Monasca Documentation

Release 7.0.1.dev6

['OpenStack Foundation']

Feb 03, 2022

CONTENTS

The monitoring requirements in OpenStack environments are vast, varied, and highly complex. Monascas project mission is to provide a monitoring-as-a-service solution that is multi-tenant, highly scalable, performant, and fault-tolerant. Monasca provides an extensible platform for advanced monitoring that can be used by both operators and tenants to gain operational insights about their infrastructure and applications.

Monasca uses REST APIs for high-speed metrics, logs processing and querying. It integrates a streaming alarm engine, a notification engine and an aggregation engine.

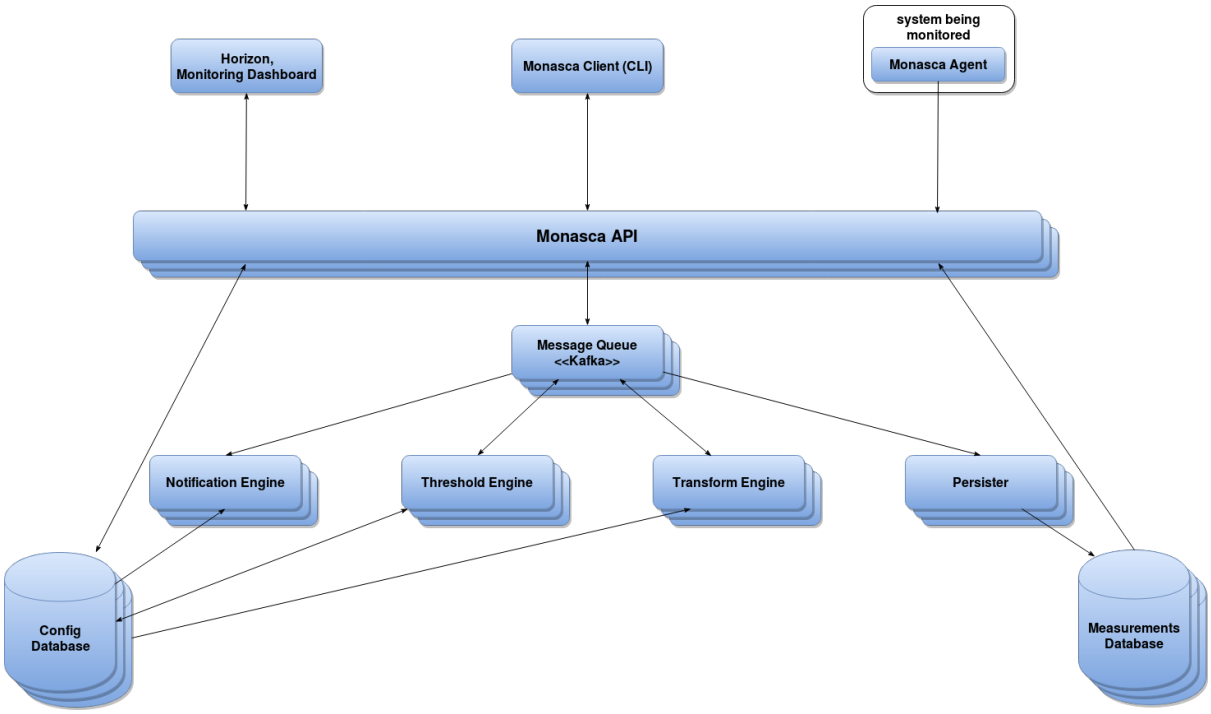
The use cases you can implement with Monasca are very diverse. Monasca follows a micro-services architecture, with several services split across multiple repositories. Each module is designed to provide a discrete service in the overall monitoring solution and can be deployed or omitted according to operators/customers needs.

CHAPTER

ONE

ARCHITECTURE

The following illustration provides an overview of Monascas metrics pipeline and the interaction of the involved components. For information on Monascas log pipeline, refer to this wiki page.



