, 117
  - `all`, 116
  - `conductor`, 116
  - `dbsync`, 116
  - `migration`, 117
  - `status`, 117
  - `wsgi`, 117
- `ironic_inspector.common`, 131
  - `auth_basic`, 117
  - `context`, 119
  - `coordination`, 119
  - `device_hints`, 120
  - `exception`, 122
  - `ironic`, 123
  - `keystone`, 125
  - `lldp_parsers`, 125
  - `lldp_tlvs`, 127
  - `locking`, 127
  - `mdns`, 128
  - `rpc`, 129
  - `rpc_service`, 130
  - `service_utils`, 130
  - `swift`, 130
- `ironic_inspector.conductor`, 132
  - `manager`, 131
- `ironic_inspector.conf`, 134
  - `accelerators`, 132
  - `capabilities`, 132
  - `coordination`, 132
  - `default`, 132
  - `discovery`, 132
  - `dnsmasq_pxe_filter`, 133
  - `exception`, 133
  - `extra_hardware`, 133
  - `healthcheck`, 133
  - `iptables`, 133
  - `ironic`, 133
  - `mdns`, 133
  - `opts`, 133
  - `pci_devices`, 134
  - `port_physnet`, 134
  - `processing`, 134
  - `pxe_filter`, 134
  - `service_catalog`, 134
  - `swift`, 134
- `ironic_inspector.db`, 142
  - `api`, 134
  - `migration`, 140
  - `model`, 140
- `ironic_inspector.introspect`, 164
- `ironic_inspector.introspection_state`, 164
- `ironic_inspector.main`, 165
- `ironic_inspector.node_cache`, 167
- `ironic_inspector.plugins`, 158
  - `accel_device`, 142
  - `base`, 143
  - `base_physnet`, 145
  - `capabilities`, 146
  - `discovery`, 146
  - `example`, 147
  - `extra_hardware`, 148
  - `introspection_data`, 148
  - `lldp_basic`, 149
  - `local_link_connection`, 149

150  
ironic\_inspector.plugins.pci\_devices,  
150  
ironic\_inspector.plugins.physnet\_cidr\_map,  
150  
ironic\_inspector.plugins.raid\_device,  
151  
ironic\_inspector.plugins.rules, 152  
ironic\_inspector.plugins.standard, 157  
ironic\_inspector.policy, 174  
ironic\_inspector.process, 175  
ironic\_inspector.pxe\_filter, 163  
ironic\_inspector.pxe\_filter.base, 159  
ironic\_inspector.pxe\_filter.dnsmasq,  
161  
ironic\_inspector.pxe\_filter.interface,  
162  
ironic\_inspector.pxe\_filter.iptables,  
162  
ironic\_inspector.rules, 176  
ironic\_inspector.utils, 177  
ironic\_inspector.version, 181  
ironic\_inspector.wsgi\_service, 181

## INDEX

### A

- `abort` (*ironic\_inspector.introspection\_state.Events* attribute), 164
- `abort()` (in module *ironic\_inspector.introspect*), 164
- `abort_end` (*ironic\_inspector.introspection\_state.Events* attribute), 164
- `aborting` (*ironic\_inspector.introspection\_state.States* attribute), 165
- `AccelDevicesHook` (class in *ironic\_inspector.plugins.accel\_device*), 142
- `acquire()` (*ironic\_inspector.common.locking.BaseLock* method), 127
- `acquire()` (*ironic\_inspector.common.locking.InternalLock* method), 128
- `acquire()` (*ironic\_inspector.common.locking.ToozLock* method), 128
- `acquire_lock()` (*ironic\_inspector.node\_cache.NodeInfo* method), 167
- `action` (*ironic\_inspector.db.model.RuleAction* attribute), 142
- `actions` (*ironic\_inspector.db.model.Rule* attribute), 141
- `actions_schema()` (in module *ironic\_inspector.rules*), 177
- `active_macs()` (in module *ironic\_inspector.node\_cache*), 170
- `add_alembic_command()` (in module *ironic\_inspector.cmd.dbsync*), 116
- `add_attribute()` (*ironic\_inspector.node\_cache.NodeInfo* method), 167
- `add_auth_middleware()` (in module *ironic\_inspector.utils*), 179
- `add_auth_options()` (in module *ironic\_inspector.common.keystone*), 125
- `add_basic_auth_middleware()` (in module *ironic\_inspector.utils*), 179
- `add_capabilities()` (*ironic\_inspector.common.lldp\_parsers.LLDPBasicMgmt* method), 125
- `add_command_parsers()` (in module *ironic\_inspector.cmd.dbsync*), 116
- `add_cors_middleware()` (in module *ironic\_inspector.utils*), 179
- `add_dot1_link_aggregation()` (*ironic\_inspector.common.lldp\_parsers.LLDPParser* method), 126
- `add_dot1_port_protocol_vlan()` (*ironic\_inspector.common.lldp\_parsers.LLDPdot1Parser* method), 126
- `add_dot1_protocol_identities()` (*ironic\_inspector.common.lldp\_parsers.LLDPdot1Parser* method), 126
- `add_dot1_vlans()` (*ironic\_inspector.common.lldp\_parsers.LLDPdot1Parser* method), 127
- `add_dot3_macphy_config()` (*ironic\_inspector.common.lldp\_parsers.LLDPdot3Parser* method), 127
- `add_healthcheck_middleware()` (in module *ironic\_inspector.utils*), 179
- `add_mgmt_address()` (*ironic\_inspector.common.lldp\_parsers.LLDPBasicMgmt* method), 125
- `add_nested_value()` (*ironic\_inspector.common.lldp\_parsers.LLDPParser* method), 126
- `add_node()` (in module *ironic\_inspector.db.api*), 134
- `add_node()` (in module *ironic\_inspector.node\_cache*), 170
- `add_single_value()` (*ironic\_inspector.common.lldp\_parsers.LLDPParser* method), 126
- `add_trait()` (*ironic\_inspector.node\_cache.NodeInfo* method), 167
- `add_version_headers()` (in module *ironic\_inspector.main*), 165

**AddTraitAction** (class in `apply()` (`ironic_inspector.plugins.rules.SetCapabilityAction` method), 157  
     `ironic_inspector.plugins.rules`), 152

