1 Overview ........................................... 2
   1.1 Introduction ........................................... 2
       1.1.1 Background Story ................................... 2
       1.1.2 Definition Breakdown ................................ 2
   1.2 Cyborg architecture ................................. 3
   1.3 Usage .................................................. 4

2 Documentation for Operators .................. 5
   2.1 Installation ........................................... 5
       2.1.1 Compute service ..................................... 5
       2.1.2 Installing Cyborg API via WSGI .................. 16
   2.2 Configuration Reference ....................... 16
       2.2.1 Configuration Guide ................................ 16
       2.2.2 Cyborg Support Matrix ........................... 87
   2.3 Maintenance .......................................... 89
       2.3.1 Acceleration Service .............................. 89

3 For End Users ......................................... 95
   3.1 Tools for using Cyborg ......................... 95
       3.1.1 Command-Line Interface Reference .............. 95
   3.2 Using the API ......................................... 96

4 Documentation for Developers .............. 97
   4.1 Contributor Documentation ...................... 97
       4.1.1 Basic Information .................................. 97
       4.1.2 Reviewing ........................................ 98
   4.2 REST API Version History ...................... 109
       4.2.1 2.0 ............................................. 109
       4.2.2 2.1 (Maximum in Victoria) ..................... 109
       4.2.3 2.2 ............................................. 110

5 Indices and tables .................................. 111
Cyborg is a general management framework for accelerators
1.1 Introduction

1.1.1 Background Story

OpenStack Acceleration Discussion Started from Telco Requirements:

- High level requirements first drafted in the standard organization ETSI NFV ISG
- High level requirements transformed into detailed requirements in OPNFV DPACC project.
- New project called Nomad established to address the requirements.
- BoF discussions back in OpenStack Austin Summit.

Transition to Cyborg Project:

- From a long period of conversation and discussion within the OpenStack community, we found that the initial goal of Nomad project to address acceleration management in Telco is too limited. From design summit session in Barcelona Summit, we have developers from Scientific WG help us understanding the need for acceleration management in HPC cloud, and we also had a lot of discussion on the Public Cloud support of accelerated instances.
- We decide to formally establish a project that will work on the management framework for dedicated devices in OpenStack, and there comes the Cyborg Project.

1.1.2 Definition Breakdown

General Management Framework:
- Acceleration Resource Discovery
- Life Cycle Management

Accelerators:
- Software: dpdk/spdk, pmem,
- Hardware: FPGA, GPU, ARM SoC, NVMe SSD, CCIX based Caches,
1.2 Cyborg architecture

Cyborg design can be described by following diagram:

![Cyborg Architecture Diagram]

**cyborg-api** - cyborg-api is a cyborg service that provides REST API interface for the Cyborg project. It supports POST/PUT/DELETE/GET operations and interacts with cyborg-agent and cyborg-db via cyborg-conductor.

**cyborg-conductor** - cyborg-conductor is a cyborg service that coordinates interaction, DB access between cyborg-api and cyborg-agent.

**cyborg-agent** - cyborg-agent is a cyborg service that is responsible for interaction with accelerator backends via the Cyborg Driver. For now the only implementation in play is the Cyborg generic Driver. It will also handle the communication with the Nova placement service. Cyborg-Agent will also write to a local cache for local accelerator events.

**Vendor drivers** - Cyborg can be integrated with drivers for various accelerator device types, such as FPGA, GPU, NIC, and so forth. You are welcome to extend your own driver for a new type of accelerator device.
1.3 Usage

To use cyborg in a project:

```python
import cyborg
```
2.1 Installation

The detailed install guide for Cyborg.

2.1.1 Compute service

Common Configuration

Regardless of the package or code source you must do the following to properly setup the Accelerator Life Cycle Management service. A database, service credentials, and API endpoints must be created.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the `root` user:
     
     ```bash
     $ mysql -u root -p
     ```
   - Create the `cyborg` database:
     
     ```sql
     CREATE DATABASE cyborg;
     ```
   - Grant proper access to the `cyborg` database:
     
     ```sql
     GRANT ALL PRIVILEGES ON cyborg.* TO 'cyborg'@'localhost' IDENTIFIED BY 'CYBORG_DBPASS';
     ```
     Replace `CYBORG_DBPASS` with a suitable password.
   - Exit the database access client.
     
     ```bash
     exit;
     ```

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

   ```bash
   $ . admin-openrc
   ```

3. To create the service credentials, complete these steps:
• Create the cyborg user:

```bash
$ openstack user create --domain default --password-prompt cyborg
```

• Add the admin role to the cyborg user:

```bash
$ openstack role add --project service --user cyborg admin
```

• Create the cyborg service entities:

```bash
$ openstack service create --name cyborg --description "Acceleration\nService" accelerator
```

4. Create the Acceleration service API endpoints:

• If cyborg-api service is deployed using uwsgi, create the following endpoints:

```bash
$ openstack endpoint create --region RegionOne \accelerator public http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \accelerator internal http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \accelerator admin http://<cyborg-ip>/accelerator/v2
```

• Otherwise, if cyborg-api service is running as a python process, create the following endpoints:

```bash
$ openstack endpoint create --region RegionOne \accelerator public http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \accelerator internal http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \accelerator admin http://<cyborg-ip>:6666/v2
```

Note: URLs (publicurl, internalurl and adminurl) may be different depending on your environment.

Configure Cyborg

1. Edit cyborg.conf with your favorite editor. Below is an example which contains basic settings you likely need to configure.

```ini
[DEFAULT]
transport_url = rabbit://%RABBITMQ_USER%:%RABBITMQ_PASSWORD%@%OPENSTACK_\nHOST_IP%:5672/
use_syslog = False
state_path = /var/lib/cyborg
debug = True
...
```

(continues on next page)
[api]
host_ip = 0.0.0.0
...

[database]
connection = mysql+pymysql://%DATABASE_USER%:%DATABASE_PASSWORD%@%OPENSTACK_HOST_IP%/cyborg
...

[service_catalog]
cafile = /opt/stack/data/ca-bundle.pem
project_domain_id = default
user_domain_id = default
project_name = service
password = cyborg
username = cyborg
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password
...

[placement]
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = placement
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password
auth_section = keystone_authtoken
...

[nova]
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = nova
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password
auth_section = keystone_authtoken
...

[keystone_authtoken]
memcached_servers = localhost:11211
signing_dir = /var/cache/cyborg/api
cacertfile = /opt/stack/data/ca-bundle.pem
project_domain_name = Default
project_name = service
user_domain_name = Default
password = cyborg
username = cyborg
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password

2. Create database tables for Cyborg.

   cyborg-dbsync --config-file /etc/cyborg/cyborg.conf upgrade

3. Install Cyborg API via WSGI api-uwsgi

   Note: Cyborg-api service can also be run as a Python command that runs a web serve, which can be launched as follows with different Acceleration service API endpoints as mentioned in Prerequisites part. However, we would like to recommend you the uwsgi way since when a project provides a WSGI application the API service gains flexibility in terms of deployment, performance, configuration and scaling. BYW, if you choose devstack to deploy your acceleration service, uwsgi is a default choice.

   cyborg-api config-file=/etc/cyborg/cyborg.conf

1. Launch Cyborg Conductor, Cyborg Agent services. Open a separate terminal for each service since the console will be locked by a running process.

   cyborg-conductor --config-file=/etc/cyborg/cyborg.conf
   cyborg-agent --config-file=/etc/cyborg/cyborg.conf

### Install Cyborg from Source

This section describes how to install and configure the Acceleration Service for Ubuntu 18.04.1 LTS from source code.

#### Install from git repository

1. Create a folder which will hold all Cyborg components.

   mkdir ~/cyborg

2. Clone the cyborg git repository to the management server.

   cd ~/cyborg
   git clone https://opendev.org/openstack/cyborg

3. Set up the cyborg config file
First, generate a sample configuration file, using tox

```
cd ~/cyborg/cyborg
tox -e genconfig
```

And make a copy of it for further modifications

```
cp -r ~/cyborg/cyborg/etc/cyborg /etc
cd /etc/cyborg
ln -s cyborg.conf.sample cyborg.conf
```

4. Install Cyborg packages.

```
cd ~/cyborg/cyborg
sudo python setup.py install
```

## Common Configuration

Regardless of the package or code source you must do the following to properly setup the Accelerator Life Cycle Management service. A database, service credentials, and API endpoints must be created.

1. To create the database, complete these steps:
   - Use the database access client to connect to the database server as the root user:
     
     `$ mysql -u root -p`
   - Create the cyborg database:
     
     ```
     CREATE DATABASE cyborg;
     ```
   - Grant proper access to the cyborg database:
     
     ```
     GRANT ALL PRIVILEGES ON cyborg.* TO 'cyborg'@'localhost' IDENTIFIED BY 'CYBORG_DBPASS';
     ```
     
     Replace CYBORG_DBPASS with a suitable password.
   - Exit the database access client.
     
     ```
     exit:
     ```

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

   ```
   . admin-openrc
   ```

3. To create the service credentials, complete these steps:
   - Create the cyborg user:
     
     ```
     $ openstack user create --domain default --password-prompt cyborg
     ```
   - Add the admin role to the cyborg user:
Create the cyborg service entities:

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

4. Create the Acceleration service API endpoints:

   • If cyborg-api service is deployed using uwsgi, create the following endpoints:

```
$ openstack endpoint create --region RegionOne \  
  accelerator public http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \  
  accelerator internal http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \  
  accelerator admin http://<cyborg-ip>/accelerator/v2
```

   • Otherwise, if cyborg-api service is running as a python process, create the following endpoints:

```
$ openstack endpoint create --region RegionOne \  
  accelerator public http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \  
  accelerator internal http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \  
  accelerator admin http://<cyborg-ip>:6666/v2
```

**Note:** URLs (publicurl, internalurl and adminurl) may be different depending on your environment.

**Configure Cyborg**

1. Edit `cyborg.conf` with your favorite editor. Below is an example which contains basic settings you likely need to configure.

```
[DEFAULT]
transport_url = rabbit://%RABBITMQ_USER%:%RABBITMQ_PASSWORD%@%OPENSTACK_\  
  HOST_IP%:5672/
use_syslog = False
state_path = /var/lib/cyborg
debug = True
...

[api]
host_ip = 0.0.0.0
```

(continues on next page)
...  

[database]  
connection = mysql+pymysql://%DATABASE_USER%:%DATABASE_PASSWORD%@  
   → %OPENSTACK_HOST_IP%/cyborg

...

[service_catalog]  
cafile = /opt/stack/data/ca-bundle.pem  
project_domain_id = default  
user_domain_id = default  
project_name = service  
password = cyborg  
username = cyborg  
auth_url = http://%OPENSTACK_HOST_IP%/identity  
auth_type = password

...

[placement]  
project_domain_name = Default  
project_name = service  
user_domain_name = Default  
password = password  
username = placement  
auth_url = http://%OPENSTACK_HOST_IP%/identity  
auth_type = password  
auth_section = keystone_authtoken

...

[nova]  
project_domain_name = Default  
project_name = service  
user_domain_name = Default  
password = password  
username = nova  
auth_url = http://%OPENSTACK_HOST_IP%/identity  
auth_type = password  
auth_section = keystone_authtoken

...

[keystone_authtoken]  
memcached_servers = localhost:11211  
signing_dir = /var/cache/cyborg/api  
cafile = /opt/stack/data/ca-bundle.pem  
project_domain_name = Default

(continues on next page)
2. Create database tables for Cyborg.

```bash
cyborg-dbsync --config-file /etc/cyborg/cyborg.conf upgrade
```

3. Install Cyborg API via WSGI `api-uwsgi`

**Note:** Cyborg-api service can also be run as a Python command that runs a web serve, which can be launched as follows with different Acceleration service API endpoints as mentioned in Prerequisites part. However, we would like to recommend you the uwsgi way since when a project provides a WSGI application the API service gains flexibility in terms of deployment, performance, configuration and scaling. BYW, if you choose devstack to deploy your acceleration service, uwsgi is a default choice.

```bash
cyborg-api config-file=/etc/cyborg/cyborg.conf
```

1. Launch Cyborg Conductor, Cyborg Agent services. Open a separate terminal for each service since the console will be locked by a running process.

```bash
cyborg-conductor --config-file=/etc/cyborg/cyborg.conf
cyborg-agent --config-file=/etc/cyborg/cyborg.conf
```

**Installation with pip**

At the command line:

```bash
$ pip install openstack-cyborg
```

Or, if you have virtualenvwrapper installed:

```bash
$ mkvirtualenv cyborg
$ pip install openstack-cyborg
```

**Common Configuration**

Regardless of the package or code source you must do the following to properly setup the Accelerator Life Cycle Management service. A database, service credentials, and API endpoints must be created.

1. To create the database, complete these steps:

   - Use the database access client to connect to the database server as the `root` user:

     ```bash
     $ mysql -u root -p
     ```
• Create the cyborg database:

```sql
CREATE DATABASE cyborg;
```

• Grant proper access to the cyborg database:

```sql
GRANT ALL PRIVILEGES ON cyborg.* TO 'cyborg'@'localhost' IDENTIFIED BY 'CYBORG_DBPASS';
```

Replace CYBORG_DBPASS with a suitable password.