1.1 Repositories

- monasca-api: RESTful API for metrics, alarms, and notifications.
- monasca-agent: Agent for retrieving metrics data.
- monasca-persister: Writes metrics and alarm state transitions to a time-series database.
- monasca-thresh: Thresholding engine for computing thresholds on metrics and determining alarm states.
- monasca-notification: Pluggable notification engine for consuming alarm state transitions and sending notifications for alarms.
- monasca-transform: Aggregation engine based on Apache Spark.
- monasca-aggregator: Light-weight metrics aggregator.

Apart from sending requests directly to the API, the following tools are available for interacting with Monasca:

- Monasca Client: CLI and Python client.
- Horizon plugin: Plugin adding the monitoring panel to Horizon.
- Grafana app: Plugin for Grafana to view and configure alarm definitions, alarms, and notifications.

Libraries:

- monasca-common: Common code used in the Monasca components.
- monasca-statsd: StatsD-compatible library for sending metrics from instrumented applications.

Grafana integration:

- monasca-grafana-datasource: Multi-tenant Monasca data source for Grafana.
- grafana: Forked version of Grafana 4.1.2 with Keystone authentication added.

CHAPTER

TWO

FOR CONTRIBUTORS

2.1 Contribution documentation

2.1.1 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 Monasca.

Communication

For communicating with Monasca Team, you can reach out to us on *#openstack-monasca* IRC channel at OFTC.

We hold weekly team meetings in our IRC channel which is a good opportunity to ask questions, propose new features or just get in touch with the team.

You can also send us an email to the mailing list openstack-discuss@lists.openstack.org. Please use *[Monasca]* tag for easier thread filtering.

Contacting the Core Team

Name	IRC nick	Email
Martin Chacon Piza	chaconpiza	MartinDavid.ChaconPiza1@est.fujitsu.com
Witek Bedyk	witek	witold.bedyk@suse.com
Doug Szumski	dougsz	doug@stackhpc.com
Adrian Czarnecki	adriancz	adrian.czarnecki@ts.fujitsu.com
Joseph Davis	joadavis	joseph.davis@suse.com

New Feature Planning

Our process is meant to allow users, developers, and operators to express their desires for new features using Storyboard stories. The workflow is very simple:

- If something is clearly broken, submit a *bug report* in Storyboard.
- If you want to change or add a feature, submit a *story* in Storyboard.
- Monasca core reviewers may request that you submit a *specification* to gerrit to elaborate on the feature request.
- Significant features require *release notes* to be included when the code is merged.

Stories

New features can be proposed in Storyboard as new Story.

The initial story primarily needs to express the intent of the idea with enough details that it can be evaluated for compatibility with the project mission and whether or not the change requires a *specification*. It is *not* expected to contain all of the implementation details. If the feature is very simple and well understood by the team, then describe it simply. The story is then used to track all the related code reviews. Team members will request more information as needed.

Specifications

We use the monasca-specs repository for specification reviews. Specifications:

- Provide a review tool for collaborating on feedback and reviews for complex features.
- Collect team priorities.
- Serve as the basis for documenting the feature once implemented.
- Ensure that the overall impact on the system is considered.

Release Notes

The release notes for a patch should be included in the patch. If not, the release notes should be in a follow-on review.

If any of the following applies to the patch, a release note is required:

- The deployer needs to take an action when upgrading
- A new feature is implemented
- Plugin API function was removed or changed
- Current behavior is changed
- A new config option is added that the deployer should consider changing from the default
- A security bug is fixed
- Change may break previous versions of the client library(ies)

- Requirement changes are introduced for important libraries like oslo, six requests, etc.
- Deprecation period starts or code is purged

A release note is suggested if a long-standing or important bug is fixed. Otherwise, a release note is not required.

Task Tracking

We track our tasks in Storyboard

https://storyboard.openstack.org/#!/project_group/monasca

If youre looking for some smaller, easier work item to pick up and get started on, search for the *low-hanging-fruit* tag.

Kanban Board

Progress on implementation of important stories in Ussuri release is tracked in Monasca Board on StoryBoard.

Reporting a Bug

You found an issue and want to make sure we are aware of it? You can report them on Storyboard.

When filing a bug please remember to add the *bug* tag to the story. Please provide information on what the problem is, how to replicate it, any suggestions for fixing it, and a recommendation of the priority.