**aliases** (`ironic_inspector.plugins.pci_devices.PciDeviceActions()` attribute), 150

**all()** (`ironic_inspector.introspection_state.Events` class method), 164

**all()** (`ironic_inspector.introspection_state.States` class method), 165

**ALLOW\_NONE** (`ironic_inspector.plugins.base.RuleConditionPlugin` attribute), 144

**ALLOW\_NONE** (`ironic_inspector.plugins.rules.EmptyCondition` attribute), 152

**api()** (in module `ironic_inspector.main`), 165

**api\_continue()** (in module `ironic_inspector.main`), 166

**api\_introspection()** (in module `ironic_inspector.main`), 166

**api\_introspection\_abort()** (in module `ironic_inspector.main`), 166

**api\_introspection\_data()** (in module `ironic_inspector.main`), 166

**api\_introspection\_reapply()** (in module `ironic_inspector.main`), 166

**api\_introspection\_statuses()** (in module `ironic_inspector.main`), 166

**api\_introspection\_unprocessed\_data()** (in module `ironic_inspector.main`), 166

**api\_root()** (in module `ironic_inspector.main`), 166

**api\_rule()** (in module `ironic_inspector.main`), 166

**api\_rules()** (in module `ironic_inspector.main`), 166

**append\_value()** (`ironic_inspector.common.lldp_parsers.LLDPParser` method), 126

**apply()** (in module `ironic_inspector.rules`), 177

**apply()** (`ironic_inspector.plugins.base.RuleActionPlugin` method), 144

**apply()** (`ironic_inspector.plugins.example.ExampleRuleAction` method), 147

**apply()** (`ironic_inspector.plugins.rules.AddTraitAction` method), 152

**apply()** (`ironic_inspector.plugins.rules.ExtendAttributeAction` method), 153

**apply()** (`ironic_inspector.plugins.rules.FailAction` method), 153

**apply()** (`ironic_inspector.plugins.rules.RemoveTraitAction` method), 156

**apply()** (`ironic_inspector.plugins.rules.SetAttributeAction` method), 156

**apply()** (`ironic_inspector.plugins.rules.SetCapabilityAction` method), 157

**apply\_hook()** (`ironic_inspector.rules.IntrospectionRule` method), 176

**as\_dict()** (`ironic_inspector.db.model.RuleAction` method), 142

**as\_dict()** (`ironic_inspector.db.model.RuleCondition` method), 142

**as\_dict()** (`ironic_inspector.rules.IntrospectionRule` method), 176

**Attribute** (class in `ironic_inspector.db.model`), 140

**attributes** (`ironic_inspector.node_cache.NodeInfo` property), 167

**auth\_entry()** (in module `ironic_inspector.common.auth_basic`), 117

**authenticate()** (in module `ironic_inspector.common.auth_basic`), 118

**authorize()** (in module `ironic_inspector.policy`), 174

**B**

**BadRequest**, 122

**BaseFilter** (class in `ironic_inspector.pxe_filter.base`), 159

**BaseLock** (class in `ironic_inspector.common.locking`), 127

**BasePhysnetHook** (class in `ironic_inspector.plugins.base_physnet`), 145

**BaseStorageBackend** (class in `ironic_inspector.plugins.introspection_data`), 148

**BasicAuthMiddleware** (class in `ironic_inspector.common.auth_basic`), 117

**before\_processing()** (`ironic_inspector.plugins.base.ProcessingHook` method), 143

**before\_processing()** (`ironic_inspector.plugins.example.ExampleProcessingHook` method), 147

**before\_processing()** (`ironic_inspector.plugins.raid_device.RaidDeviceDetection` method), 151

**before\_processing()** (`ironic_inspector.plugins.standard.RamdiskErrorHook` method), 151



method), 158

before\_processing() (ironic\_inspector.plugins.standard.ValidateInterfacesHook), 158

before\_update() (ironic\_inspector.plugins.accel\_device.AccelDevicesHook), 142

before\_update() (ironic\_inspector.plugins.base.ProcessingHook), 143

before\_update() (ironic\_inspector.plugins.base\_physnet.BasePhysnetHook), 145

before\_update() (ironic\_inspector.plugins.capabilities.CapabilitiesHook), 146

before\_update() (ironic\_inspector.plugins.example.ExampleProcessingHook), 147

before\_update() (ironic\_inspector.plugins.extra\_hardware.ExtraHardwareHook), 148

before\_update() (ironic\_inspector.plugins.lldp\_basic.LLDPBasicProcessingHook), 149

before\_update() (ironic\_inspector.plugins.local\_link\_connection.GeneralLocalLinkConnectionHook), 150

before\_update() (ironic\_inspector.plugins.pci\_devices.PciDevicesHook), 150

before\_update() (ironic\_inspector.plugins.raid\_device.RaidDeviceDefinitionHook), 151

before\_update() (ironic\_inspector.plugins.standard.RootDiskSelectionHook), 158

before\_update() (ironic\_inspector.plugins.standard.ScheduledEventHook), 158

before\_update() (ironic\_inspector.plugins.standard.ValidateInterfacesHook), 167

bytes\_to\_int() (in module ironic\_inspector.common.lldp\_tlvs), 127

**C**

call\_with\_retries() (in module ironic\_inspector.common.ironic), 123

capabilities\_to\_dict() (in module ironic\_inspector.common.ironic), 123

CapabilitiesHook (class in ironic\_inspector.plugins.capabilities), 146

check() (ironic\_inspector.plugins.base.RuleConditionPluginHook), 144

check() (ironic\_inspector.plugins.rules.ContainsConditionHook), 152

check() (ironic\_inspector.plugins.rules.EmptyConditionHook), 153

check() (ironic\_inspector.plugins.rules.MatchesConditionHook), 154

check() (ironic\_inspector.plugins.rules.NetConditionHook), 155

check() (ironic\_inspector.plugins.rules.SimpleConditionHook), 157

check\_api\_version() (in module ironic\_inspector.main), 166

check\_auth() (in module ironic\_inspector.utils), 179

check\_hardware() (in module ironic\_inspector.rules.IntrospectionRuleHook), 176

check\_processing\_hook\_state() (in module ironic\_inspector.common.ironic), 123