• Exit the database access client.

```sql
exit;
```

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

```
$. admin-openrc
```

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

• Create the cyborg user:

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

• Add the admin role to the cyborg user:

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

• Create the cyborg service entities:

```
$ openstack service create --name cyborg --description "Acceleration Service" accelerator
```

4. Create the Acceleration service API endpoints:

• If cyborg-api service is deployed using uwsgi, create the following endpoints:

```
$ openstack endpoint create --region RegionOne \
accelerator public http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \
accelerator internal http://<cyborg-ip>/accelerator/v2
$ openstack endpoint create --region RegionOne \
accelerator admin http://<cyborg-ip>/accelerator/v2
```

• Otherwise, if cyborg-api service is running as a python process, create the following endpoints:

```
$ openstack endpoint create --region RegionOne \
accelerator public http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \
accelerator internal http://<cyborg-ip>:6666/v2
$ openstack endpoint create --region RegionOne \
accelerator admin http://<cyborg-ip>:6666/v2
```
Configure Cyborg

1. Edit cyborg.conf with your favorite editor. Below is an example which contains basic settings you likely need to configure.

```ini
[DEFAULT]
transport_url = rabbit://%RABBITMQ_USER%:%RABBITMQ_PASSWORD%@%OPENSTACK_HOST_IP%:5672/
use_syslog = False
state_path = /var/lib/cyborg
debug = True
...

[api]
host_ip = 0.0.0.0
...

[database]
connection = mysql+pymysql://%DATABASE_USER%:%DATABASE_PASSWORD%@%OPENSTACK_HOST_IP%/cyborg
...

[service_catalog]
cafile = /opt/stack/data/ca-bundle.pem
project_domain_id = default
user_domain_id = default
project_name = service
password = cyborg
username = cyborg
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password
...

[placement]
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = placement
auth_url = http://%OPENSTACK_HOST_IP%/identity
auth_type = password
```

(continues on next page)
2. Create database tables for Cyborg.

```
cyborg-dbsync --config-file /etc/cyborg/cyborg.conf upgrade
```

3. Install Cyborg API via WSGI `api-uwsgi`

```
cyborg-api config-file=/etc/cyborg/cyborg.conf
```

**Note:** Cyborg-api service can also be run as a Python command that runs a web serve, which can be launched as follows with different Acceleration service API endpoints as mentioned in Prerequisites part. However, we would like to recommend you the uwsgi way since when a project provides a WSGI application the API service gains flexibility in terms of deployment, performance, configuration and scaling. BYW, if you choose devstack to deploy your acceleration service, uwsgi is a default choice.

```
cyborg-api config-file=/etc/cyborg/cyborg.conf
```

1. Launch Cyborg Conductor, Cyborg Agent services. Open a separate terminal for each service since the console will be locked by a running process.

```
cyborg-conductor --config-file=/etc/cyborg/cyborg.conf
cyborg-agent --config-file=/etc/cyborg/cyborg.conf
```
2.1.2 Installing Cyborg API via WSGI

Cyborg-api service can be run either as a Python command that runs a web serve or as a WSGI application hosted by uwsgi. This document is a guide to deploy cyborg-api using uwsgi. In devstack, uwsgi is used by default for development.

WSGI Application

The function cyborg.api.wsgi_app.init_application will setup a WSGI application to run behind uwsgi.

Cyborg API behind uwsgi

Create a cyborg-api-uwsgi file with content below:

```ini
[uwsgi]
chmod-socket = 666
socket = /var/run/uwsgi/cyborg-wsgi-api.socket
lazy-apps = true
add-header = Connection: close
buffer-size = 65535
hook-master-start = unix_signal:15 gracefully_kill_them_all
thunder-lock = true
plugins = python
enable-threads = true
worker-reload-mercy = 90
exit-on-reload = false
die-on-term = true
master = true
processes = 2
wsgi-file = /usr/local/bin/cyborg-wsgi-api
```

Start cyborg-api:

```
# uwsgi --ini /etc/cyborg/cyborg-api-uwsgi.ini
```

2.2 Configuration Reference

2.2.1 Configuration Guide

Configuration options for the Acceleration service

The following options can be set in the /etc/cyborg/cyborg.conf config file. A sample configuration file is also available.
Cyborg Documentation, Release 8.1.0.dev14

DEFAULT

**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.

**host**

  **Type** host address
  
  **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.

**periodic_interval**

  **Type** integer
  
  **Default** 60

  Default interval (in seconds) for running periodic tasks.

**thread_pool_size**

  **Type** integer
  
  **Default** 10

  This option specifies the size of the pool of threads used by API to do async jobs. It is possible to limit the number of concurrent connections using this option.

**bind_timeout**

  **Type** integer
  
  **Default** 60

  This option specifies the timeout of async job for ARQ bind.

**pybasedir**

  **Type** string
  
  **Default** /usr/lib/python/site-packages/cyborg/cyborg

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

  Directory where the cyborg python module is installed.

2.2. Configuration Reference 17
bindir

Type string
Default $pybasedir/bin
Directory where cyborg binaries are installed.

state_path

Type string
Default $pybasedir
Top-level directory for maintaining cyborgs state.

rpc_conn_pool_size

Type integer
Default 30
Minimum Value 1
Size of RPC connection pool.

Table 1: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>rpc_conn_pool_size</td>
</tr>
</tbody>
</table>

conn_pool_min_size

Type integer
Default 2
The pool size limit for connections expiration policy

cconn_pool_ttl

Type integer
Default 1200
The time-to-live in sec of idle connections in the pool

executor_thread_pool_size

Type integer
Default 64
Size of executor thread pool when executor is threading or eventlet.

Table 2: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>rpc_thread_pool_size</td>
</tr>
</tbody>
</table>
**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](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

**run_external_periodic_tasks**

- **Type**: boolean
- **Default**: True

Some periodic tasks can be run in a separate process. Should we run them here?

**backdoor_port**

- **Type**: string
- **Default**: <None>

Enable eventlet backdoor. Acceptable values are 0, <port>, and <start>:<end>, where 0 results in listening on a random tcp port number; <port> results in listening on the specified port number (and not enabling backdoor if that port is in use); and <start>:<end> results in listening on the smallest unused port number within the specified range of port numbers. The chosen port is displayed in the services log file.
backdoor_socket

    Type  string
    Default  <None>

Enable eventlet backdoor, using the provided path as a unix socket that can receive connections. This option is mutually exclusive with backdoor_port in that only one should be provided. If both are provided then the existence of this option overrides the usage of that option. Inside the path {pid} will be replaced with the PID of the current process.

log_options

    Type  boolean
    Default  True

Enables or disables logging values of all registered options when starting a service (at DEBUG level).

graceful_shutdown_timeout

    Type  integer
    Default  60

Specify a timeout after which a gracefully shutdown server will exit. Zero value means endless wait.

debug

    Type  boolean
    Default  False
    Mutable  This option can be changed without restarting.

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

log_config_append

    Type  string
    Default  <None>
    Mutable  This option can be changed without restarting.

The name of a logging configuration file. This file is appended to any existing logging configuration files. For details about logging configuration files, see the Python logging module documentation. Note that when logging configuration files are used then all logging configuration is set in the configuration file and other logging configuration options are ignored (for example, log-date-format).

Table 3: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>log-config</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>log_config</td>
</tr>
</tbody>
</table>

log_date_format

    Type  string
Default %Y-%m-%d %H:%M:%S

Defines the format string for %asctime)s in log records. Default: the value above. This option is ignored if log_config_append is set.

log_file

Type string

Default <None>

(Optional) Name of log file to send logging output to. If no default is set, logging will go to stderr as defined by use_stderr. This option is ignored if log_config_append is set.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>logfile</td>
</tr>
</tbody>
</table>

Table 4: Deprecated Variations

log_dir

Type string

Default <None>

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

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>logdir</td>
</tr>
</tbody>
</table>

Table 5: Deprecated Variations

watch_log_file

Type boolean

Default False

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

use_syslog

Type boolean

Default False

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

use_journal

Type boolean

Default False

Enable journald for logging. If running in a systemd environment you may wish to enable journal support. Doing so will use the journal native protocol which includes structured metadata in addition to log messages. This option is ignored if log_config_append is set.
syslog_log_facility
    Type  string
    Default LOG_USER
    Syslog facility to receive log lines. This option is ignored if log_config_append is set.

use_json
    Type  boolean
    Default False
    Use JSON formatting for logging. This option is ignored if log_config_append is set.

use_stderr
    Type  boolean
    Default False
    Log output to standard error. This option is ignored if log_config_append is set.

use_eventlog
    Type  boolean
    Default False
    Log output to Windows Event Log.

log_rotate_interval
    Type  integer
    Default 1
    The amount of time before the log files are rotated. This option is ignored unless log_rotation_type is set to interval.

log_rotate_interval_type
    Type  string
    Default days
    Valid Values  Seconds, Minutes, Hours, Days, Weekday, Midnight
    Rotation interval type. The time of the last file change (or the time when the service was started) is used when scheduling the next rotation.

max_logfile_count
    Type  integer
    Default 30
    Maximum number of rotated log files.

max_logfile_size_mb
    Type  integer
    Default 200
Log file maximum size in MB. This option is ignored if log_rotation_type is not set to size.

**log_rotation_type**

*Type* string  
*Default* none  
*Valid Values* interval, size, none  

Log rotation type.

**Possible values**

- **interval**  Rotate logs at predefined time intervals.
- **size**  Rotate logs once they reach a predefined size.
- **none**  Do not rotate log files.

**logging_context_format_string**

*Type* string  
*Default* 
```
%(asctime)s.%(msecs)03d %(process)d %(levelname)s %(name)s
%(global_request_id)s %(request_id)s %(user_identity)s
%(instance)s%(message)s
```

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

**logging_default_format_string**

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

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

**logging_debug_format_suffix**

*Type* string  
*Default* 
```
%(funcName)s %(pathname)s:%(lineno)d
```

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

**logging_exception_prefix**

*Type* string  
*Default* 
```
%(asctime)s.%(msecs)03d %(process)d ERROR %(name)s
%(instance)s
```

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

Type string

Default %\(\text{user}\)s %\(\text{project}\)s %\(\text{domain}\)s %\(\text{system\_scope}\)s
%\(\text{user\_domain}\)s %\(\text{project\_domain}\)s

Defines the format string for %\(\text{user\_identity}\)s that is used in logging\_context\_format\_string.
Used by oslo\_log\_formatters.ContextFormatter

default_log_levels

Type list

Default ['amqp=WARN', 'amqplib=WARN', 'boto=WARN', 'qpid=WARN',
'sqlalchemy=WARN', 'suds=INFO', 'oslo.messaging=INFO',
'oslo\_messaging=INFO', 'iso8601=WARN', 'requests.packages.
urllib3\_connectionpool=WARN', 'urllib3\_connectionpool=WARN',
'websocket=WARN', 'requests.packages.urllib3\_util\_retry=WARN',
'urllib3\_util\_retry=WARN', 'keystonemiddleware=WARN',
'routes\_middleware=WARN', 'stevedore=WARN', 'taskflow=WARN',
'keystoneauth=WARN', 'oslo\_cache=INFO', 'oslo\_policy=INFO',
'dogpile\_core\_dogpile=INFO']

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 "$[\text{instance: } %\{uuid\}\text{s}] ~$

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

instance_uuid_format

Type string

Default "$[\text{instance: } %\{uuid\}\text{s}] ~$

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

rate_limit_interval

Type integer

Default 0

Interval, number of seconds, of log rate limiting.

rate_limit_burst

Type integer
Default 0

Maximum number of logged messages per rate_limit_interval.

**rate_limit_except_level**

Type string

Default CRITICAL

Log level name used by rate limiting: CRITICAL, ERROR, INFO, WARNING, DEBUG or empty string. Logs with level greater or equal to rate_limit_except_level are not filtered. An empty string means that all levels are filtered.

**fatal_deprecations**

Type boolean

Default False

Enables or disables fatal status of deprecations.

**agent**

**enabled_drivers**

Type list

Default ['fake_driver']

The accelerator drivers enabled on this agent. Such as intel_fpga_driver, inspur_fpga_driver, nvidia_gpu_driver, intel_qat_driver, etc.

**api**

**host_ip**

Type host address

Default 127.0.0.1

The IP address on which cyborg-api listens.

**port**

Type port number

Default 6666

Minimum Value 0

Maximum Value 65535

The TCP port on which cyborg-api listens.

**api_workers**

Type integer

Default <None>
Number of workers for OpenStack Cyborg API service. The default is equal to the number of CPUs available if that can be determined, else a default worker count of 1 is returned.

**enable_ssl_api**

- **Type**: boolean
- **Default**: False

Enable the integrated stand-alone API to service requests via HTTPS instead of HTTP. If there is a front-end service performing HTTPS offloading from the service, this option should be False; note, you will want to change public API endpoint to represent SSL termination URL with public_endpoint option.