All open bugs can be found in this Worklist.

Getting Your Patch Merged

All changes proposed to Monasca requires at least one `Code-Review +2` votes from Monasca core reviewers before one of the core reviewers can approve patch by giving `Workflow +1` vote.

Reviews Prioritisation

Monasca project uses *Review-Priority* field in Gerrit to emphasize prioritized code changes.

Every developer can propose the changes which should be prioritized in weekly team meeting or in the mailing list. Any core reviewer, preferably from a different company, can confirm such proposed change by setting *Review-Priority* +1.

Prioritized changes can be listed in this dashboard.

Project Team Lead Duties

All common PTL duties are enumerated in the PTL guide.

2.1.2 Database Migrations

Monasca uses Alembic migrations to set up its configuration database. If you need to change the configuration databases schema, you need to create a migration to adjust the database accordingly, as follows:

```
cd monasca_api/db/
alembic revision
```

This will create a new skeleton revision for you to edit. You will find existing revisions to use for inspiration in the `/monasca_api/db/alembic/versions/` directory.

Measurement data stored in a Time Series database (such as InfluxDB) would be migrated to a new version using standard practice for a given TSDB.

2.1.3 Codebase documentation

Following section contains codebase documenation generated with, a little bit of assistance, sphinx.ext.autodoc.

Modules

The `monasca_api.api.alarm_definitions_api_v2` Module

class monasca_api.api.alarm_definitions_api_v2.**AlarmDefinitionsV2API**

Bases: `object`

on_delete(*req*, *res*, *alarm_definition_id*)

on_get(*req*, *res*, *alarm_definition_id*)

on_patch(*req*, *res*, *alarm_definition_id*)

on_post(*req*, *res*)

on_put(*req*, *res*, *alarm_definition_id*)

The `monasca_api.api.alarms_api_v2` Module

class monasca_api.api.alarms_api_v2.**AlarmsCountV2API**

Bases: `object`

on_get(*req*, *res*)

class monasca_api.api.alarms_api_v2.**AlarmsStateHistoryV2API**

Bases: `object`

on_get(*req*, *res*, *alarm_id*)

class monasca_api.api.alarms_api_v2.**AlarmsV2API**

Bases: object

on_delete(*req*, *res*, *alarm_id*)

on_get(*req*, *res*, *alarm_id*)

on_patch(*req*, *res*, *alarm_id*)

on_put(*req*, *res*, *alarm_id*)

The monasca_api.api.core.log.exceptions Module

exception monasca_api.api.core.log.exceptions.**HTTPUnprocessableEntity**(*message*, ***kwargs*)

Bases: falcon.http_error.HTTPError

HTTPUnprocessableEntity http error.

HTTPError that comes with 422 Unprocessable Entity status

Argument message(str) - meaningful description of what caused an error

Argument kwargs - any other option defined in falcon.HTTPError

code

description

headers

link

status

title

The monasca_api.api.core.log.log_publisher Module

exception monasca_api.api.core.log.log_publisher.**InvalidMessageException**

Bases: Exception

class monasca_api.api.core.log.log_publisher.**LogPublisher**

Bases: object

Publishes log data to Kafka

LogPublisher is able to send single message to multiple configured topic. It uses following configuration written in conf file

```
[log_publisher]
topics = 'logs'
kafka_url = 'localhost:8900'
```

Note: Uses monasca_common.kafka.producer.KafkaProducer to ship logs to kafka. For more details see monasca_common github repository.

send_message(*messages*)

Sends message to each configured topic.