check\_scope() (ironic\_inspector.rules.IntrospectionRuleHook), 176

Checks (class in ironic\_inspector.cmd.status), 117

clean\_up() (in module ironic\_inspector.node\_cache), 171

close() (ironic\_inspector.common.mdns.ZeroconfHook), 128

code (ironic\_inspector.common.exception.IronicException attribute), 123

code (ironic\_inspector.common.exception.UnauthorizedException attribute), 123

commit() (ironic\_inspector.node\_cache.NodeInfoHook), 167

conditions (ironic\_inspector.db.model.Rule attribute), 141

conditions\_schema() (in module ironic\_inspector.rules), 177

ConductorManager (class in ironic\_inspector.conductor.manager), 131

ConfigInvalid, 123

ContainsCondition (class in

- ironic\_inspector.plugins.rules*), 152
- `convert_exceptions()` (in module *ironic\_inspector.main*), 166
- `Coordinator` (class in *ironic\_inspector.common.coordination*), 119
- `create()` (in module *ironic\_inspector.rules*), 177
- `create_link_object()` (in module *ironic\_inspector.main*), 166
- `create_node()` (in module *ironic\_inspector.db.api*), 135
- `create_node()` (in module *ironic\_inspector.node\_cache*), 171
- `create_object()` (*ironic\_inspector.common.swift.SwiftAPI* method), 130
- `create_ports()` (*ironic\_inspector.node\_cache.NodeInfo* method), 167
- `create_rule()` (in module *ironic\_inspector.db.api*), 135
- `create_schema()` (in module *ironic\_inspector.db.migration*), 140
- `created_at` (*ironic\_inspector.db.model.Rule* attribute), 141
- ## D
- `data` (*ironic\_inspector.db.model.IntrospectionData* attribute), 141
- `DatabaseStore` (class in *ironic\_inspector.plugins.introspection\_data*), 148
- `DeferredBasicAuthMiddleware` (class in *ironic\_inspector.utils*), 177
- `del_host()` (*ironic\_inspector.conductor.manager.ConductorManager* method), 131
- `delete()` (in module *ironic\_inspector.rules*), 177
- `delete_all()` (in module *ironic\_inspector.rules*), 177
- `delete_all_rules()` (in module *ironic\_inspector.db.api*), 135
- `delete_attributes()` (in module *ironic\_inspector.db.api*), 135
- `delete_node()` (in module *ironic\_inspector.db.api*), 136
- `delete_nodes()` (in module *ironic\_inspector.db.api*), 136
- `delete_nodes_not_in_list()` (in module *ironic\_inspector.node\_cache*), 171
- `delete_options()` (in module *ironic\_inspector.db.api*), 136
- `delete_port()` (*ironic\_inspector.node\_cache.NodeInfo* method), 168
- `delete_rule()` (in module *ironic\_inspector.db.api*), 136
- `dependencies` (*ironic\_inspector.plugins.base.ProcessingHook* attribute), 143
- `description` (*ironic\_inspector.db.model.Rule* attribute), 141
- `description` (*ironic\_inspector.rules.IntrospectionRule* property), 177
- `dict_to_capabilities()` (in module *ironic\_inspector.common.ironic*), 124
- `disabled` (*ironic\_inspector.db.model.Rule* attribute), 141
- `DnsmasqFilter` (class in *ironic\_inspector.pxe\_filter.dnsmasq*), 161
- `do_abort()` (*ironic\_inspector.conductor.manager.ConductorManager* method), 131
- `do_alembic_command()` (in module *ironic\_inspector.cmd.dbsync*), 116
- `do_continue()` (*ironic\_inspector.conductor.manager.ConductorManager* method), 131
- `do_introspection()` (*ironic\_inspector.conductor.manager.ConductorManager* method), 131
- `do_reapply()` (*ironic\_inspector.conductor.manager.ConductorManager* method), 131
- `do_revision()` (in module *ironic\_inspector.cmd.dbsync*), 116
- `downgrade()` (in module *ironic\_inspector.db.migration*), 140
- `driver()` (in module *ironic\_inspector.pxe\_filter.base*), 160
- ## E
- `EmptyCondition` (class in *ironic\_inspector.plugins.rules*), 152
- `enroll_node_not_found_hook()` (in module *ironic\_inspector.plugins.discovery*), 146
- `enrolling` (*ironic\_inspector.introspection\_state.States* attribute), 165
- `EqCondition` (class in *ironic\_inspector.plugins.rules*), 153
- `Error`, 177
- `error` (*ironic\_inspector.db.model.Node* attribute), 141
- `error` (*ironic\_inspector.introspection\_state.Events* attribute), 165
- `error` (*ironic\_inspector.introspection\_state.States* attribute), 165

`error_response()` (in module `ironic_inspector.main`), 166

`Events` (class in `ironic_inspector.introspection_state`), 164

`Events` (class in `ironic_inspector.pxe_filter.base`), 160

`example_not_found_hook()` (in module `ironic_inspector.plugins.example`), 147

`ExampleProcessingHook` (class in `ironic_inspector.plugins.example`), 147

`ExampleRuleAction` (class in `ironic_inspector.plugins.example`), 147

`execute()` (in module `ironic_inspector.utils`), 179

`executor()` (in module `ironic_inspector.utils`), 180

`ExtendAttributeAction` (class in `ironic_inspector.plugins.rules`), 153

`ExtraHardwareHook` (class in `ironic_inspector.plugins.extra_hardware`), 148

## F

`FailAction` (class in `ironic_inspector.plugins.rules`), 153

`field` (`ironic_inspector.db.model.RuleCondition` attribute), 142

`FilterDriver` (class in `ironic_inspector.pxe_filter.interface`), 162

`find_devices_by_hints()` (in module `ironic_inspector.common.device_hints`), 120

`find_node()` (in module `ironic_inspector.node_cache`), 171

`finish()` (`ironic_inspector.introspection_state.Events` attribute), 165

`finished()` (`ironic_inspector.introspection_state.States` attribute), 165

`finished()` (`ironic_inspector.node_cache.NodeInfo` method), 168

`finished_at` (`ironic_inspector.db.model.Node` attribute), 141

`format_exception()` (`ironic_inspector.common.auth_basic.BasicAuthMiddleware` method), 117

`FORMATTED_PARAMS` (`ironic_inspector.plugins.base.RuleActionPlugin` attribute), 143

`FORMATTED_PARAMS` (`ironic_inspector.plugins.rules.ExtendAttributeAction` attribute), 153

`FORMATTED_PARAMS` (`ironic_inspector.plugins.rules.SetAttributeAction` attribute), 156

`FORMATTED_PARAMS` (`ironic_inspector.plugins.rules.SetCapabilityAction` attribute), 157

`from_dict()` (`ironic_inspector.common.context.RequestContext` class method), 119

`from_environ()` (`ironic_inspector.common.context.RequestContext` class method), 119

`from_row()` (`ironic_inspector.node_cache.NodeInfo` class method), 168

`fsm` (`ironic_inspector.pxe_filter.base.BaseFilter` attribute), 159

`fsm_event()` (`ironic_inspector.node_cache.NodeInfo` method), 168

`fsm_event_after()` (in module `ironic_inspector.node_cache`), 171

`fsm_event_before()` (in module `ironic_inspector.node_cache`), 172

`fsm_reset_on_error()` (`ironic_inspector.pxe_filter.base.BaseFilter` method), 159

`fsm_transition()` (in module `ironic_inspector.node_cache`), 172

## G

`GeCondition` (class in `ironic_inspector.plugins.rules`), 154

`generate_introspection_status()` (in module `ironic_inspector.main`), 166

`generate_resource_data()` (in module `ironic_inspector.main`), 166

`GenericLocalLinkConnectionHook` (class in `ironic_inspector.plugins.local_link_connection`), 150

`get()` (in module `ironic_inspector.rules`), 177

`get()` (`ironic_inspector.plugins.introspection_data.BaseStorageBackend` method), 148

`get()` (`ironic_inspector.plugins.introspection_data.DatabaseStore` method), 149