**public_endpoint**

- **Type**: string
- **Default**: <None>

Public URL to use when building the links to the API resources (for example, https://cyborg.rocks:6666). If None the links will be built using the requests host URL. If the API is operating behind a proxy, you will want to change this to represent the proxys URL. Defaults to None.

**api_paste_config**

- **Type**: string
- **Default**: api-paste.ini

Configuration file for WSGI definition of API.

**database**

**mysql_engine**

- **Type**: string
- **Default**: InnoDB

MySQL engine to use.

**sqlite_synchronous**

- **Type**: boolean
- **Default**: True

If True, SQLite uses synchronous mode.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sqlite_synchronous</td>
</tr>
</tbody>
</table>

**backend**

- **Type**: string
- **Default**: sqlalchemy
The back end to use for the database.

Table 7: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>db_backend</td>
</tr>
</tbody>
</table>

**connection**

_Type_ string

_Default_ <None>

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

Table 8: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_connection</td>
</tr>
<tr>
<td>DATABASE</td>
<td>sql_connection</td>
</tr>
<tr>
<td>sql</td>
<td>connection</td>
</tr>
</tbody>
</table>

**slave_connection**

_Type_ string

_Default_ <None>

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

**mysql_sql_mode**

_Type_ string

_Default_ TRADITIONAL

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

**mysql_enable_ndb**

_Type_ boolean

_Default_ False

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

**connection_recycle_time**

_Type_ integer

_Default_ 3600

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

**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.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_max_retries</td>
</tr>
<tr>
<td>DATABASE</td>
<td>sql_max_retries</td>
</tr>
</tbody>
</table>

**retry_interval**

Type integer

Default 10

Interval between retries of opening a SQL connection.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_retry_interval</td>
</tr>
<tr>
<td>DATABASE</td>
<td>reconnect_interval</td>
</tr>
</tbody>
</table>

**max_overflow**

Type integer

Default 50

If set, use this value for max_overflow with SQLAlchemy.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_max_overflow</td>
</tr>
<tr>
<td>DATABASE</td>
<td>sqlalchemy_max_overflow</td>
</tr>
</tbody>
</table>

**connection_debug**

Type integer

Default 0

Minimum Value 0

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

Table 12: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_connection_debug</td>
</tr>
</tbody>
</table>

connection_trace

Type  boolean
Default False

Add Python stack traces to SQL as comment strings.

Table 13: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>sql_connection_trace</td>
</tr>
</tbody>
</table>

pool_timeout

Type  integer
Default <None>

If set, use this value for pool_timeout with SQLAlchemy.

Table 14: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE</td>
<td>sqlalchemy_pool_timeout</td>
</tr>
</tbody>
</table>

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&

glance
Configuration options for the Image service

num_retries
  Type integer
  Default 0
  Minimum Value 0
  Enable glance operation retries.
  Specifies the number of retries when uploading / downloading an image to / from glance. 0 means no retries.

verify_glance_signatures
  Type boolean
  Default False
  Enable image signature verification.
  cyborg uses the image signature metadata from glance and verifies the signature of a signed image while downloading that image. If the image signature cannot be verified or if the image signature metadata is either incomplete or unavailable, then cyborg will not boot the image and instead will place the instance into an error state. This provides end users with stronger assurances of the integrity of the image data they are using to create servers.

  Related options:
  • The options in the key_manager group, as the key_manager is used for the signature validation.
• Both enable_certificate_validation and default_trusted_certificate_ids below depend on this option being enabled.

enable_certificate_validation

Type  boolean
Default  False

Enable certificate validation for image signature verification.

During image signature verification cyborg will first verify the validity of the images signing certificate using the set of trusted certificates associated with the instance. If certificate validation fails, signature verification will not be performed and the instance will be placed into an error state. This provides end users with stronger assurances that the image data is unmodified and trustworthy. If left disabled, image signature verification can still occur but the end user will not have any assurance that the signing certificate used to generate the image signature is still trustworthy.

Related options:

• This option only takes effect if verify_glance_signatures is enabled.

• The value of default_trusted_certificate_ids may be used when this option is enabled.

Warning:  This option is deprecated for removal since 16.0.0. Its value may be silently ignored in the future.

Reason  This option is intended to ease the transition for deployments leveraging image signature verification. The intended state long-term is for signature verification and certificate validation to always happen together.

default_trusted_certificate_ids

Type  list
Default  []

List of certificate IDs for certificates that should be trusted.

May be used as a default list of trusted certificate IDs for certificate validation. The value of this option will be ignored if the user provides a list of trusted certificate IDs with an instance API request. The value of this option will be persisted with the instance data if signature verification and certificate validation are enabled and if the user did not provide an alternative list. If left empty when certificate validation is enabled the user must provide a list of trusted certificate IDs otherwise certificate validation will fail.

Related options:

• The value of this option may be used if both verify_glance_signatures and enable_certificate_validation are enabled.

debug

Type  boolean
Default  False

Enable or disable debug logging with glanceclient.
cafile

    Type  string
    Default <None>

    PEM encoded Certificate Authority to use when verifying HTTPS connections.

certfile

    Type  string
    Default <None>

    PEM encoded client certificate cert file

keyfile

    Type  string
    Default <None>

    PEM encoded client certificate key file

insecure

    Type  boolean
    Default False

    Verify HTTPS connections.

timeout

    Type  integer
    Default <None>

    Timeout value for http requests

collect_timing

    Type  boolean
    Default False

    Collect per-API call timing information.

split_loggers

    Type  boolean
    Default False

    Log requests to multiple loggers.

service_type

    Type  string
    Default image

    The default service_type for endpoint URL discovery.
service_name

Type string
Default <None>

The default service_name for endpoint URL discovery.

valid_interfaces

Type list
Default ['internal', 'public']

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

region_name

Type string
Default <None>

The default region_name for endpoint URL discovery.

endpoint_override

Type string
Default <None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the version, min-version, and/or max-version options.

connect_retries

Type integer
Default <None>

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

connect_retry_delay

Type floating point
Default <None>

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

status_code_retries

Type integer
Default <None>

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

status_code_retry_delay

Type floating point
Default <None>
Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

**gpu_devices**

This is used to config vGPU types for nvidia GPU devices.

**enabled_vgpu_types**

<table>
<thead>
<tr>
<th>Type</th>
<th>list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>[]</td>
</tr>
</tbody>
</table>

The vGPU types enabled in the compute node.

Cyborg supports multiple vGPU types in one host. Usually, a single physical GPU can only set one vgpu type. Some pGPUs (e.g. NVIDIA GRID K1) support multiple vGPU types.

If more than one single vGPU type are provided, then for each vGPU type, you must add an additional section `[vgpu_${VGPU_TYPE}]` with a single configuration option `device_addresses` to assign this type to the target physical GPU(s). PGPUs should be configured explictly now, we will improve this after we implement the enable/disable interface.

If the same PCI address is provided for two different types, cyborg-agent will return an InvalidGPUConfig exception at restart.

An example is as the following:

```
[gpu_devices]
enabled_vgpu_types = nvidia-35, nvidia-36

[vgpu_nvidia-35]
device_addresses = 0000:84:00.0,0000:85:00.0

[vgpu_nvidia-36]
device_addresses = 0000:86:00.0
```

**keystone**

Configuration options for the identity service

**cafile**

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>&lt;None&gt;</td>
</tr>
</tbody>
</table>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

**certfile**

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>&lt;None&gt;</td>
</tr>
</tbody>
</table>

PEM encoded client certificate cert file
keyfile

Type string
Default <None>
PEM encoded client certificate key file

insecure

Type boolean
Default False
Verify HTTPS connections.

timeout

Type integer
Default <None>
Timeout value for http requests

collect_timing

Type boolean
Default False
Collect per-API call timing information.

split_loggers

Type boolean
Default False
Log requests to multiple loggers.

service_type

Type string
Default identity
The default service_type for endpoint URL discovery.

service_name

Type string
Default <None>
The default service_name for endpoint URL discovery.

valid_interfaces

Type list
Default ['internal', 'public']
List of interfaces, in order of preference, for endpoint URL.
region_name

Type: string
Default: <None>

The default region_name for endpoint URL discovery.

endpoint_override

Type: string
Default: <None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the version, min-version, and/or max-version options.

connect_retries

Type: integer
Default: <None>

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

connect_retry_delay

Type: floating point
Default: <None>

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

status_code_retries

Type: integer
Default: <None>

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

status_code_retry_delay

Type: floating point
Default: <None>

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

www_authenticate_uri

  Type  string
  Default  <None>

  Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If 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>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystone_authtoken</td>
<td>auth_uri</td>
</tr>
</tbody>
</table>

auth_uri

  Type  string
  Default  <None>

  Complete public Identity API endpoint. This endpoint should not be an admin endpoint, as it should be accessible by all end users. Unauthenticated clients are redirected to this endpoint to authenticate. Although this endpoint should ideally be unversioned, client support in the wild varies. If 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  <None>

  API version of the Identity API endpoint.

interface

  Type  string
  Default  internal

  Interface to use for the Identity API endpoint. Valid values are public, internal (default) or admin.
**delay_auth_decision**

Type boolean  
Default False  

Do not handle authorization requests within the middleware, but delegate the authorization decision to downstream WSGI components.

**http_connect_timeout**

Type integer  
Default <None>  

Request timeout value for communicating with Identity API server.

**http_request_max_retries**

Type integer  
Default 3  

How many times are we trying to reconnect when communicating with Identity API Server.

**cache**

Type string  
Default <None>  

Request environment key where the Swift cache object is stored. When auth_token middleware is deployed with a Swift cache, use this option to have the middleware share a caching backend with swift. Otherwise, use the memcached_servers option instead.

**certfile**

Type string  
Default <None>  

Required if identity server requires client certificate

**keyfile**

Type string  
Default <None>  

Required if identity server requires client certificate

**cafile**

Type string  
Default <None>  

A PEM encoded Certificate Authority to use when verifying HTTPs connections. Defaults to system CAs.

**insecure**

Type boolean  
Default False
Verify HTTPS connections.

**region_name**

- **Type**: string
- **Default**: <None>
  
The region in which the identity server can be found.

**memcached_servers**

- **Type**: list
- **Default**: <None>
  
Optionally specify a list of memcached server(s) to use for caching. If left undefined, tokens will instead be cached in-process.

### Table 16: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystone_authtoken</td>
<td>memcache_servers</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystone_authtoken</td>
<td>auth_plugin</td>
</tr>
</tbody>
</table>

auth_section

    Type  unknown type
    Default  <None>

Config Section from which to load plugin specific options
**nic_devices**

This is used to config specific nic devices.

**enabled_nic_types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>[]</td>
</tr>
</tbody>
</table>

**nova**

**cafile**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>None</td>
</tr>
</tbody>
</table>

PEM encoded Certificate Authority to use when verifying HTTPS connections.

**certfile**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>None</td>
</tr>
</tbody>
</table>

PEM encoded client certificate cert file

**keyfile**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>None</td>
</tr>
</tbody>
</table>

PEM encoded client certificate key file

**insecure**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>False</td>
</tr>
</tbody>
</table>

Verify HTTPS connections.

**timeout**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>None</td>
</tr>
</tbody>
</table>

Timeout value for http requests

**collect_timing**

<table>
<thead>
<tr>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>False</td>
</tr>
</tbody>
</table>

Collect per-API call timing information.
split_loggers

Type  boolean
Default  False
Log requests to multiple loggers.

auth_type

Type  unknown type
Default  <None>
Authentication type to load

Table 18: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova</td>
<td>auth_plugin</td>
</tr>
</tbody>
</table>

auth_section

Type  unknown type
Default  <None>
Config Section from which to load plugin specific options

auth_url

Type  unknown type
Default  <None>
Authentication URL

system_scope

Type  unknown type
Default  <None>
Scope for system operations

domain_id

Type  unknown type
Default  <None>
Domain ID to scope to

domain_name

Type  unknown type
Default  <None>
Domain name to scope to
**project_id**

Type: unknown type  
Default: <None>

Project ID to scope to

**project_name**

Type: unknown type  
Default: <None>

Project name to scope to

**project_domain_id**

Type: unknown type  
Default: <None>

Domain ID containing project

**project_domain_name**

Type: unknown type  
Default: <None>

Domain name containing project

**trust_id**

Type: unknown type  
Default: <None>

ID of the trust to use as a trustee use

**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.

**user_id**

Type: unknown type  
Default: <None>

User ID
username

    Type unknown type
    Default <None>

Username

Table 19: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>nova</td>
<td>user-name</td>
</tr>
<tr>
<td>nova</td>
<td>user_name</td>
</tr>
</tbody>
</table>

user_domain_id

    Type unknown type
    Default <None>

Users domain id

user_domain_name

    Type unknown type
    Default <None>

Users domain name

password

    Type unknown type
    Default <None>

Users password

tenant_id

    Type unknown type
    Default <None>

Tenant ID

tenant_name

    Type unknown type
    Default <None>

Tenant Name

service_type

    Type string
    Default compute

The default service_type for endpoint URL discovery.