Note: Falsy messages (i.e. empty) are not shipped to kafka

See also

- monasca_log_api.common.model.Envelope
- _is_message_valid()

Parameters **messages** (*dict* | *list*) instance (or instances) of log envelope

The monasca_api.api.core.log.model Module

class monasca_api.api.core.log.model.**Envelope**(*log*, *meta*)

Bases: dict

property creation_time

property log

property meta

classmethod new_envelope(*log*, *tenant_id*, *region*, *dimensions=None*)

Creates new log envelope

Log envelope is combined ouf of following properties

- log - dict
- creation_time - timestamp
- meta - meta block

Example output json would like this:

```
{
    "log": {
      "message": "Some message",
      "dimensions": {
        "hostname": "devstack"
      }
    },
    "creation_time": 1447834886,
    "meta": {
      "tenantId": "e4bd29509eda473092d32aadfee3e7b1",
      "region": "pl"
    }
}
```

Parameters

- **log** (*dict*) original log element (containing message and other params
- **tenant_id** (*str*) tenant id to be put in meta field
- **region** (*str*) region to be put in meta field

- **dimensions** (*dict*) additional dimensions to be appended to log object dimensions

exception monasca_api.api.core.log.model.**LogEnvelopeException**
Bases: Exception

monasca_api.api.core.log.model.**serialize_envelope**(*envelope*)
Returns json representation of an envelope.

Returns json object of envelope

Return type six.text_type

The monasca_api.api.core.log.validation Module

monasca_api.api.core.log.validation.**APPLICATION_TYPE_CONSTRAINTS** = **{'MAX_LENGTH': 255, 'PATTERN': re.compile('^[a-zA-Z0-9_.\\-]+$')}**
Application type constraint used in validation.

See Validations.validate_application_type()

monasca_api.api.core.log.validation.**DIMENSION_NAME_CONSTRAINTS** = **{'MAX_LENGTH': 255, 'PATTERN': re.compile('[^><={}(), \'";&]+$')}**
Constraint for name of single dimension.

See Validations.validate_dimensions()

monasca_api.api.core.log.validation.**DIMENSION_VALUE_CONSTRAINTS** = **{'MAX_LENGTH': 255}**
Constraint for value of single dimension.

See Validations.validate_dimensions()

monasca_api.api.core.log.validation.**validate_application_type**(*application_type=None*)
Validates application type.

Validation wont take place if application_type is None. For details see: *APPLICATION_TYPE_CONSTRAINTS*

Parameters **application_type** (*str*) application type

monasca_api.api.core.log.validation.**validate_content_type**(*req*, *allowed*)
Validates content type.

Method validates request against correct content type.

If content-type cannot be established (i.e. header is missing), falcon.HTTPMissingHeader is thrown. If content-type is not **application/json** or **text/plain**, falcon.HTTPUnsupportedMediaType is thrown.

Parameters

- **req** (*falcon.Request*) current request
- **allowed** (*iterable*) allowed content type

Exception falcon.HTTPMissingHeader

Exception falcon.HTTPUnsupportedMediaType

monasca_api.api.core.log.validation.**validate_cross_tenant**(*tenant_id*, *cross_tenant_id*, *roles*)

monasca_api.api.core.log.validation.**validate_dimensions**(*dimensions*)

Validates dimensions type.

Empty dimensions are not being validated. For details see:

Parameters **dimensions** (*dict*) dimensions to validate

- *DIMENSION_NAME_CONSTRAINTS*
- *DIMENSION_VALUE_CONSTRAINTS*

monasca_api.api.core.log.validation.**validate_is_delegate**(*roles*)

monasca_api.api.core.log.validation.**validate_log_message**(*log_object*)

Validates log property.

Log property should have message property.

Args: log_object (dict): log property

monasca_api.api.core.log.validation.**validate_payload_size**(*req*)

Validates payload size.

Method validates sent payload size. It expects that http header **Content-Length** is present. If it does not, method raises `falcon.HTTPLengthRequired`. Otherwise values is being compared with

```
[service]
max_log_size = 1048576
```

max_log_size refers to the maximum allowed content length. If it is exceeded `falcon.HTTPRequestEntityTooLarge` is thrown.

Parameters **req** (*falcon.Request*) current request

Exception `falcon.HTTPLengthRequired`

Exception `falcon.HTTPRequestEntityTooLarge`

The `monasca_api.api.core.request` Module

class monasca_api.api.core.request.**Request**(*env*, *options=None*)

Bases: `falcon.request.Request`

Variation of falcon.Request with context

Following class enhances `falcon.Request` with `context.RequestContext`.

can(*action*, *target=None*)

content_type

context

property cross_project_id

Returns project-id (tenant-id) found in query params.