`get()` (`ironic_inspector.plugins.introspection_data.NoStore` method), 149

`get()` (`ironic_inspector.plugins.introspection_data.SwiftStore` method), 149

`get_active_macs()` (in module `ironic_inspector.pxe_filter.base`), 161

`get_active_nodes()` (in module `ironic_inspector.db.api`), 136

`get_adapter()` (in module `ironic_inspector.common.keystone`), 125

`get_all()` (in module `ironic_inspector.rules`), 177

`get_app()` (in module `ironic_inspector.main`), 166

`get_attributes()` (in module `ironic_inspector.db.api`), 136

`get_autoneg_cap()` (in module `ironic_inspector.common.lldp_tlvs`), 127

`get_by_path()` (`ironic_inspector.node_cache.NodeInfo` method), 168

`get_client()` (in module `ironic_inspector.common.ironic`), 124

`get_client()` (in module `ironic_inspector.common.rpc`), 129

`get_client_compat()` (in module `ironic_inspector.main`), 166

`get_coordinator()` (in module `ironic_inspector.common.coordination`), 120

`get_endpoint()` (in module `ironic_inspector.common.keystone`), 125

`get_endpoint()` (`ironic_inspector.common.mdns.Zeroconf` method), 129

`get_enforcer()` (in module `ironic_inspector.policy`), 174

`get_inactive_macs()` (in module `ironic_inspector.pxe_filter.base`), 161

`get_introspection_data()` (in module `ironic_inspector.common.swift`), 131

`get_introspection_data()` (in module `ironic_inspector.db.api`), 137

`get_introspection_data()` (in module `ironic_inspector.node_cache`), 172

`get_introspection_data()` (in module `ironic_inspector.process`), 175

`get_inventory()` (in module `ironic_inspector.utils`), 180

`get_ipmi_address()` (in module `ironic_inspector.common.ironic`), 124

`get_ipmi_address_from_data()` (in module `ironic_inspector.utils`), 180

`get_ipmi_v6address_from_data()` (in module `ironic_inspector.utils`), 180

`get_ironic_macs()` (in module `ironic_inspector.pxe_filter.base`), 161

`get_lock()` (in module `ironic_inspector.common.locking`), 128

`get_lock()` (`ironic_inspector.common.coordination.Coordinator` method), 119

`get_members()` (`ironic_inspector.common.coordination.Coordinator` method), 119

`get_node()` (in module `ironic_inspector.common.ironic`), 124

`get_node()` (in module `ironic_inspector.db.api`), 137

`get_node()` (in module `ironic_inspector.node_cache`), 172

`get_node_list()` (in module `ironic_inspector.node_cache`), 172

`get_nodes()` (in module `ironic_inspector.db.api`), 137

`get_object()` (`ironic_inspector.common.swift.SwiftAPI` method), 130

`get_options()` (in module `ironic_inspector.db.api`), 137

`get_oslo_policy_enforcer()` (in module `ironic_inspector.policy`), 174

`get_periodic_sync_task()` (`ironic_inspector.pxe_filter.base.BaseFilter` method), 159

`get_periodic_sync_task()` (`ironic_inspector.pxe_filter.base.NoopFilter` method), 160

`get_periodic_sync_task()` (`ironic_inspector.pxe_filter.interface.FilterDriver` method), 162

`get_physnet()` (`ironic_inspector.plugins.base_physnet.BasePhysnet` method), 146

`get_physnet()` (`ironic_inspector.plugins.physnet_cidr_map.PhysnetCIDRMap` method), 151

`get_pxe_mac()` (in module `ironic_inspector.utils`), 180

`get_random_topic()` (in module `ironic_inspector.main`), 166

`get_route_source()` (in module `ironic_inspector.utils`), 180

`get_rule()` (in module `ironic_inspector.db.api`), 137

`get_rules()` (in module `ironic_inspector.db.api`), 137

`get_rules_actions()` (in module `ironic_inspector.db.api`), 137

`get_rules_conditions()` (in module

*ironic\_inspector.db.api*), 138

get\_server() (in module *ironic\_inspector.common.rpc*), 129

get\_session() (in module *ironic\_inspector.common.keystone*), 125

get\_valid\_macs() (in module *ironic\_inspector.utils*), 180

get\_writer\_session() (in module *ironic\_inspector.db.api*), 138

getProcessingLogger() (in module *ironic\_inspector.utils*), 180

group\_name(*ironic\_inspector.common.coordination.Coordination* attribute), 120

GtCondition (class in *ironic\_inspector.plugins.rules*), 154

## H

handle\_404() (in module *ironic\_inspector.main*), 166

handle\_org\_specific\_tlv() (in module *ironic\_inspector.common.lldp\_parsers.LLDPParser* method), 125

headers(*ironic\_inspector.common.exception.IronicException* attribute), 123

headers(*ironic\_inspector.common.exception.Unauthorized* attribute), 123

## I

id (*ironic\_inspector.db.model.RuleAction* attribute), 142

id (*ironic\_inspector.db.model.RuleCondition* attribute), 142

init() (in module *ironic\_inspector.common.rpc*), 129

init() (in module *ironic\_inspector.db.api*), 138

init\_enforcer() (in module *ironic\_inspector.policy*), 175

init\_filter() (*ironic\_inspector.pxe\_filter.base.BaseFilter* method), 159

init\_filter() (*ironic\_inspector.pxe\_filter.dnsmasq.DnsmasqFilter* method), 161

init\_filter() (*ironic\_inspector.pxe\_filter.interface.InterfaceFilterDriver* method), 162

init\_filter() (*ironic\_inspector.pxe\_filter.iptables.IptablesFilter* method), 162

init\_host() (*ironic\_inspector.conductor.manager.ConductorManager* method), 131

initialize(*ironic\_inspector.pxe\_filter.base.Events* attribute), 160

initialize\_wsgi\_app() (in module *ironic\_inspector.cmd.wsgi*), 117

initialized(*ironic\_inspector.pxe\_filter.base.States* attribute), 160

InternalLock (class in *ironic\_inspector.common.locking*), 128

introspect() (in module *ironic\_inspector.introspect*), 164

introspection\_active() (in module *ironic\_inspector.node\_cache*), 173

introspection\_data\_manager() (in module *ironic\_inspector.plugins.base*), 145

IntrospectionData (class in *ironic\_inspector.db.model*), 141

IntrospectionDataNotFound, 178

IntrospectionDataStoreDisabled, 178

IntrospectionRule (class in *ironic\_inspector.rules*), 176

invalidate\_cache() (*ironic\_inspector.node\_cache.NodeInfo* method), 168

InvalidFilterDriverState, 160

InvalidMirrorParser(*ironic\_inspector.db.model.RuleCondition* attribute), 142

IptablesFilter (class in *ironic\_inspector.pxe\_filter.iptables*), 162

ironic (*ironic\_inspector.node\_cache.NodeInfo* property), 168

ironic\_inspector module, 181

ironic\_inspector.api\_tools module, 163

ironic\_inspector.cmd module, 117

ironic\_inspector.cmd.all module, 116

ironic\_inspector.cmd.conductor module, 116

ironic\_inspector.cmd.dbsync module, 116

ironic\_inspector.cmd.migration module, 117

ironic\_inspector.cmd.status module, 117

ironic\_inspector.cmd.wsgi module, 117

ironic\_inspector.common module, 131

ironic\_inspector.common.auth\_basic module, 117

ironic\_inspector.common.context



- module, 119
- `ironic_inspector.common.coordination`
  - module, 119
- `ironic_inspector.common.device_hints`
  - module, 120
- `ironic_inspector.common.exception`
  - module, 122
- `ironic_inspector.common.ironic`
  - module, 123
- `ironic_inspector.common.keystone`
  - module, 125
- `ironic_inspector.common.lldp_parsers`
  - module, 125
- `ironic_inspector.common.lldp_tlvs`
  - module, 127
- `ironic_inspector.common.locking`
  - module, 127
- `ironic_inspector.common.mdns`
  - module, 128
- `ironic_inspector.common.rpc`
  - module, 129
- `ironic_inspector.common.rpc_service`
  - module, 130
- `ironic_inspector.common.service_utils`
  - module, 130
- `ironic_inspector.common.swift`
  - module, 130
- `ironic_inspector.conductor`
  - module, 132
- `ironic_inspector.conductor.manager`
  - module, 131
- `ironic_inspector.conf`
  - module, 134
- `ironic_inspector.conf.accelerators`
  - module, 132
- `ironic_inspector.conf.capabilities`
  - module, 132
- `ironic_inspector.conf.coordination`
  - module, 132
- `ironic_inspector.conf.default`
  - module, 132
- `ironic_inspector.conf.discovery`
  - module, 132
- `ironic_inspector.conf.dnsmasq_pxe_filter`
  - module, 133
- `ironic_inspector.conf.exception`
  - module, 133
- `ironic_inspector.conf.extra_hardware`
  - module, 133
- `ironic_inspector.conf.healthcheck`
  - module, 133
- `ironic_inspector.conf.iptables`
  - module, 133
- `ironic_inspector.conf.ironic`
  - module, 133
- `ironic_inspector.conf.mdns`
  - module, 133
- `ironic_inspector.conf.opts`
  - module, 133
- `ironic_inspector.conf.pci_devices`
  - module, 134
- `ironic_inspector.conf.port_physnet`
  - module, 134
- `ironic_inspector.conf.processing`
  - module, 134
- `ironic_inspector.conf.pxe_filter`
  - module, 134
- `ironic_inspector.conf.service_catalog`
  - module, 134
- `ironic_inspector.conf.swift`
  - module, 134
- `ironic_inspector.db`
  - module, 142
- `ironic_inspector.db.api`
  - module, 134
- `ironic_inspector.db.migration`
  - module, 140
- `ironic_inspector.db.model`
  - module, 140
- `ironic_inspector.introspect`
  - module, 164
- `ironic_inspector.introspection_state`
  - module, 164
- `ironic_inspector.main`
  - module, 165
- `ironic_inspector.node_cache`
  - module, 167
- `ironic_inspector.plugins`
  - module, 158
- `ironic_inspector.plugins.accel_device`
  - module, 142
- `ironic_inspector.plugins.base`
  - module, 143
- `ironic_inspector.plugins.base_physnet`
  - module, 145
- `ironic_inspector.plugins.capabilities`
  - module, 146
- `ironic_inspector.plugins.discovery`
  - module, 146
- `ironic_inspector.plugins.example`
  - module, 147
- `ironic_inspector.plugins.extra_hardware`