2.2. Configuration Reference
service_name
   Type  string
   Default  <None>
   The default service_name for endpoint URL discovery.

valid_interfaces
   Type  list
   Default  ['internal', 'public']
   List of interfaces, in order of preference, for endpoint URL.

region_name
   Type  string
   Default  <None>
   The default region_name for endpoint URL discovery.

default_endpoint
   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.

connect_retries
   Type  integer
   Default  <None>
   The maximum number of retries that should be attempted for connection errors.

connect_retry_delay
   Type  floating point
   Default  <None>
   Delay (in seconds) between two retries for connection errors. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

status_code_retries
   Type  integer
   Default  <None>
   The maximum number of retries that should be attempted for retriable HTTP status codes.

status_code_retry_delay
   Type  floating point
   Default  <None>
Delay (in seconds) between two retries for retriable status codes. If not set, exponential retry starting with 0.5 seconds up to a maximum of 60 seconds is used.

**oslo_messaging_amqp**

**container_name**

*Type* string  
*Default* `<None>`

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

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>container_name</td>
</tr>
</tbody>
</table>

**idle_timeout**

*Type* integer  
*Default* 0

Timeout for inactive connections (in seconds)

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>idle_timeout</td>
</tr>
</tbody>
</table>

**trace**

*Type* boolean  
*Default* False

Debug: dump AMQP frames to stdout

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>trace</td>
</tr>
</tbody>
</table>

**ssl**

*Type* boolean  
*Default* False

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

**ssl_ca_file**

*Type* string  
*Default* `'`

---

2.2. Configuration Reference
CA certificate PEM file used to verify the servers certificate

Table 23: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>ssl_ca_file</td>
</tr>
</tbody>
</table>

ssl_cert_file

Type string

Default ''

Self-identifying certificate PEM file for client authentication

Table 24: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>ssl_cert_file</td>
</tr>
</tbody>
</table>

ssl_key_file

Type string

Default ''

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

Table 25: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>ssl_key_file</td>
</tr>
</tbody>
</table>

ssl_key_password

Type string

Default <None>

Password for decrypting ssl_key_file (if encrypted)

Table 26: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>ssl_key_password</td>
</tr>
</tbody>
</table>

ssl_verify_vhost

Type boolean

Default False

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

2.2. Configuration Reference
sasl_mechanisms

Type  string
Default  ''

Space separated list of acceptable SASL mechanisms

Table 27: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>sasl_mechanisms</td>
</tr>
</tbody>
</table>

sasl_config_dir

Type  string
Default  ''

Path to directory that contains the SASL configuration

Table 28: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>sasl_config_dir</td>
</tr>
</tbody>
</table>

sasl_config_name

Type  string
Default  ''

Name of configuration file (without .conf suffix)

Table 29: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>sasl_config_name</td>
</tr>
</tbody>
</table>

sasl_default_realm

Type  string
Default  ''

SASL realm to use if no realm present in username

connection_retry_interval

Type  integer
Default  1
Minimum Value  1

Seconds to pause before attempting to re-connect.
connection_retry_backoff

Type integer
Default 2
Minimum Value 0

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

connection_retry_interval_max

Type integer
Default 30
Minimum Value 1

Maximum limit for connection_retry_interval + connection_retry_backoff

link_retry_delay

Type integer
Default 10
Minimum Value 1

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

default_reply_retry

Type integer
Default 0
Minimum Value -1

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

default_reply_timeout

Type integer
Default 30
Minimum Value 5

The deadline for an rpc reply message delivery.

default_send_timeout

Type integer
Default 30
Minimum Value 5

The deadline for an rpc cast or call message delivery. Only used when caller does not provide a timeout expiry.
default_notify_timeout
   Type  integer
   Default  30
   Minimum Value  5

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

default_sender_link_timeout
   Type  integer
   Default  600
   Minimum Value  1

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

addressing_mode
   Type  string
   Default  dynamic

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

pseudo_vhost
   Type  boolean
   Default  True

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

server_request_prefix
   Type  string
   Default  exclusive

   address prefix used when sending to a specific server

   Table 30: Deprecated Variations
<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>server_request_prefix</td>
</tr>
</tbody>
</table>

broadcast_prefix
   Type  string
   Default  broadcast
address prefix used when broadcasting to all servers

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>broadcast_prefix</td>
</tr>
</tbody>
</table>

**group_request_prefix**

**Type** string

**Default** unicast

address prefix when sending to any server in group

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>amqp1</td>
<td>group_request_prefix</td>
</tr>
</tbody>
</table>

**rpc_address_prefix**

**Type** string

**Default** openstack.org/om/rpc

Address prefix for all generated RPC addresses

**notify_address_prefix**

**Type** string

**Default** openstack.org/om/notify

Address prefix for all generated Notification addresses

**multicast_address**

**Type** string

**Default** multicast

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

**unicast_address**

**Type** string

**Default** unicast

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

**anycast_address**

**Type** string

**Default** anycast

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

Type  string
Default  <None>

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

default_rpc_exchange

Type  string
Default  <None>

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

reply_link_credit

Type  integer
Default  200
Minimum Value  1

Window size for incoming RPC Reply messages.

rpc_server_credit

Type  integer
Default  100
Minimum Value  1

Window size for incoming RPC Request messages

notify_server_credit

Type  integer
Default  100
Minimum Value  1

Window size for incoming Notification messages

pre_settled

Type  multi-valued
Default  rpc-cast
Default  rpc-reply

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

2.2. Configuration Reference 53
oslo_messaging_kafka

kafka_max_fetch_bytes

Type  integer
Default  1048576

Max fetch bytes of Kafka consumer

kafka_consumer_timeout

Type  floating point
Default  1.0

Default timeout(s) for Kafka consumers

pool_size

Type  integer
Default  10

Pool Size for Kafka Consumers

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

**Reason** Driver no longer uses connection pool.

conn_pool_min_size

Type  integer
Default  2

The pool size limit for connections expiration policy

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

**Reason** Driver no longer uses connection pool.

conn_pool_ttl

Type  integer
Default  1200

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

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

**Reason** Driver no longer uses connection pool.
consumer_group

  Type  string
  Default oslo_messaging_consumer

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

producer_batch_timeout

  Type  floating point
  Default 0.0

  Upper bound on the delay for KafkaProducer batching in seconds

producer_batch_size

  Type  integer
  Default 16384

  Size of batch for the producer async send

compression_codec

  Type  string
  Default none

  Valid Values  none, gzip, snappy, lz4, zstd

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

enable_auto_commit

  Type  boolean
  Default False

  Enable asynchronous consumer commits

max_poll_records

  Type  integer
  Default 500

  The maximum number of records returned in a poll call

security_protocol

  Type  string
  Default PLAINTEXT

  Valid Values  PLAINTEXT, SASL_PLAINTEXT, SSL, SASL_SSL

  Protocol used to communicate with brokers

sasl_mechanism

  Type  string
  Default PLAIN
Mechanism when security protocol is SASL

ssl_cafile

Type: string
Default: ''
CA certificate PEM file used to verify the server certificate

ssl_client_cert_file

Type: string
Default: ''
Client certificate PEM file used for authentication.

ssl_client_key_file

Type: string
Default: ''
Client key PEM file used for authentication.

ssl_client_key_password

Type: string
Default: ''
Client key password file used for authentication.

oslo_messaging_notifications

driver

Type: multi-valued
Default: ''
The Drivers(s) to handle sending notifications. Possible values are messaging, messagingv2, routing, log, test, noop

Table 33: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>notification_driver</td>
</tr>
</tbody>
</table>

transport_url

Type: string
Default: <None>
A URL representing the messaging driver to use for notifications. If not set, we fall back to the same configuration used for RPC.
Table 34: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>notification_transport_url</td>
</tr>
</tbody>
</table>

**topics**

**Type** list

**Default** ['notifications']

AMQP topic used for OpenStack notifications.

Table 35: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpc_notifier2</td>
<td>topics</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>notification_topics</td>
</tr>
</tbody>
</table>

**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.

Table 36: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>amqp_auto_delete</td>
</tr>
</tbody>
</table>

**ssl**

**Type** boolean

**Default** False
Connect over SSL.

### Table 37: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>rabbit_use_ssl</td>
</tr>
</tbody>
</table>

### ssl_version

**Type** string  
**Default** ''  
SSL version to use (valid only if SSL enabled). Valid values are TLSv1 and SSLv23. SSLv2, SSLv3, TLSv1_1, and TLSv1_2 may be available on some distributions.

### Table 38: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>kombu_ssl_version</td>
</tr>
</tbody>
</table>

### ssl_key_file

**Type** string  
**Default** ''  
SSL key file (valid only if SSL enabled).

### Table 39: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>kombu_ssl_keyfile</td>
</tr>
</tbody>
</table>

### ssl_cert_file

**Type** string  
**Default** ''  
SSL cert file (valid only if SSL enabled).

### Table 40: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>kombu_ssl_certfile</td>
</tr>
</tbody>
</table>

### ssl_ca_file

**Type** string  
**Default** ''  
SSL certification authority file (valid only if SSL enabled).

### Table 41: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>kombu_ssl_ca_certs</td>
</tr>
</tbody>
</table>
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  True

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.

kombu_reconnect_delay

Type  floating point
Default  1.0

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

Table 42: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>kombu_reconnect_delay</td>
</tr>
</tbody>
</table>

kombu_compression

Type  string
Default  "<None>"

EXPERIMENTAL: Possible values are: gzip, bz2. If not set compression will not be used. This option may not be available in future versions.

kombu_missing_consumer_retry_timeout

Type  integer
Default  60

How long to wait a missing client before abandoning to send it its replies. This value should not be longer than rpc_response_timeout.

Table 43: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>oslo_messaging_rabbit</td>
<td>kombu_reconnect_timeout</td>
</tr>
</tbody>
</table>
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.

Table 44: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>rabbit_login_method</td>
</tr>
</tbody>
</table>

rabbit_retry_interval

Type  integer
Default  1

How frequently to retry connecting with RabbitMQ.

rabbit_retry_backoff

Type  integer
Default  2

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

Table 45: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>rabbit_retry_backoff</td>
</tr>
</tbody>
</table>

rabbit_interval_max

Type  integer
Default  30

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

rabbit_ha_queues

Type  boolean
Default  False
Try to use HA queues in RabbitMQ (x-ha-policy: all). If you change this option, you must wipe the RabbitMQ database. In RabbitMQ 3.0, queue mirroring is no longer controlled by the x-ha-policy argument when declaring a queue. If you just want to make sure that all queues (except those with auto-generated names) are mirrored across all nodes, run: rabbitmqctl set_policy HA ^(?!amq\.)\.* {ha-mode: all}

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>rabbit_ha_queues</td>
</tr>
</tbody>
</table>

**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 durable by default so the amqp_durable_queues option is ignored when this option enabled.

**rabbit_transient_queues_ttl**

*Type* integer

*Default* 1800

*Minimum Value* 1

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

**rabbit_qos_prefetch_count**

*Type* integer

*Default* 0

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

**heartbeat_timeout_threshold**

*Type* integer

*Default* 60

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

**heartbeat_rate**

*Type* integer

*Default* 2

How often times during the heartbeat_timeout_threshold we check the heartbeat.

2.2. Configuration Reference
**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.

**oslo_policy**

**enforce_scope**

*Type*  boolean  

*Default*  False  

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

**enforce_new_defaults**

*Type*  boolean  

*Default*  False  

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

**policy_file**

*Type*  string  

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

Table 47: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>policy_file</td>
</tr>
</tbody>
</table>

**policy_default_rule**

Type string  
Default default  

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

Table 48: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>policy_direction_rule</td>
</tr>
</tbody>
</table>

**policy_dirs**

Type multi-valued  
Default policy.d  

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

Table 49: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>policy_dirs</td>
</tr>
</tbody>
</table>

**remote_content_type**

Type string  
Default application/x-www-form-urlencoded  
Valid Values application/x-www-form-urlencoded, application/json  

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

**remote_ssl_verify_server_crt**

Type boolean  
Default False  

Server identity verification for REST based policy check

**remote_ssl_ca_crt_file**

Type string  
Default <None>
Absolute path to ca cert file for REST based policy check

**remote_ssl_client_crt_file**

- **Type** string
- **Default** <None>

Absolute path to client cert for REST based policy check

**remote_ssl_client_key_file**

- **Type** string
- **Default** <None>

Absolute path client key file REST based policy check

**placement**

**cafile**

- **Type** string
- **Default** <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

**certfile**

- **Type** string
- **Default** <None>

PEM encoded client certificate cert file

**keyfile**

- **Type** string
- **Default** <None>

PEM encoded client certificate key file

**insecure**

- **Type** boolean
- **Default** False

Verify HTTPS connections.