module, 148  
 ironic\_inspector.plugins.introspection\_data  
   module, 148  
 ironic\_inspector.plugins.lldp\_basic  
   module, 149  
 ironic\_inspector.plugins.local\_link\_connection  
   module, 150  
 ironic\_inspector.plugins.pci\_devices  
   module, 150  
 ironic\_inspector.plugins.physnet\_cidr\_map  
   module, 150  
 ironic\_inspector.plugins.raid\_device  
   module, 151  
 ironic\_inspector.plugins.rules  
   module, 152  
 ironic\_inspector.plugins.standard  
   module, 157  
 ironic\_inspector.policy  
   module, 174  
 ironic\_inspector.process  
   module, 175  
 ironic\_inspector.pxe\_filter  
   module, 163  
 ironic\_inspector.pxe\_filter.base  
   module, 159  
 ironic\_inspector.pxe\_filter.dnsmasq  
   module, 161  
 ironic\_inspector.pxe\_filter.interface  
   module, 162  
 ironic\_inspector.pxe\_filter.iptables  
   module, 162  
 ironic\_inspector.rules  
   module, 176  
 ironic\_inspector.utils  
   module, 177  
 ironic\_inspector.version  
   module, 181  
 ironic\_inspector.wsgi\_service  
   module, 181  
 IronicException, 123  
 is\_locked() (*ironic\_inspector.common.locking.BaseLock*  
   *method*), 128  
 is\_locked() (*ironic\_inspector.common.locking.InternalLock*  
   *method*), 128  
 is\_locked() (*ironic\_inspector.common.locking.ToozLock*  
   *method*), 128  
 iso\_timestamp() (in module *ironic\_inspector.utils*), 180  
  
**J**  
 join\_group() (*ironic\_inspector.common.coordination.Coordinator*  
   *method*), 120  
  
**K**  
 KEYS (*ironic\_inspector.plugins.standard.SchedulerHook*  
   *attribute*), 158  
  
**L**  
 leave\_group() (*ironic\_inspector.common.coordination.Coordinator*  
   *method*), 120  
  
**U**  
 UnreadableCondition (class in  
   *ironic\_inspector.plugins.rules*), 154  
 limit\_field() (in module  
   *ironic\_inspector.api\_tools*), 163  
 list\_nodes\_by\_attributes() (in module  
   *ironic\_inspector.db.api*), 138  
 list\_nodes\_options\_by\_uuid() (in module  
   *ironic\_inspector.db.api*), 138  
 list\_opts() (in module  
   *ironic\_inspector.conf.accelerators*),  
   132  
 list\_opts() (in module  
   *ironic\_inspector.conf.capabilities*),  
   132  
 list\_opts() (in module  
   *ironic\_inspector.conf.coordination*),  
   132  
 list\_opts() (in module  
   *ironic\_inspector.conf.default*), 132  
 list\_opts() (in module  
   *ironic\_inspector.conf.discovery*), 132  
 list\_opts() (in module  
   *ironic\_inspector.conf.dnsmasq\_pxe\_filter*),  
   133  
 list\_opts() (in module  
   *ironic\_inspector.conf.extra\_hardware*),  
   133  
 list\_opts() (in module  
   *ironic\_inspector.conf.healthcheck*),  
   133  
 list\_opts() (in module  
   *ironic\_inspector.conf.iptables*), 133  
 list\_opts() (in module  
   *ironic\_inspector.conf.ironic*), 133  
 list\_opts() (in module  
   *ironic\_inspector.conf.opts*), 133  
 list\_opts() (in module  
   *ironic\_inspector.conf.pci\_devices*),  
   134  
 list\_opts() (in module  
   *ironic\_inspector.conf.port\_physnet*),  
   134

list\_opts() (in module *ironic\_inspector.conf.processing*), 134

list\_opts() (in module *ironic\_inspector.conf.pxe\_filter*), 134

list\_opts() (in module *ironic\_inspector.conf.service\_catalog*), 134

list\_opts() (in module *ironic\_inspector.conf.swift*), 134

list\_policies() (in module *ironic\_inspector.policy*), 175

LLDPBasicMgmtParser (class in *ironic\_inspector.common.lldp\_parsers*), 125

LLDPBasicProcessingHook (class in *ironic\_inspector.plugins.lldp\_basic*), 149

LLDPdot1Parser (class in *ironic\_inspector.common.lldp\_parsers*), 126

LLDPdot3Parser (class in *ironic\_inspector.common.lldp\_parsers*), 127

LLDPParser (class in *ironic\_inspector.common.lldp\_parsers*), 125

lock\_prefix(*ironic\_inspector.common.coordination.Coordinator* attribute), 120

locked\_driver\_event() (in module *ironic\_inspector.pxe\_filter.base*), 161

lookup\_node() (in module *ironic\_inspector.common.ironic*), 124

lookup\_node\_by\_bmc\_addresses() (in module *ironic\_inspector.common.ironic*), 124

lookup\_node\_by\_mac() (in module *ironic\_inspector.common.ironic*), 124

LtCondition (class in *ironic\_inspector.plugins.rules*), 154

**M**

main() (in module *ironic\_inspector.cmd.all*), 116

main() (in module *ironic\_inspector.cmd.conductor*), 116

main() (in module *ironic\_inspector.cmd.dbsync*), 116

main() (in module *ironic\_inspector.cmd.migration*), 117

main() (in module *ironic\_inspector.cmd.status*), 117

main() (*ironic\_inspector.cmd.migration.MigrationTool* method), 117

manage\_boot (*ironic\_inspector.db.model.Node* attribute), 141

manage\_boot (*ironic\_inspector.node\_cache.NodeInfo* property), 169

mapping\_for\_enum() (in module *ironic\_inspector.common.lldp\_tlvs*), 127

mapping\_for\_switch() (in module *ironic\_inspector.common.lldp\_tlvs*), 127

marker\_field() (in module *ironic\_inspector.api\_tools*), 163

match\_root\_device\_hints() (in module *ironic\_inspector.common.device\_hints*), 121

MatchesCondition (class in *ironic\_inspector.plugins.rules*), 154

MigrationTool (class in *ironic\_inspector.cmd.migration*), 117

missing\_entrypoints\_callback() (in module *ironic\_inspector.plugins.base*), 145

model\_query() (in module *ironic\_inspector.db.api*), 138

ModelBase (class in *ironic\_inspector.db.model*), 141

module

*ironic\_inspector*, 181

*ironic\_inspector.api\_tools*, 163

*ironic\_inspector.cmd*, 117

*ironic\_inspector.cmd.all*, 116

*ironic\_inspector.cmd.conductor*, 116

*ironic\_inspector.cmd.dbsync*, 116

*ironic\_inspector.cmd.migration*, 117

*ironic\_inspector.cmd.status*, 117

*ironic\_inspector.cmd.wsgi*, 117

*ironic\_inspector.common*, 131

*ironic\_inspector.common.auth\_basic*, 117

*ironic\_inspector.common.context*, 119

*ironic\_inspector.common.coordination*, 119

*ironic\_inspector.common.device\_hints*, 120

*ironic\_inspector.common.exception*, 122

*ironic\_inspector.common.ironic*, 123

*ironic\_inspector.common.keystone*, 125

*ironic\_inspector.common.lldp\_parsers*, 125

*ironic\_inspector.common.lldp\_tlvs*,



- 127
  - `ironic_inspector.common.locking`, 127
  - `ironic_inspector.common.mdns`, 128
  - `ironic_inspector.common.rpc`, 129
  - `ironic_inspector.common.rpc_service`, 130
  - `ironic_inspector.common.service_utils`, 130
  - `ironic_inspector.common.swift`, 130
  - `ironic_inspector.conductor`, 132
  - `ironic_inspector.conductor.manager`, 131
  - `ironic_inspector.conf`, 134
  - `ironic_inspector.conf.accelerators`, 132
  - `ironic_inspector.conf.capabilities`, 132
  - `ironic_inspector.conf.coordination`, 132
  - `ironic_inspector.conf.default`, 132
  - `ironic_inspector.conf.discovery`, 132
  - `ironic_inspector.conf.dnsmasq_pxe_filter`, 133
  - `ironic_inspector.conf.exception`, 133
  - `ironic_inspector.conf.extra_hardware`, 133
  - `ironic_inspector.conf.healthcheck`, 133
  - `ironic_inspector.conf.iptables`, 133
  - `ironic_inspector.conf.ironic`, 133
  - `ironic_inspector.conf.mdns`, 133
  - `ironic_inspector.conf.opts`, 133
  - `ironic_inspector.conf.pci_devices`, 134
  - `ironic_inspector.conf.port_physnet`, 134
  - `ironic_inspector.conf.processing`, 134
  - `ironic_inspector.conf.pxe_filter`, 134
  - `ironic_inspector.conf.service_catalog`, 134
  - `ironic_inspector.conf.swift`, 134
  - `ironic_inspector.db`, 142
  - `ironic_inspector.db.api`, 134
  - `ironic_inspector.db.migration`, 140
  - `ironic_inspector.db.model`, 140
  - `ironic_inspector.introspect`, 164
  - `ironic_inspector.introspection_state`, 164
  - `ironic_inspector.main`, 165
  - `ironic_inspector.node_cache`, 167
  - `ironic_inspector.plugins`, 158
  - `ironic_inspector.plugins.accel_device`, 142
  - `ironic_inspector.plugins.base`, 143
  - `ironic_inspector.plugins.base_physnet`, 145
  - `ironic_inspector.plugins.capabilities`, 146
  - `ironic_inspector.plugins.discovery`, 146
  - `ironic_inspector.plugins.example`, 147
  - `ironic_inspector.plugins.extra_hardware`, 148
  - `ironic_inspector.plugins.introspection_data`, 148
  - `ironic_inspector.plugins.lldp_basic`, 149
  - `ironic_inspector.plugins.local_link_connection`, 150
  - `ironic_inspector.plugins.pci_devices`, 150
  - `ironic_inspector.plugins.physnet_cidr_map`, 150
  - `ironic_inspector.plugins.raid_device`, 151
  - `ironic_inspector.plugins.rules`, 152
  - `ironic_inspector.plugins.standard`, 157
  - `ironic_inspector.policy`, 174
  - `ironic_inspector.process`, 175
  - `ironic_inspector.pxe_filter`, 163
  - `ironic_inspector.pxe_filter.base`, 159
  - `ironic_inspector.pxe_filter.dnsmasq`, 161
  - `ironic_inspector.pxe_filter.interface`, 162
  - `ironic_inspector.pxe_filter.iptables`, 162
  - `ironic_inspector.rules`, 176
  - `ironic_inspector.utils`, 177
  - `ironic_inspector.version`, 181
  - `ironic_inspector.wsgi_service`, 181
  - multiple (*ironic\_inspector.db.model.RuleCondition attribute*), 142
- N**
- name (*ironic\_inspector.db.model.Attribute attribute*), 140

name (*ironic\_inspector.db.model.Option* attribute), 141

NeCondition (class in *ironic\_inspector.plugins.rules*), 154

NetCondition (class in *ironic\_inspector.plugins.rules*), 155

NoAvailableConductor, 178

Node (class in *ironic\_inspector.db.model*), 141

node() (*ironic\_inspector.node\_cache.NodeInfo* method), 169

node\_not\_found\_hook\_manager() (in module *ironic\_inspector.plugins.base*), 145

node\_uuid (*ironic\_inspector.db.model.Attribute* attribute), 140

NodeInfo (class in *ironic\_inspector.node\_cache*), 167

NodeNotFoundInDBError, 178

NodeStateInvalidEvent, 178

NodeStateRaceCondition, 178

NoopFilter (class in *ironic\_inspector.pxe\_filter.base*), 160

NoStore (class in *ironic\_inspector.plugins.introspection\_data*), 149

NotFound, 123

NotFoundInCacheError, 178

**O**

Octal (class in *ironic\_inspector.conf.default*), 132

op (*ironic\_inspector.db.model.RuleCondition* attribute), 142

op (*ironic\_inspector.plugins.rules.SimpleCondition* attribute), 157

op() (*ironic\_inspector.plugins.rules.EqCondition* method), 153

op() (*ironic\_inspector.plugins.rules.GeCondition* method), 154

op() (*ironic\_inspector.plugins.rules.GtCondition* method), 154

op() (*ironic\_inspector.plugins.rules.LeCondition* method), 154

op() (*ironic\_inspector.plugins.rules.LtCondition* method), 154

op() (*ironic\_inspector.plugins.rules.NeCondition* method), 155

Option (class in *ironic\_inspector.db.model*), 141

OPTIONAL\_PARAMS (*ironic\_inspector.plugins.base.WithValidation* attribute), 144

OPTIONAL\_PARAMS (*ironic\_inspector.plugins.rules.ExtendAttributeAction* attribute), 153

OPTIONAL\_PARAMS (*ironic\_inspector.plugins.rules.SetAttributeAction* attribute), 156