**timeout**

- **Type** integer
- **Default** <None>

Timeout value for http requests
collect_timing
  Type boolean
  Default False
  Collect per-API call timing information.

split_loggers
  Type boolean
  Default False
  Log requests to multiple loggers.

auth_type
  Type unknown type
  Default <None>
  Authentication type to load

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>placement</td>
<td>auth_plugin</td>
</tr>
</tbody>
</table>

auth_section
  Type unknown type
  Default <None>
  Config Section from which to load plugin specific options

auth_url
  Type unknown type
  Default <None>
  Authentication URL

system_scope
  Type unknown type
  Default <None>
  Scope for system operations

domain_id
  Type unknown type
  Default <None>
  Domain ID to scope to
domain_name
  Type  unknown type
  Default  <None>
  Domain name to scope to

project_id
  Type  unknown type
  Default  <None>
  Project ID to scope to

project_name
  Type  unknown type
  Default  <None>
  Project name to scope to

project_domain_id
  Type  unknown type
  Default  <None>
  Domain ID containing project

project_domain_name
  Type  unknown type
  Default  <None>
  Domain name containing project

trust_id
  Type  unknown type
  Default  <None>
  ID of the trust to use as a trustee use

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.
user_id

    Type  unknown type
    Default  <None>

User ID

username

    Type  unknown type
    Default  <None>

Username

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>placement</td>
<td>user-name</td>
</tr>
<tr>
<td>placement</td>
<td>user_name</td>
</tr>
</tbody>
</table>

Table 51: Deprecated Variations

user_domain_id

    Type  unknown type
    Default  <None>

Users domain id

user_domain_name

    Type  unknown type
    Default  <None>

Users domain name

password

    Type  unknown type
    Default  <None>

Users password

tenant_id

    Type  unknown type
    Default  <None>

Tenant ID

tenant_name

    Type  unknown type
    Default  <None>

Tenant Name
service_type

Type  string
Default  placement

The default service_type for endpoint URL discovery.

service_name

Type  string
Default  <None>

The default service_name for endpoint URL discovery.

valid_interfaces

Type  list
Default  ['internal', 'public']

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

region_name

Type  string
Default  <None>

The default region_name for endpoint URL discovery.

endpoint_override

Type  string
Default  <None>

Always use this endpoint URL for requests for this client. NOTE: The unversioned endpoint should be specified here; to request a particular API version, use the version, min-version, and/or max-version options.

connect_retries

Type  integer
Default  <None>

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

connect_retry_delay

Type  floating point
Default  <None>

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

status_code_retries

Type  integer
Default  <None>
The maximum number of retries that should be attempted for retriable HTTP status codes.

**status_code_retry_delay**

Type: floating point

Default: <None>

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

**service_user**

Configuration options for service to service authentication using a service token. These options allow sending a service token along with the users token when contacting external REST APIs.

**send_service_user_token**

Type: boolean

Default: False

When True, if sending a user token to a REST API, also send a service token.

**cafile**

Type: string

Default: <None>

PEM encoded Certificate Authority to use when verifying HTTPs connections.

**certfile**

Type: string

Default: <None>

PEM encoded client certificate cert file

**keyfile**

Type: string

Default: <None>

PEM encoded client certificate key file

**insecure**

Type: boolean

Default: False

Verify HTTPS connections.

**timeout**

Type: integer

Default: <None>

Timeout value for http requests
collect_timing

Type boolean
Default False
Collect per-API call timing information.

split_loggers

Type boolean
Default False
Log requests to multiple loggers.

auth_type

Type unknown type
Default <None>
Authentication type to load

auth_section

Type unknown type
Default <None>
Config Section from which to load plugin specific options

auth_url

Type unknown type
Default <None>
Authentication URL

system_scope

Type unknown type
Default <None>
Scope for system operations

domain_id

Type unknown type
Default <None>
Domain ID to scope to
domain_name

  Type  unknown type
  Default  <None>
  Domain name to scope to

project_id

  Type  unknown type
  Default  <None>
  Project ID to scope to

project_name

  Type  unknown type
  Default  <None>
  Project name to scope to

project_domain_id

  Type  unknown type
  Default  <None>
  Domain ID containing project

project_domain_name

  Type  unknown type
  Default  <None>
  Domain name containing project

trust_id

  Type  unknown type
  Default  <None>
  ID of the trust to use as a trustee use

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.
user_id

Type unknown type
Default <None>

User ID

username

Type unknown type
Default <None>

Username

Table 53: Deprecated Variations

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_user</td>
<td>user-name</td>
</tr>
<tr>
<td>service_user</td>
<td>user_name</td>
</tr>
</tbody>
</table>

user_domain_id

Type unknown type
Default <None>

Users domain id

user_domain_name

Type unknown type
Default <None>

Users domain name

password

Type unknown type
Default <None>

Users password

tenant_id

Type unknown type
Default <None>

Tenant ID

tenant_name

Type unknown type
Default <None>

Tenant Name
ssl

c_a_file

Type string

Default <None>

CA certificate file to use to verify connecting clients.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>ssl_ca_file</td>
</tr>
</tbody>
</table>

cert_file

Type string

Default <None>

Certificate file to use when starting the server securely.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>ssl_cert_file</td>
</tr>
</tbody>
</table>

key_file

Type string

Default <None>

Private key file to use when starting the server securely.

<table>
<thead>
<tr>
<th>Group</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT</td>
<td>ssl_key_file</td>
</tr>
</tbody>
</table>

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.
**Cyborg Configuration Sample**

Sample configuration files for all Cyborg services are found in the online version of this documentation.

**Cyborg Sample Policy**

```
# Default rule for System Admin APIs.
"system_admin_api": "role:admin and system_scope:all"

# Default rule for System level read only APIs.
"system_reader_api": "role:reader and system_scope:all"

# Default rule for Project level admin APIs.
"project_admin_api": "role:admin and project_id:%(project_id)s"

# Default rule for Project level non admin APIs.
"project_member_api": "role:member and project_id:%(project_id)s"

# Default rule for Project level read only APIs.
"project_reader_api": "role:reader and project_id:%(project_id)s"

# Default rule for system_admin+owner APIs.
"system_admin_or_owner": "rule:system_admin_api or rule:project_member_api"

# Default rule for System+Project read only APIs.
"system_or_project_reader": "rule:system_reader_api or rule:project_reader_api"

# DEPRECATED
# "public_api" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
```

(continues on next page)
# consideration if you are relying on overrides in your deployment for
# the policy API.
# legacy rule of Internal flag for public API routes
# "public_api": "is_public_api:True"

# DEPRECATED
# "allow" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# legacy rule: any access will be passed
# "allow": "@"

# DEPRECATED
# "deny" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# legacy rule: all access will be forbidden
# "deny": "!"

# DEPRECATED
# "default" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# Legacy rule for default rule
# "default": "rule:admin_or_owner"

# DEPRECATED
# "admin_api" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# Legacy rule for cloud admin access
# "admin_api": "role:admin or role:administrator"
# DEPRECATED
# "is_admin" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# Full read/write API access
# "is_admin": "rule:admin_api"

# DEPRECATED
# "admin_or_owner" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# Admin or owner API access
# "admin_or_owner": "is_admin:True or project_id:%(project_id)s"

# DEPRECATED
# "admin_or_user" has been deprecated since W.
# Cyborg API policies are introducing new default roles with
# scope_type capabilities. We will start to deprecate old policies
# from WALLABY release, and are going to ignore all the old policies
# silently from X release. Be sure to take these new defaults into
# consideration if you are relying on overrides in your deployment for
# the policy API.
# Admin or user API access
# "admin_or_user": "is_admin:True or user_id:%(user_id)s"

# Retrieve all device_profiles
# GET /v2/device_profiles
# Intended scope(s): system, project
# "cyborg:device_profile:get_all": "rule:system_or_project_reader"

# DEPRECATED
# "cyborg:device_profile:get_all": "rule:admin_or_owner" has been
deprecated since W in favor of
# "cyborg:device_profile:get_all": "rule:system_or_project_reader".
# request admin_or_owner rule is too strict for listing device_profile

# Retrieve a specific device_profile
# GET /v2/device_profiles/{device_profiles_uuid}
# Intended scope(s): system, project
# "cyborg:device_profile:get_one": "rule:system_or_project_reader"
# DEPRECATED
# "cyborg:device_profile:get_one": "rule:admin_or_owner" has been deprecated since W in favor of
# "cyborg:device_profile:get_one": "rule:system_or_project_reader".
# request admin_or_owner rule is too strict for retrieving a
# device_profile

# Create a device_profile
# POST /v2/device_profiles
# Intended scope(s): system
# "cyborg:device_profile:create": "rule:system_admin_api"

# DEPRECATED
# "cyborg:device_profile:create": "rule:is_admin" has been deprecated since W in favor of
# "cyborg:device_profile:create": "rule:system_admin_api".
# project_admin_or_owner is too permissive, introduce systemScoped admin for creation

# Delete device_profile(s)
# DELETE /v2/device_profiles/{device_profiles_uuid}
# DELETE /v2/device_profiles?value={device_profile_name1}
# Intended scope(s): system
# "cyborg:device_profile:delete": "rule:system_admin_api"

# DEPRECATED
# "cyborg:device_profile:delete": "rule:admin_or_owner" has been deprecated since W in favor of
# "cyborg:device_profile:delete": "rule:system_admin_api".
# project_admin_or_owner is too permissive, introduce systemScoped admin for deletion

# Show device detail
# "cyborg:device:get_one": "rule:allow"

# Retrieve all device records
# "cyborg:device:get_all": "rule:allow"

# Show deployable detail
# "cyborg:deployable:get_one": "rule:allow"

# Retrieve all deployable records
# "cyborg:deployable:get_all": "rule:allow"

# FPGA programming.
# "cyborg:deployable:program": "rule:allow"

# Retrieve accelerator request records.
# "cyborg:arq:get_all": "rule:default"
# Get an accelerator request record.
"cyborg:arq:get_one": "rule:default"

# Create accelerator request records.
"cyborg:arq:create": "rule:allow"

# Delete accelerator request records.
"cyborg:arq:delete": "rule:default"

# Update accelerator request records.
"cyborg:arq:update": "rule:default"

# Show fpga detail
"cyborg:fpga:get_one": "rule:allow"

# Retrieve all fpga records
"cyborg:fpga:get_all": "rule:allow"

# Update fpga records
"cyborg:fpga:update": "rule:allow"

## Cyborg Policy Configuration Guide

Cyborg, like most OpenStack projects, uses a policy language to restrict permissions on REST API actions.

- **Policy Concepts**: In the Victoria release, Cyborg API policy defines new default roles with system scope capabilities. These new changes improve the security level and manageability of Cyborg API as they are richer in terms of handling access at system and project level token with Read and Write roles.

### Understanding Cyborg Policies

**Warning**: JSON formatted policy file is deprecated since Cyborg (Victoria). Use YAML formatted file. Use oslopolicy-convert-json-to-yaml tool to convert the existing JSON to YAML formatted policy file in backward compatible way.

Cyborg supports a rich policy system that has evolved significantly over its lifetime. Initially, cyborg policy defaults have been defined in the codebase, requiring the policy.json file only to override these defaults. Starting in the Victoria release, policy file has been changed from policy.json to policy.yaml.

The old default policy in Cyborg is incomplete and not good enough. Since Cyborg V2 API is newly implemented in Train, RBAC check for V2 API still remains incomplete. So in the Ussuri release, the
specification of policy refresh was approved. In the Victoria release, Cyborg landed the new default roles to improve some issues that had been identified:

1. **No allow.** Old policy allow means any access will be passed. allow rule was used by cyborg:arq:create, which is too slack.

2. **No global vs project admin.** The old role is_admin is used for the global admin that is able to make almost any change to Cyborg, and see all details of the Cyborg system. The rule passes for any user with an admin role, it does not matter which project is used.

3. **No admin_or_owner.** Old role admin_or_owner sounds like it checks if the user is a member of a project. However, for most APIs we use the default target which means this rule will pass for any authenticated user.

4. **Introduce scope_type and reader role.** There still some cases which are not well covered. For example, it is impossible to allow a user to retrieve/update devices which are shared by multiple projects from a system level without being given the global admin role. In addition, cyborg now doesn't have a reader role.

Keystone comes with admin, member and reader roles by default. Please refer to the [keystone documentation](http://docs.openstack.org/keystone) for more information about these new defaults. In addition, keystone supports a new system scope concept that makes it easier to protect deployment level resources from project or system level resources. Please refer to the [token scopes](http://docs.openstack.org/keystone) and [system scope specification](http://docs.openstack.org/keystone) to understand the scope concept.

In the Cyborg (Victoria) release, Cyborg policies implemented the scope concept and default roles provided by keystone (admin, member, and reader). Using common roles from keystone reduces the likelihood of similar, but different, roles implemented across projects or deployments. With the help of the new defaults it is easier to understand who can do what across projects, reduces divergence, and increases interoperability.

The below sections explain how these new defaults in the Cyborg can solve the issues mentioned above and extend more functionality to end users in a safe and secure way.

More information is provided in the [cyborg specification](http://docs.openstack.org/keystone).

### Scope

OpenStack Keystone supports different scopes in tokens. These are described [here](http://docs.openstack.org/keystone). Token scopes represent the layer of authorization. Policy scope_types represent the layer of authorization required to access an API.

**Note:** The scope_type of each policy is hardcoded and is not overridable via the policy file.

Cyborg policies have implemented the scope concept by defining the scope_type in policies. To know each policy's scope_type, please refer to the [Policy Reference](http://docs.openstack.org/keystone) and look for Scope Types or Intended scope(s) in Policy Sample File as shown in below examples.
system scope

Policies with a scope_type of system means a user with a system-scoped token has permission to access the resource. This can be seen as a global role. All the system-level operations policies have defaulted to scope_type of ['system'].