OPTIONAL\_PARAMS (*ironic\_inspector.plugins.rules.SetCapabilityAction* attribute), 157

options (*ironic\_inspector.node\_cache.NodeInfo* property), 169

**P**

params (*ironic\_inspector.db.model.RuleAction* attribute), 142

params (*ironic\_inspector.db.model.RuleCondition* attribute), 142

parse\_args() (in module *ironic\_inspector.conf.opts*), 133

parse\_entry() (in module *ironic\_inspector.common.auth\_basic*), 118

parse\_header() (in module *ironic\_inspector.common.auth\_basic*), 118

parse\_root\_device\_hints() (in module *ironic\_inspector.common.device\_hints*), 122

parse\_tlv() (*ironic\_inspector.common.lldp\_parsers.LLDPParser* method), 126

parse\_token() (in module *ironic\_inspector.common.auth\_basic*), 118

patch() (*ironic\_inspector.node\_cache.NodeInfo* method), 169

patch\_port() (*ironic\_inspector.node\_cache.NodeInfo* method), 169

PciDevicesHook (class in *ironic\_inspector.plugins.pci\_devices*), 150

periodic\_clean\_up() (in module *ironic\_inspector.conductor.manager*), 132

periodic\_leader\_election() (in module *ironic\_inspector.conductor.manager*), 132

PhysnetCidrMapHook (class in *ironic\_inspector.plugins.physnet\_cidr\_map*), 150

ports() (*ironic\_inspector.node\_cache.NodeInfo* method), 169

prepare\_service() (in module *ironic\_inspector.common.service\_utils*), 150

130  
 process(*ironic\_inspector.introspection\_state.Events* attribute), 165  
 process() (in module *ironic\_inspector.process*), 175  
 process() (*ironic\_inspector.utils.ProcessingLoggerAdapter* method), 178  
 processed (*ironic\_inspector.db.model.IntrospectionData* attribute), 141  
 processing (*ironic\_inspector.introspection\_state.States* attribute), 165  
 processing\_hooks\_manager() (in module *ironic\_inspector.plugins.base*), 145  
 processing\_logger\_prefix() (in module *ironic\_inspector.utils*), 180  
 ProcessingHook (class in *ironic\_inspector.plugins.base*), 143  
 ProcessingLoggerAdapter (class in *ironic\_inspector.utils*), 178

## R

RaidDeviceDetection (class in *ironic\_inspector.plugins.raid\_device*), 151  
 raises\_coercion\_exceptions() (in module *ironic\_inspector.api\_tools*), 163  
 RamdiskErrorHook (class in *ironic\_inspector.plugins.standard*), 157  
 reapply (*ironic\_inspector.introspection\_state.Events* attribute), 165  
 reapply() (in module *ironic\_inspector.process*), 175  
 reapplying (*ironic\_inspector.introspection\_state.States* attribute), 165  
 ReCondition (class in *ironic\_inspector.plugins.rules*), 155  
 record\_node() (in module *ironic\_inspector.node\_cache*), 173  
 register\_auth\_opts() (in module *ironic\_inspector.common.keystone*), 125  
 register\_opts() (in module *ironic\_inspector.conf.accelerators*), 132  
 register\_opts() (in module *ironic\_inspector.conf.capabilities*), 132  
 register\_opts() (in module *ironic\_inspector.conf.coordination*), 132  
 register\_opts() (in module *ironic\_inspector.conf.default*), 132  
 register\_opts() (in module *ironic\_inspector.conf.discovery*), 132  
 register\_opts() (in module *ironic\_inspector.conf.dnsmasq\_pxe\_filter*), 133  
 register\_opts() (in module *ironic\_inspector.conf.exception*), 133  
 register\_opts() (in module *ironic\_inspector.conf.extra\_hardware*), 133  
 register\_opts() (in module *ironic\_inspector.conf.healthcheck*), 133  
 register\_opts() (in module *ironic\_inspector.conf iptables*), 133  
 register\_opts() (in module *ironic\_inspector.conf.ironic*), 133  
 register\_opts() (in module *ironic\_inspector.conf.mdns*), 133  
 register\_opts() (in module *ironic\_inspector.conf.pci\_devices*), 134  
 register\_opts() (in module *ironic\_inspector.conf.port\_physnet*), 134  
 register\_opts() (in module *ironic\_inspector.conf.processing*), 134  
 register\_opts() (in module *ironic\_inspector.conf.pxe\_filter*), 134  
 register\_opts() (in module *ironic\_inspector.conf.service\_catalog*), 134  
 register\_opts() (in module *ironic\_inspector.conf.swift*), 134  
 register\_service() (*ironic\_inspector.common.mdns.Zeroconf* method), 129  
 release() (*ironic\_inspector.common.locking.BaseLock* method), 128  
 release() (*ironic\_inspector.common.locking.InternalLock* method), 128  
 release() (*ironic\_inspector.common.locking.ToozLock* method), 128  
 release\_lock() (in module *ironic\_inspector.node\_cache*), 173  
 release\_lock() (*ironic\_inspector.node\_cache.NodeInfo* method), 169  
 remove\_trait()

(*ironic\_inspector.node\_cache.NodeInfo* method), 169

**RemoveTraitAction** (class in *ironic\_inspector.plugins.rules*), 155

**replace\_field()** (*ironic\_inspector.node\_cache.NodeInfo* method), 169

**request\_field()** (in module *ironic\_inspector.api\_tools*), 163

**RequestContext** (class in *ironic\_inspector.common.context*), 119

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.base.RuleConditionPlugin* attribute), 144

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.base.WithValidationRule* attribute), 144

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.AddTraitAction* attribute), 152

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.EmptyConditionRule* attribute), 152

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.ExtendAttributeAction* attribute), 153

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.FailAction* attribute), 153

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.RemoveTraitAction* attribute), 156

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.SetAttributeAction* attribute), 156

**REQUIRED\_PARAMS** (*ironic\_inspector.plugins.rules.SetCapabilityAction* attribute), 157

**reset** (*ironic\_inspector.pxe\_filter.base.Events* attribute), 160

**reset()** (in module *ironic\_inspector.plugins.base*), 145

**reset()** (*ironic\_inspector.pxe\_filter.base.BaseFilters* method), 159

**reset()** (*ironic\_inspector.pxe\_filter.dnsmasq.DnsmasqFilter* method), 161

**reset()** (*ironic\_inspector.pxe\_filter.iptables.IptablesFilter* method), 163

**reset()** (*ironic\_inspector.wsgi\_service.WSGIService* method), 181

**reset\_ironic\_session()** (in module *ironic\_inspector.common.ironic*), 125

**reset\_swift\_session()** (in module *ironic\_inspector.common.swift*), 131

**revision()** (in module *ironic\_inspector.db.migration*), 140

**RootDiskSelectionHook** (class in *ironic\_inspector.plugins.standard*), 158

**RPC\_API\_VERSION** (*ironic\_inspector.conductor.manager.ConductorManager* attribute), 131

**RPCService** (class in *ironic\_inspector.common.rpc\_service*), 130

**Rule** (class in *ironic\_inspector.db.model*), 141

**rule** (*ironic\_inspector.db.model.RuleAction* attribute), 142

**rule** (*ironic\_inspector.db.model.RuleCondition* attribute), 142

**rule\_actions\_manager()** (in module *ironic\_inspector.plugins.base*), 145

**rule\_conditions\_manager()** (in module *ironic\_inspector.plugins.base*), 145

**rule\_repr()** (in module *ironic\_inspector.main*), 166

**RuleAction** (class in *ironic\_inspector.db.model*), 141

**RuleActionPlugin** (class in *ironic\_inspector.plugins.base*), 143

**RuleCondition** (class in *ironic\_inspector.db.model*), 142

**RuleConditionPlugin** (class in *ironic\_inspector.plugins.base*), 144

**RuleNotFoundError**, 178

**RuleUUIDExistError**, 178

**run\_elect\_coordinator()** (*ironic\_inspector.common.coordination.Coordinator* method), 120

## S

**safe** (*ironic\_inspector.common.exception.IronicException* attribute), 123

**save()** (*ironic\_inspector.plugins.introspection\_data.BaseStorageBackend* method), 148

**save()** (*ironic\_inspector.plugins.introspection\_data.DatabaseStorageBackend* method), 149

**save()** (*ironic\_inspector.plugins.introspection\_data.NoStoreStorageBackend* method), 149

**save()** (*ironic\_inspector.plugins.introspection\_data.SwiftStorageBackend* method), 149

SchedulerHook (class in *ironic\_inspector.pxe\_filter.base.BaseFilter* property), 159