For example, consider the POST /v2/device_profiles API.

```
# Create a device_profile
# POST /v2/device_profiles
# Intended scope(s): system
#"cyborg:device_profile:create": "rule:system_admin_api"
```

project scope

Policies with a scope_type of project means a user with a project-scoped token has permission to access the resource. This can be seen as a project role. All the project-level operations policies should be set to scope_type of ['project'] by default.

system and project scope

Policies with a scope_type of system and project means a user with a system-scoped or project-scoped token has permission to access the resource. All the system and project level operations policies have defaulted to scope_type of ['system', 'project'].

For example, consider the GET /v2/device_profiles/{device_profiles_uuid} API.

```
# Retrieve a specific device_profile
# GET /v2/device_profiles/{device_profiles_uuid}
# Intended scope(s): system, project
#"cyborg:device_profile:get_one": "rule:system_or_project_reader"
```

These scope types provide a way to differentiate between system-level and project-level access roles. You can control the information with scope of the users.

Policy scope is disabled by default to allow operators to migrate from the old policy enforcement system in a graceful way. This can be enabled by configuring the oslo_policy.enforce_scope option to True.

Note: [oslo_policy] enforce_scope=True
Roles

You can refer to this document to know about all available defaults from Keystone.

Along with the `scope_type` feature, Cyborg policy defines new defaults for each policy.

**reader**

This provides read-only access to the resources within the system or project. Cyborg policies are defaulted to below rules:

```
system_reader_api
  Default
    role:reader and system_scope:all

project_reader_api
  Default
    role:reader and project_id:%(project_id)s

system_or_project_reader
  Default
    rule:system_reader_api or rule:project_reader_api
```

**member**

This role is to perform the project level write operation with combination to the system admin. Cyborg policies are defaulted to below rules:

```
project_member_api
  Default
    role:member and project_id:%(project_id)s

system_admin_or_owner
  Default
    rule:system_admin_api or rule:project_member_api
```

**admin**

This role is to perform the admin level write operation at system as well as at project-level operations. Cyborg policies are defaulted to below rules:

```
system_admin_api
  Default
    role:admin and system_scope:all

project_admin_api
  Default
    role:admin and project_id:%(project_id)s
```

(continues on next page)
With these new defaults, you can solve the problem of:

1. Providing the read-only access to the user. Polices are made more granular and defaulted to reader rules. For example: If you need to let someone audit your deployment for security purposes.

2. Customize the policy in better way. For example, you will be able to provide access to project level member to perform arq patch/post for instance boot with the projects token.

**Backward Compatibility**

During the development period (Victoria and Wallaby releases), the new and old policy will both work for backward compatibility by supporting the old defaults and disabling the `scope_type` feature by default. This means the old defaults and deployments that use them will keep working as-is. However, we encourage every deployment to switch to new policy. `scope_type` will be enabled by default and the old defaults will be removed starting in the X release.

To implement the new default reader roles, some policies needed to become granular. They have been renamed, with the old names still supported for backwards compatibility.

**Migration Plan**

To have a graceful migration, Cyborg provides two flags to switch to the new policy completely. You do not need to overwrite the policy file to adopt the new policy defaults.

Here is step wise guide for migration:

1. Create scoped token:
   - You need to create the new token with scope knowledge via below CLI:
     - Create System Scoped Token.
     - Create Project Scoped Token.

2. Create new default roles in keystone if not done:
   - If you do not have new defaults in Keystone then you can create and re-run the Keystone Bootstrap. Keystone added this support in 14.0.0 (Rocky) release.

3. Enable Scope Checks
   - The `oslo_policy.enforce_scope` flag is to enable the `scope_type` features. The scope of the token used in the request is always compared to the `scope_type` of the policy. If the scopes do not match, one of two things can happen. If `oslo_policy.enforce_scope` is True, the request will be rejected. If `oslo_policy.enforce_scope` is False, an warning will be logged, but the request will be accepted (assuming the rest of the policy passes). The default value of this flag is False.
Note: Before you enable this flag, you need to audit your users and make sure everyone who needs system-level access has a system role assignment in keystone.

4. Enable new defaults

The `oslo_policy.enforce_new_defaults` flag switches the policy to new defaults-only. This flag controls whether or not to use old deprecated defaults when evaluating policies. If True, the old deprecated defaults are not evaluated. This means if any existing token is allowed for old defaults but is disallowed for new defaults, it will be rejected. The default value of this flag is False.

Note: Before you enable this flag, you need to educate users about the different roles they need to use to continue using Cyborg APIs.

5. Check for deprecated policies

A few policies were made more granular to implement the reader roles. New policy names are available to use. If old policy names which are renamed are overwritten in policy file, then warning will be logged. Please migrate those policies to new policy names.

We expect all deployments to migrate to new policy by X release so that we can remove the support of old policies.


**Cyborg Policies**

The following is an overview of all available policies in Cyborg.

**Warning:** JSON formatted policy file is deprecated since Cyborg (Victoria). Use YAML formatted file. Use `oslopolicy-convert-json-to-yaml` tool to convert the existing JSON to YAML formatted policy file in backward compatible way.

**cyborg.api**

**system_admin_api**

Default role:admin and system_scope:all

Default rule for System Admin APIs.

**system_reader_api**

Default role:reader and system_scope:all

Default rule for System level read only APIs.

**project_admin_api**

Default role:admin and project_id:%(project_id)s

Default rule for Project level admin APIs.
**project_member_api**

*Default* role:member and project_id:%(project_id)s

Default rule for Project level non admin APIs.

**project_reader_api**

*Default* role:reader and project_id:%(project_id)s

Default rule for Project level read only APIs.

**system_admin_or_owner**

*Default* rule:system_admin_api or rule:project_member_api

Default rule for system_admin+owner APIs.

**system_or_project_reader**

*Default* rule:system_reader_api or rule:project_reader_api

Default rule for System+Project read only APIs.

**public_api**

*Default* is_public_api:True

legacy rule of Internal flag for public API routes

**allow**

*Default* @

legacy rule: any access will be passed

**deny**

*Default* !

legacy rule: all access will be forbidden

**default**

*Default* rule:admin_or_owner

Legacy rule for default rule

**admin_api**

*Default* role:admin or role:administrator

Legacy rule for cloud admin access

**is_admin**

*Default* rule:admin_api

Full read/write API access

**admin_or_owner**

*Default* is_admin:True or project_id:%(project_id)s

Admin or owner API access

**admin_or_user**
Default is_admin:True or user_id:%(user_id)s
Admin or user API access

cyborg:device_profile:get_all
Default rule:system_or_project_reader
Operations
• GET /v2/device_profiles
Scope Types
• system
• project
Retrieve all device_profiles

cyborg:device_profile:get_one
Default rule:system_or_project_reader
Operations
• GET /v2/device_profiles/{device_profiles_uuid}
Scope Types
• system
• project
Retrieve a specific device_profile

cyborg:device_profile:create
Default rule:system_admin_api
Operations
• POST /v2/device_profiles
Scope Types
• system
Create a device_profile

cyborg:device_profile:delete
Default rule:system_admin_api
Operations
• DELETE /v2/device_profiles/{device_profiles_uuid}
• DELETE /v2/device_profiles?value={device_profile_name1}
Scope Types
• system
Delete device_profile(s)
cyborg:device:get_one
Default rule:allow
Show device detail
cyborg:device:get_all
  Default rule:allow
  Retrieve all device records
cyborg:deployable:get_one
  Default rule:allow
  Show deployable detail
cyborg:deployable:get_all
  Default rule:allow
  Retrieve all deployable records
cyborg:deployable:program
  Default rule:allow
  FPGA programming.
cyborg:arq:get_all
  Default rule:default
  Retrieve accelerator request records.
cyborg:arq:get_one
  Default rule:default
  Get an accelerator request record.
cyborg:arq:create
  Default rule:allow
  Create accelerator request records.
cyborg:arq:delete
  Default rule:default
  Delete accelerator request records.
cyborg:arq:update
  Default rule:default
  Update accelerator request records.
cyborg:fpga:get_one
  Default rule:allow
  Show fpga detail
cyborg:fpga:get_all
  Default rule:allow
Retrieve all fpga records

**cyborg:fpga:update**

- **Default** rule:allow

Update fpga records

### 2.2.2 Cyborg Support Matrix

Cyborg supports specific operations on VMs with attached accelerator resources, which are generally a subset of the full set of VM operations supported by Nova (*nova-vm-ops*).

In this release, these operations have a dependency on specific Nova patches (*nova-patches*). They can be expected to work in Cyborg only if and when these Nova patches get merged without significant changes. These operations are not supported in this release since the dependencies are not met.

<table>
<thead>
<tr>
<th>VM Operation</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM creation</td>
<td>openstack server create</td>
</tr>
<tr>
<td>VM deletion</td>
<td>openstack server delete</td>
</tr>
<tr>
<td>Reboot within VM</td>
<td>ssh to VM and reboot in OS</td>
</tr>
<tr>
<td>Soft reboot</td>
<td>openstack server reboot --soft</td>
</tr>
<tr>
<td>Pause/Unpause</td>
<td>openstack server pause, openstack server unpause</td>
</tr>
<tr>
<td>Backup</td>
<td>openstack server backup create</td>
</tr>
<tr>
<td>Take a snapshot</td>
<td>openstack server image create</td>
</tr>
<tr>
<td>Lock/Unlock</td>
<td>openstack server lock, openstack server unlock</td>
</tr>
<tr>
<td>Rebuild/Evacuate</td>
<td>openstack server rebuild</td>
</tr>
<tr>
<td>Shelve/Unshelve</td>
<td>openstack server shelve, openstack server unshelve</td>
</tr>
</tbody>
</table>

Operations not listed here may or may not work.
Driver Support

The list of drivers available as part of the Cyborg distribution at the time of release can be found in: cyborg.accelerator.driver section of Cyborgs setup.cfg

The following table provides additional information for individual drivers.

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Supported Products</th>
<th>Description</th>
<th>Notes</th>
<th>Temporary Test Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fake Driver</td>
<td>None</td>
<td>A driver that creates a fake device with accelerator resources of type FPGA. Useful for exploring Cyborg without hardware and for Continuous Integration testing.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Intel FPGA OPAE Driver</td>
<td>Intel PAC</td>
<td>The driver for Intel FPGA devices with OPAE software stack.</td>
<td>Supports programming of FPGA bitstreams of type gbs.</td>
<td>None</td>
</tr>
<tr>
<td>Nvidia GPU driver</td>
<td>None</td>
<td>The driver for Nvidia GPUs.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Ascend AI Chip driver</td>
<td>None</td>
<td>The driver for Huaweis Ascend AI chips.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Inspur FPGA Driver</td>
<td>None</td>
<td>The driver for Inspur FPGA Cards.</td>
<td>None</td>
<td>Test results reported at Aug 2020. Please reference: Inspur FPGA Driver Test Report</td>
</tr>
<tr>
<td>Intel NIC Driver</td>
<td>None</td>
<td>The driver for Intel NIC Cards.</td>
<td>None</td>
<td>Test results reported at Feb 2021. Please reference: Intel NIC Driver Test Report</td>
</tr>
<tr>
<td>Inspur NVMe SSD Driver</td>
<td>None</td>
<td>The driver for Inspur NVMe SSD DISK.</td>
<td>None</td>
<td>Test results reported at Feb 2021. Please reference: Inspur NVMe SSD Driver Test Report</td>
</tr>
</tbody>
</table>

Note: Temporary Test Report: This is a temporary test report, it is only valid for a short time, if you encounter problems, please contact the Cyborg team.
2.3 Maintenance

Once you are running cyborg, the following information is extremely useful.

- **Admin Guide**: A collection of guides for administrating cyborg.

2.3.1 Acceleration Service

The OpenStack Cyborg is running as an acceleration service that allows you to manage the lifecycle of accelerating for an instance in cloud computing platform. It gives you control over accelerators attached to instances easily.

**Overview**

A good understand on how Cyborg interacts with Nova and Placement help operators manage the acceleration service more effectively.
Coexistence with PCI whitelists

The operator tells Nova which PCI devices to claim and to be used by configuring the PCI Whitelists mechanism. In addition, the operator installs Cyborg drivers in compute nodes and configures/enables them. Those drivers may then discover and report some PCI devices. The operator must ensure that both configurations are compatible.

Ideally, there is a single way for the operator to identify which PCI devices should be claimed by Nova and which by Cyborg. Until that is figured out, the operator shall use Cyborgs configuration file to specify which Cyborg drivers are enabled. Since each driver claims specific PCI IDs, the operator can and must ensure that none of these PCI IDs are included in Novas PCI whitelist.

Placement update

Cyborg conductor calls Placement API directly to represent devices and accelerators. Some of the intended use cases for the API invocation are:

- Create or delete child RPs under the compute node RP.
- Create or delete custom RCs and custom traits.
- Associate traits with RPs or remove such association.
- Update RP inventory.