*ironic\_inspector.plugins.standard*), 158

scope (*ironic\_inspector.db.model.Rule* attribute), 141

ServiceLookupFailure, 123

ServiceRegistrationFailure, 123

session\_for\_read() (in module *ironic\_inspector.db.api*), 138

session\_for\_write() (in module *ironic\_inspector.db.api*), 138

set\_attribute() (in module *ironic\_inspector.db.api*), 139

set\_config\_defaults() (in module *ironic\_inspector.conf.opts*), 133

set\_cors\_middleware\_defaults() (in module *ironic\_inspector.conf.opts*), 133

set\_option() (in module *ironic\_inspector.db.api*), 139

set\_option() (*ironic\_inspector.node\_cache.NodeInfo* method), 170

set\_value() (*ironic\_inspector.common.lldp\_parser.LLDPParser* method), 126

SetAttributeAction (class in *ironic\_inspector.plugins.rules*), 156

SetCapabilityAction (class in *ironic\_inspector.plugins.rules*), 156

SimpleCondition (class in *ironic\_inspector.plugins.rules*), 157

stamp() (in module *ironic\_inspector.db.migration*), 140

start (*ironic\_inspector.introspection\_state.Events* attribute), 165

start() (*ironic\_inspector.common.coordination.Coordinator* method), 120

start() (*ironic\_inspector.common.rpc\_service.RPCService* method), 130

start() (*ironic\_inspector.wsgi\_service.WSGIService* method), 181

start\_coordinator() (in module *ironic\_inspector.main*), 166

start\_introspection() (in module *ironic\_inspector.node\_cache*), 173

started\_at (*ironic\_inspector.db.model.Node* attribute), 141

starting (*ironic\_inspector.introspection\_state.States* attribute), 165

state (*ironic\_inspector.db.model.Node* attribute), 141

state (*ironic\_inspector.node\_cache.NodeInfo* property), 170

state (*ironic\_inspector.pxe\_filter.base.BaseFilter* property), 159

state\_field() (in module *ironic\_inspector.api\_tools*), 164

States (class in *ironic\_inspector.introspection\_state*), 165

States (class in *ironic\_inspector.pxe\_filter.base*), 160

stop() (*ironic\_inspector.common.coordination.Coordinator* method), 120

stop() (*ironic\_inspector.common.rpc\_service.RPCService* method), 130

stop() (*ironic\_inspector.wsgi\_service.WSGIService* method), 181

store\_introspection\_data() (in module *ironic\_inspector.common.swift*), 131

store\_introspection\_data() (in module *ironic\_inspector.db.api*), 139

store\_introspection\_data() (in module *ironic\_inspector.node\_cache*), 173

store\_introspection\_data() (in module *ironic\_inspector.process*), 176

SwiftAPI (class in *ironic\_inspector.common.swift*), 130

SwiftStore (class in *ironic\_inspector.plugins.introspection\_data*), 149

sync (*ironic\_inspector.pxe\_filter.base.Events* attribute), 160

sync() (*ironic\_inspector.pxe\_filter.base.BaseFilter* method), 159

sync() (*ironic\_inspector.pxe\_filter.dnsmasq.DnsmasqFilter* method), 161

sync() (*ironic\_inspector.pxe\_filter.interface.FilterDriver* method), 163

sync() (*ironic\_inspector.pxe\_filter.iptables.IptablesFilter* method), 163

sync\_with\_ironic() (in module *ironic\_inspector.conductor.manager*), 132

## T

target (*ironic\_inspector.conductor.manager.ConductorManager* attribute), 132

tear\_down\_filter() (*ironic\_inspector.pxe\_filter.base.BaseFilter* method), 160

tear\_down\_filter() (*ironic\_inspector.pxe\_filter.interface.FilterDriver* method), 162



timeout (*ironic\_inspector.introspection\_state.Events* attribute), 165  
 to\_policy\_values() (*ironic\_inspector.common.context.RequestContext* method), 119  
 ToozLock (class in *ironic\_inspector.common.locking*), 128  
 triggers\_fsm\_error\_transition() (in module *ironic\_inspector.node\_cache*), 174  
**U**  
 Unauthorized, 123  
 unauthorized() (in module *ironic\_inspector.common.auth\_basic*), 119  
 uninitialized (*ironic\_inspector.pxe\_filter.base.States* attribute), 160  
 unlink\_without\_raise() (in module *ironic\_inspector.utils*), 180  
 update\_capabilities() (*ironic\_inspector.node\_cache.NodeInfo* method), 170  
 update\_node() (in module *ironic\_inspector.db.api*), 139  
 update\_properties() (*ironic\_inspector.node\_cache.NodeInfo* method), 170  
 upgrade() (in module *ironic\_inspector.db.migration*), 140  
 uuid (*ironic\_inspector.db.model.Attribute* attribute), 140  
 uuid (*ironic\_inspector.db.model.IntrospectionData* attribute), 141  
 uuid (*ironic\_inspector.db.model.Node* attribute), 141  
 uuid (*ironic\_inspector.db.model.Option* attribute), 141  
 uuid (*ironic\_inspector.db.model.Rule* attribute), 141  
**V**  
 validate() (*ironic\_inspector.plugins.base.WithValidation* method), 144  
 validate() (*ironic\_inspector.plugins.rules.NetCondition* method), 155  
 validate() (*ironic\_inspector.plugins.rules.ReCondition* method), 155  
 validate() (*ironic\_inspector.plugins.rules.SetAttributeAction* method), 156  
 validate\_auth\_file() (in module *ironic\_inspector.common.auth\_basic*),
 validate\_processing\_hooks() (in module *ironic\_inspector.plugins.base*), 145  
 ValidateInterfacesHook (class in *ironic\_inspector.plugins.standard*), 158  
 value (*ironic\_inspector.db.model.Attribute* attribute), 140  
 value (*ironic\_inspector.db.model.Option* attribute), 141  
 version() (in module *ironic\_inspector.db.migration*), 140  
 version\_id (*ironic\_inspector.db.model.Node* attribute), 141  
 version\_id (*ironic\_inspector.node\_cache.NodeInfo* property), 170  
 version\_root() (in module *ironic\_inspector.main*), 167  
**W**  
 wait (*ironic\_inspector.introspection\_state.Events* attribute), 165  
 wait() (*ironic\_inspector.wsgi\_service.WSGIService* method), 181  
 waiting (*ironic\_inspector.introspection\_state.States* attribute), 165  
 with\_revision() (in module *ironic\_inspector.cmd.dbsync*), 116  
 WithValidation (class in *ironic\_inspector.plugins.base*), 144  
 WSGIService (class in *ironic\_inspector.wsgi\_service*), 181  
**Z**  
 Zeroconf (class in *ironic\_inspector.common.mdns*), 128