Cyborg shall not modify the RPs created by any other component, such as Nova virt drivers.

User Requests

The user request for accelerators is encapsulated in a device profile, which is created and managed by the admin via the Cyborg API.

The structure overview of a `device_profile` is like this:

```
{
    "device_profiles": [
        {
            "name": "fpga-dp1",
            "uuid": "5518a925-1c2c-49a2-a8bf-0927d9456f3e",
            "description": "",
            "groups": [
                {
                    "trait": "CUSTOM_FPGA_TRAITS": "required",
                    "resources": "FPGA": "1",
                    "accel": "bitstream_id": "d5ca2f11-3108-4426-a11c-a959987565df"
                }
            ],
            "created_at": "2020-03-10 03:52:15+00:00",
            "updated_at": null,
            "links": [
                {
                    "href": "http://192.168.32.217/accelerator/v2/device_profiles/5518a925-1c2c-49a2-a8bf-0927d9456f3e"
                }
            ]
        }
    ]
}
```
The device profile is folded into the flavor as an extra spec by the operator, as below:

```
openstack flavor set --property 'accel:device_profile=<profile_name>' flavor
```

Thus the standard Nova API can be used to create an instance with only the flavor (without device profiles), like this:

```
openstack server create --flavor f .... # instance creation
```

In the future, device profile may be used by itself to specify accelerator resources for the instance creation API.

### Updating the Request Spec

When the user submits a request to create an instance, as described in Section User Requests, Nova needs to call a Cyborg API, to get back the resource request groups in the device profile and merge them into the request spec.

This call, like all the others that Nova would make to Cyborg APIs, is done through a Keystone-based adapter that would locate the Cyborg service, similar to the way Nova calls Placement. A Cyborg client module added to Nova, will encapsulate such calls.

VM images in Glance may be associated with image properties (other than image traits), such as bitstream/function IDs needed for that image. So, Nova should pass the VM image UUID from the request spec to Cyborg.

The groups in the device profile are numbered by Cyborg. The request groups that are merged into the request spec are numbered by Nova. These numberings would not be the same in general, i.e., the N-th device profile group may not correspond to the N-th request group in the request spec.

When the device profile request groups are added to other request groups in the flavor, the group_policy of the flavor shall govern the overall semantics of all request groups.

### Accelerator Requests

An accelerator request (ARQ) is an object that represents the state of the request for an accelerator to be assigned to an instance. The creation and management of ARQs are handled by Cyborg, and ARQs are persisted in Cyborg database.

An ARQ represents a request for a single accelerator by definition. The device profile in the user request may have N request groups, each asking for M accelerators, then N * M ARQs will be created for that device profile.

When an ARQ is initially created by Cyborg, it is not yet associated with a specific host name or a device resource provider. So it is said to be in an unbound state. Subsequently, Nova calls Cyborg to bind the
ARQ to a host name, a device RP UUID and an instance UUID. If the instance fails to spawn, Nova would unbind the ARQ with deleting it. On instance termination, Nova would delete the ARQs after unbinding them.

Each ARQ needs to be matched to the specific RP in the allocation candidate that Nova has chosen, before the ARQ is bound. The current Nova code maps request groups to RPs, while the Cyborg client module in Nova (cyborg-client-module) matches ARQs to request groups. The matching is done using the request_id field in the RequestGroup object as below:

- The order of request groups in a device profile is not significant, but it is preserved by Cyborg. Thus, each device profile request group has a unique index.

- When the device profile request groups returned by Cyborg are added to the request spec, the request_id field is set to device_profile_<N> for the N-th device profile request group (starting from zero). The device profile name need not be included here because there is only one device profile per request spec.

- When Cyborg creates an ARQ for a device profile, it embeds the device profile request group index in the ARQ before returning it to Nova.

- The matching is done in two steps:
  - Each ARQ is mapped to a specific request group in the request spec using the request_id field.
  - Each request group is mapped to a specific RP using the same logic as the Neutron bandwidth provider.

**Cyborg and Nova interaction workflow**

This flow is captured by the following sequence diagram, in which the Nova conductor and scheduler are together represented as the Nova controller.
A Cyborg client module is added to nova (cyborg-client-module). All Cyborg API calls are routed through that.

1. The Nova API server receives a `POST /servers` API request with a flavor that includes a device profile name.

2. The Nova API server calls the Cyborg API `GET /v2/device_profiles?name=$device_profile_name` and gets back the device profile. The request groups in that device profile are added to the request spec.

3. The Nova scheduler invokes Placement and gets a list of allocation candidates. It selects one of those candidates and makes claim(s) in Placement. The Nova conductor then sends a RPC message `build_and_run_instances` to the Nova compute manager.

4. Nova conductor manager calls the Cyborg API `POST /v2/accelerator_requests` with the device profile name. Cyborg creates a set of unbound ARQs for that device profile and returns them to Nova.

5. The Cyborg client in Nova matches each ARQ to the resource provider picked for that accelerator.
6. The Nova compute manager calls the Cyborg API `PATCH /v2/accelerator_requests` to bind the ARQ with the host name, devices RP UUID and instance UUID. This is an asynchronous call which prepares or reconfigures the device in the background.

7. Cyborg, on completion of the bindings (successfully or otherwise), calls Novas `POST /os-server-external-events` API with:

```json
{
    "events": [
        {
            "name": "accelerator-request-bound",
            "tag": $device_profile_name,
            "server_uuid": $instance_uuid,
            "status": "completed" # or "failed"
        },
        ...
    ]
}
```

8. The Nova compute manager waits for the notification, subject to the timeout mentioned in Section Other deployer impact. It then calls the Cyborg REST API `GET /v2/accelerator_requests?instance=<uuid>&bind_state=resolved`.

9. The Nova virt driver uses the attach handles returned from the Cyborg call to compose PCI passthrough devices into the VMs definition.

10. If there is any error after binding has been initiated, Nova must unbind the relevant ARQs by calling Cyborg API. It may then retry on another host or delete the (unbound) ARQs for the instance.
As an end user of Cyborg, you'll use Cyborg to create and manage accelerators with either tools or the API directly.

### 3.1 Tools for using Cyborg

Information on the commands available through Cyborg's Command Line Interface (CLI) can be found in this section of documentation.

#### 3.1.1 Command-Line Interface Reference

**cyborg-status**

**Synopsis**

```
cyborg-status <category> <command> [<args>]
```

**Description**

`cyborg-status` is a tool that provides routines for checking the status of a Cyborg deployment.

**Options**

The standard pattern for executing a `cyborg-status` command is:

```
cyborg-status <category> <command> [<args>]
```

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

```
cyborg-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:

```
cyborg-status upgrade
```

These sections describe the available categories and arguments for `cyborg-status`.

### Upgrade

**cyborg-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**

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All upgrade readiness checks passed successfully and there is nothing to do.</td>
</tr>
<tr>
<td>1</td>
<td>At least one check encountered an issue and requires further investigation. This is considered a warning but the upgrade may be OK.</td>
</tr>
<tr>
<td>2</td>
<td>There was an upgrade status check failure that needs to be investigated. This should be considered something that stops an upgrade.</td>
</tr>
<tr>
<td>255</td>
<td>An unexpected error occurred.</td>
</tr>
</tbody>
</table>

**History of Checks**

**2.0.0 (Stein)**

- Placeholder to be filled in with checks as they are added in Stein.

### 3.2 Using the API

Following the Ussuri release, every Cyborg deployment should have the following endpoints:

- `/` - list of available versions
- `/v2` - the version 2 of the Acceleration API, it uses microversions
- `/v2.0` - same API as v2, except uses microversions

The following guide concentrates on documenting the v2 API, please note that the v2.0 is the first microversion of the v2 API and are also covered by this guide.

- **Cyborg API Reference**: The complete reference for the accelerator API, including all methods and request / response parameters and their meaning.
- **REST API Version History**: The Cyborg API evolves over time through Microversions. This provides the history of all those changes. Consider it a what's new in the Cyborg API.
4.1 Contributor Documentation

Contributing to Cybrog gives you the power to help add features, fix bugs, enhance documentation, and increase testing. Contributions of any type are valuable, and part of what keeps the project going. Here are a list of resources to get your started.

4.1.1 Basic Information

So You Want to Contribute

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

Below will cover the more project specific information you need to get started with Cyborg.

Communication

We use the #openstack-cyborg channel on the OFTC IRC network.

The weekly meetings happen in this channel. You can find the meeting times, previous meeting logs and proposed meeting agendas at Cyborg Team Meeting Page.

The openstack-discuss mailing list is another way we make announcement and discuss some topics in public. Cyborg related discussions usually start with tag: {cyborg}. Theopenstack-discussArchives

Contacting the Core Team

The core reviewers of Cyborg and their emails are listed in Cyborg core team.
New Feature Planning

To propose or plan new features, we add a new story in the Cyborg Launchpad and/or propose a specification in the cyborg-specs repository.

Task Tracking

We track our tasks in the Launchpad.

We also have our planned goals and schedule for each OpenStack official release. If you are a new contributor who is looking for tasks, please take a look. We have tasks from low-hanging-fruit to middle, and advanced levels. Development Release Schedule If you're looking for some smaller, easier work item to pick up and get started on, ask in the IRC meeting.

Reporting a Bug

You found an issue and want to make sure we are aware of it? You can do so on Launchpad. More info about Launchpad usage can be found on OpenStack docs page. But before you report a bug or register a new feature, we recommend you to check if it is already registered there. Otherwise, it might be a duplicated patch and will be abandoned. If you're still confused, as a more efficient way, we encourage you to go and ask PTL and cores in the IRC channel directly. Or mail PTL if IRC is not convenient for you.

Getting Your Patch Merged

To merge a patch, it must pass all voting Zuul checks and get two +2s from core reviewers. We strive to avoid scenarios where one person from a company or organization proposes a patch, and two other core reviewers from the same organization approve it to get it merged. In other words, at least one among the patch author and the two approving reviewers must be from another organization.

We are constantly striving to improve quality. Proposed patches must generally have unit tests and/or functional tests that cover the changes, and strive to improve code coverage.

Project Team Lead Duties

All common PTL duties are enumerated in the PTL guide.

4.1.2 Reviewing

- **API Microversions**: How the API is (micro)versioned and what you need to do when adding an API exposed feature that needs a new microversion.
- **Release Notes**: When we need a release note for a contribution.
- **DevStack Quick Start**: Guidelines for handling setup devstack
- **Driver Development Guide**: Get your driver development guide to contribute
API Microversions

Background

Cyborg uses a framework we call API Microversions for allowing changes to the API while preserving backward compatibility. The basic idea is that a user has to explicitly ask for their request to be treated with a particular version of the API. So breaking changes can be added to the API without breaking users who don’t specifically ask for it. This is done with an HTTP header `OpenStack-API-Version` which has as its value a string containing the name of the service, `accelerator`, and a monotonically increasing semantic version number starting from 2.0. The full form of the header takes the form:

```
OpenStack-API-Version: accelerator 2.0
```

If a user makes a request without specifying a version, they will get the `_MIN_VERSION_STRING` (defined in `cyborg/api/controllers/v2/versions.py`) as the default version. This value is currently 2.0 and is expected to remain so for quite a long time.

There is a special value `latest` which can be specified, which will allow a client to always receive the most recent version (`_MAX_VERSION_STRING` defined in `cyborg/api/controllers/v2/versions.py`) of API responses from the server.

**Warning:** The `latest` value is mostly meant for integration testing and would be dangerous to rely on in client code since Cyborg microversions are not following server and therefore backward compatibility is not guaranteed. Clients, like python-cyborgclient, should always require a specific microversion but limit what is acceptable to the version range that it understands at the time.

For full details please read the [Ussuri spec for microversions](https://docs.openstack.org/ussuri/server.md) and [Microversion Specification](https://docs.openstack.org/ussuri/server.md).

When do I need a new Microversion?

A microversion is needed when the contract to the user is changed. The user contract covers many kinds of information such as:

- the Request
  - the list of resource urls which exist on the accelerator
    Example: adding a new `accelerator_requests/{ID}/foo` which didn’t exist in a previous version of the code
  - the list of query parameters that are valid on urls
    Example: adding a new parameter `is_yellow` `accelerator_requests/{ID}?is_yellow=True`
  - the list of query parameter values for non free form fields
    Example: parameter `filter_by` takes a small set of constants/enums A, B, C. Adding support for new enum D.
  - new headers accepted on a request
  - the list of attributes and data structures accepted.
    Example: adding a new attribute description to the accelerator request body
• the Response
  – the list of attributes and data structures returned
    Example: adding a new attribute description to the output of accelerator_requests/{ID}
  – the allowed values of non free form fields
    Example: adding a new allowed state to accelerator_requests/{ID}
  – the list of status codes allowed for a particular request
    Example: an API previously could return 200, 400, 403, 404 and the change would make the API now also be allowed to return 409.
    See\(^2\) for the 400, 403, 404 and 415 cases.
  – new headers returned on a response.
  – changing a status code on a particular response.
    Example: changing the return code of an API from 501 to 400.

  **Note:** Fixing a bug so that a 400+ code is returned rather than a 500 or 503 does not require a microversion change. Its assumed that clients are not expected to handle a 500 or 503 response and therefore should not need to opt-in to microversion changes that fixes a 500 or 503 response from happening. According to the OpenStack API Working Group, a **500 Internal Server Error** should **not** be returned to the user for failures due to user error that can be fixed by changing the request on the client side. See\(^1\).

The following flow chart attempts to walk through the process of do we need a microversion.

\(^2\) The exception to not needing a microversion when returning a previously unspecified error code is the 400, 403, 404 and 415 cases. This is considered OK to return even if previously unspecified in the code since its implied given keystone authentication can fail with a 403 and API validation can fail with a 400 for invalid json request body. Request to url/resource that does not exist always fails with 404. Invalid content types are handled before API methods are called which results in a 415.

**Note:** When in doubt about whether or not a microversion is required for changing an error response code, consult the Cyborg team.

\(^1\) When fixing 500 errors that previously caused stack traces, try to map the new error into the existing set of errors that API call could previously return (400 if nothing else is appropriate). Changing the set of allowed status codes from a request is changing the contract, and should be part of a microversion (except in\(^2\)).

The reason why we are so strict on contract is that wed like application writers to be able to know, for sure, what the contract is at every microversion in Cyborg. If they do not, they will need to write conditional code in their application to handle ambiguities.

When in doubt, consider application authors. If it would work with no client side changes on both Cyborg versions, you probably dont need a microversion. If, on the other hand, there is any ambiguity, a microversion is probably needed.
Footnotes

When a microversion is not needed

A microversion is not needed in the following situation:

- the response
  - Changing the error message without changing the response code does not require a new microversion.
  - Removing an inapplicable HTTP header, for example, suppose the Retry-After HTTP header is being returned with a 4xx code. This header should only be returned with a 503 or 3xx response, so it may be removed without bumping the microversion.
  - An obvious regression bug in an admin-only API where the bug can still be fixed upstream on active stable branches. Admin-only APIs are less of a concern for interoperability and generally a regression in behavior can be dealt with as a bug fix when the documentation...
clearly shows the API behavior was unexpectedly regressed.

In Code

In cyborg/api/controllers/v2/versions.py we define some constants below:

- **BASE_VERSION**: value is 2 which is intended to be used as the Cyborg API version.
- **MINOR_0_INITIAL_VERSION**: value is 0 to be used as the initial value of microversion.
- **MINOR_X_Y**: Y is the change you want to make, X is the min version to support Y. For example, MINOR_1_PROJECT_ID means that the request project_id is supported from microversion 2.1.
- **MINOR_MAX_VERSION**: the max version, which equals to latest.
- **_MIN_VERSION_STRING**: the combination of BASE_VERSION and MINOR_0_INITIAL_VERSION, which means the min version of Cyborg API.
- **_MAX_VERSION_STRING** with the combination of BASE_VERSION and MINOR_MAX_VERSION, which means the max version of Cyborg API.

In cyborg/api/controllers/v2/utils.py, we define the check function of microversion.

For the example of `allow_project_id()` function, we compare the request version and the defined MINOR_1_PROJECT_ID to check whether the request is allowed. If the users request with the version which is lower than MINOR_1_PROJECT_ID, we will raise Request not acceptable exception to the user.

```python
def allow_project_id():
    # v2.1 added project_id for arq patch
    return api.request.version.minor >= versions.MINOR_1_PROJECT_ID
```

Adding a new API method

In the controller class:

```python
def my_api_method(self, req, id):
    if not utils.allow_project_id():
        raise exception.NotAcceptable(_("Request not acceptable. The minimal required API version should be %%(base)s.%%(opr)s") %
            {'base': versions.BASE_VERSION,
             'opr': versions.MINOR_1_PROJECT_ID})
```

This method would only be available if the caller had specified an OpenStack-API-Version of >= accelerator 2.1. If they had specified a lower version (or not specified it and received the default of accelerator 2.0) the server would respond with HTTP/406.
Other necessary changes

If you are adding a patch which adds a new microversion, it is necessary to add changes to other places which describe your change:

- Define `MINOR_{int}_**` in `cyborg/api/controllers/v2/versions.py`
- Update `MINOR_MAX_VERSION` to the defined `MINOR_{int}_**` in `cyborg/api/controllers/v2/versions.py`
- Add a verbose description of what changed in the new version to `cyborg/api/rest_api_version_history.rst`.
- Add a release note with a features section announcing the new or changed feature and the microversion.
- Update the expected versions in affected tests, for example in `cyborg.tests.unit.api.controllers.v2.test_arqs.TestARQsController#test_apply_patch_allow_project_id`.
- Make a new commit to python-cyborgclient and update corresponding files to enable the newly added microversion API.
- Update the API Reference documentation as appropriate. The source is located under `api-ref/source`.

If applicable, add functional sample tests under `cyborg_tempest_plugin/tests/api/`

Allocating a microversion

If you are adding a patch which adds a new microversion, it is necessary to allocate the next microversion number. The minor number of `_MAX_API_VERSION` will be incremented. This will also be the new microversion number for the API change. Developers may need over time to rebase their patch calculating a new version number as above based on the updated value of `_MAX_API_VERSION`.

Testing Microversioned API Methods

Testing a microversioned API method is very similar to a normal controller method test, you just need to add the OpenStack-API-Version header, for example:

```python
req = fakes.HTTPRequest.blank('/testable/url/endpoint')
req.headers = {'OpenStack-API-Version': 'accelerator 2.1'}
req.api_version_request = api_version.APIVersionRequest('2.1')
controller = controller.TestableController()
res = controller.index(req)
... assertions about the response ...
```

For many examples of testing, the canonical examples are in `cyborg.tests.unit.api.controllers.v2.test_arqs.TestARQsController#test_apply_patch_allow_project_id`.
Release Notes

What is reno?

Cyborg uses reno for providing release notes in-tree. That means that a patch can include a reno file or a series can have a follow-on change containing that file explaining what the impact is.

A reno file is a YAML file written in the releasenotes/notes tree which is generated using the reno tool this way:

```
$ tox -e venv -- reno new <name-your-file>
```

where usually <name-your-file> can be bp-<blueprint_name> for a blueprint or bug-XXXXXX for a bug fix.

Refer to the reno documentation for more information.

When a release note is needed

A release note is required anytime a reno section is needed. Below are some examples for each section. Any sections that would be blank should be left out of the note file entirely. If no section is needed, then you know you dont need to provide a release note :-) 

- **upgrade**
  - The patch has an UpgradeImpact tag
  - A DB change needs some deployer modification (like a migration)
  - A configuration option change (deprecation, removal or modified default)
  - some specific changes that have a DocImpact tag but require further action from an deployer perspective
  - any patch that requires an action from the deployer in general

- **security**
  - If the patch fixes a known vulnerability

- **features**
  - If the patch has an APIImpact tag
  - For Cyborg api and python-cyborgclient changes, if it adds or changes a new command, including adding new options to existing commands
  - a new accelerator driver is provided or an existing driver impacts the DriversSupport-Matrix

- **critical**
  - Bugfixes categorized as Critical in launchpad impacting users

- **fixes**
  - No clear definition of such bugfixes. Hairy long-standing bugs with high importance that have been fixed are good candidates though.
Three sections are left intentionally unexplained (prelude, issues and other). Those are targeted to be filled in close to the release time for providing details about the soon-ish release. Don’t use them unless you know exactly what you are doing.

**DevStack Quick Start**

**Create stack user (optional)**

Devstack should be run as a non-root user with sudo enabled (standard logins to cloud images such as ubuntu or cloud-user are usually fine).

You can quickly create a separate stack user to run DevStack with.

```bash
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
```

Since this user will be making many changes to your system, it should have sudo privileges:

```bash
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
```

```bash
$ sudo su - stack
```

**Download DevStack**

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

```bash
$ cd devstack
```

The `devstack` repo contains a script that installs OpenStack.

**Create local.conf file**

Create a `local.conf` file at the root of the devstack git repo.

**Host settings**

```bash
[[local|localrc]]
HOST_IP=YOUR_IP_CONFIG
SERVICE_HOST=$HOST_IP
DATABASE_TYPE=mysql
MYSQL_HOST=$HOST_IP
RABBIT_HOST=$HOST_IP
```

- Replace `YOUR_IP_CONFIG` with your host IP (e.g. 10.0.0.72 or localhost).
Password settings

```bash
# Passwords
DATABASE_PASSWORD=123
ADMIN_PASSWORD=123
MYSQL_PASSWORD=123
RABBIT_PASSWORD=123
SERVICE_PASSWORD=123
SERVICE_TOKEN=ADMIN
```

- Pre-set the passwords to prevent interactive prompts.

Enable services

```bash
#FIXED_RANGE=192.168.128.0/24
#IPV4_ADDRS_SAFE_TO_USE=192.168.128.0/24
#GIT_BASE=/opt/git
disable_service n-net
disable_service tempest
disable_service heat
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service n-novnc
enable_plugin cyborg https://opendev.org/openstack/cyborg
NOVA_VNC_ENABLED=True
NOVNCPROXY_URL="http://$SERVICE_HOST:6080/vnc_auto.html"
VNCSERVER_LISTEN=0.0.0.0
VNCSERVER_PROXYCLIENT_ADDRESS=$SERVICE_HOST
RECLONE=False
#enable Logging
LOGFILE=/opt/stack/logs/stack.sh.log
VERBOSE=True
LOG_COLOR=True
LOGDIR=/opt/stack/logs
```

- Uncomment GIT_BASE configuration if you have a local git repo
- enable_plugin cyborg will execute cyborg/devstack/plugin.sh and start cyborg service
- The devstack logs will appear in $LOGDIR

**Note:** If you got version conflicts, please set `PIP_UPGRADE` to `True` in local.conf
Multi-Node Lab

If you want to setup an OpenStack with cyborg in a realistic test configuration with multiple physical servers. Please ref\(^1\).

Cluster Controller

```bash
disable_service cyborg-agent
```

Compute Nodes

```bash
enable_service cyborg-agent
disable_service cyborg-api
disable_service cyborg-cond
```

- If you do not want to setup cyborg-agent on controller, you can disable it.
- You do not need to enable cyborg-api and cyborg-cond on compute nodes.

Testing with unmerged changes

To test with changes that have not been merged, the enable_plugin line can be modified to specify the branch/reference to be cloned.

```bash
enable_plugin cyborg https://review.opendev.org/openstack/cyborg.refs/changes/708728/1
```

the format is

```bash
enable_plugin <directory name> <git repo url> <change/revision>
```

Cell V2 Deployment

Compute node services must be mapped to a cell before they can be used. Cell V2 deployment, please ref\(^2\).

---

1 Openstack Multi-Node Lab Setup  
2 Openstack Cell V2 Deployment Guide
Run DevStack

$ ./stack.sh

This will take a 30-40 minutes, largely depending on the speed of your internet connection. Many git trees and packages will be installed during this process.

It will speed up your installation if you have a local GIT_BASE.

Use OpenStack

Command line

You can source openrc YOUR_USER YOUR_USER (e.g. source openrc admin admin) in your shell, and then use the openstack command line tool to manage your devstack.

Horizon

You can access horizon to experience the web interface to OpenStack, and manage vms, networks, volumes, and images from there.

References

Driver Development Guide

The goal of this document is to explain how to develop a new kind of Cyborg accelerator driver.

Note: Make sure you have installed Openstack environment using devstack before development.

4.1. Derive a new driver class

Imply the necessary interface, the list of interfaces is as follows:

class NewCyborgDriver(object):
    """Cyborg new accelerator driver."
    
    def discover(self):
        """Discover specific accelerator
        :return: list of cyborg.objects.driver_objects.driver_device.
        DriverDevice
        """
        pass

4.1. Contributor Documentation
Modify setup.cfg

Add the new driver map into file cyborg/setup.cfg:

```
[entry_points]
cyborg.accelerator.driver =
    intel_fpga_driver = cyborg.accelerator.drivers.fpga.intel.
        driver:IntelFPGADriver
cyborg.accelerator.drivers.example.
        driver:NewCyborgDriver
```

Reinstall and Test

Reinstall cyborg:

```
$ python setup.py develop
```

Restart cyborg-agent:

```
$ sudo systemctl restart devstack@cyborg-agent
```

4.2 REST API Version History

This documents the changes made to the REST API with every microversion change. The description for each version should be a verbose one which has enough information to be suitable for use in user documentation.

A user can specify a header in the API request:

```
OpenStack-API-Version: accelerator <microversion>
```

where <microversion> is any valid api microversion for this API.

If no version is specified then the API will behave as if a version request of v2.0 was requested.

4.2.1 2.0

This is the initial version of the v2 API which supports microversions.

4.2.2 2.1 (Maximum in Victoria)

Add `project_id` for Accelerator Requests PATCH API. `project_id` is used to control the operation of `arq` with different roles.
4.2.3 2.2

Changed device_profile_uuid to device_profile_name_or_uuid in Get One Device Profile API path, so support getting device profile by name (newly introduced) and uuid.

- GET /v2/device_profiles/{device_profile_name_or_uuid}
• search