This particular project-id is later on identified as cross-project-id

Returns project-id

Return type str

env

is_websocket

property limit

Returns LIMIT query param value.

limit is not required query param. In case it is not found, py:data:.constants.PAGE_LIMIT value is returned.

Returns value of limit query param or default value

Return type int

Raises *exceptions.HTTPUnprocessableEntityError* if limit is not valid integer

method

options

path

property project_id

Returns project-id (tenant-id)

Returns project-id

Return type str

query_string

property roles

Returns roles associated with user

Returns users roles

Return type list

stream

uri_template

property user_id

Returns user-id

Returns user-id

Return type str

The monasca_api.api.core.request_context Module

class monasca_api.api.core.request_context.**RequestContext**(*auth_token=None*, *user_id=None*, *project_id=None*, *domain_id=None*, *user_domain_id=None*, *project_domain_id=None*, *is_admin=False*, *read_only=False*, *show_deleted=False*, *request_id=None*, *resource_uuid=None*, *overwrite=True*, *roles=None*, *user_name=None*, *project_name=None*, *domain_name=None*, *user_domain_name=None*, *project_domain_name=None*, *is_admin_project=True*, *service_token=None*, *service_user_id=None*, *service_user_name=None*, *service_user_domain_id=None*, *service_user_domain_name=None*, *service_project_id=None*, *service_project_name=None*, *service_project_domain_id=None*, *service_project_domain_name=None*, *service_roles=None*, *global_request_id=None*, *system_scope=None*)

Bases: oslo_context.context.RequestContext

RequestContext.

RequestContext is customized version of :py:class:oslo_context.context.RequestContext.

can(*action*, *target=None*)

The monasca_api.api.healthcheck_api Module

class monasca_api.api.healthcheck_api.**HealthCheckApi**

Bases: object

HealthCheck Api.

HealthCheckApi server information regarding health of the Api.

on_get(*req*, *res*)

Complex healthcheck report on GET

Returns complex report regarding API health and all dependent services

Parameters

- **req** (*falcon.Request*) current request
- **res** (*falcon.Response*) current response

on_head(*req*, *res*)

Simple healthcheck report on HEAD.

In opposite to *HealthCheckApi.on_get()*, this method is supposed to execute ASAP to inform user that API is up and running.

Parameters

- **req** (*falcon.Request*) current request
- **res** (*falcon.Response*) current response

The monasca_api.api.logs_api Module

class monasca_api.api.logs_api.**LogsApi**

Bases: object

Logs API.

Logs API acts as RESTful endpoint accepting messages contains collected log entries from the system. Works as gateway for any further processing for accepted data.

on_get(*req*, *res*)

Queries logs matching specified dimension values.

Performs queries on the underlying log storage against a time range and set of dimension values.

Parameters

- **req** current request
- **res** current response

on_post(*req*, *res*)

Accepts sent logs as text or json.

Accepts logs sent to resource which should be sent to kafka queue.

Parameters

- **req** current request
- **res** current response

property version

The monasca_api.api.metrics_api_v2 Module

class monasca_api.api.metrics_api_v2.**DimensionNamesV2API**
Bases: object

on_get(*req*, *res*)

class monasca_api.api.metrics_api_v2.**DimensionValuesV2API**
Bases: object

on_get(*req*, *res*)

class monasca_api.api.metrics_api_v2.**MetricsMeasurementsV2API**
Bases: object

on_get(*req*, *res*)

class monasca_api.api.metrics_api_v2.**MetricsNamesV2API**
Bases: object

on_get(*req*, *res*)

class monasca_api.api.metrics_api_v2.**MetricsStatisticsV2API**
Bases: object

on_get(*req*, *res*)

class monasca_api.api.metrics_api_v2.**MetricsV2API**
Bases: object

on_get(*req*, *res*)

on_post(*req*, *res*)

The monasca_api.api.notifications_api_v2 Module

class monasca_api.api.notifications_api_v2.**NotificationsV2API**
Bases: object

on_delete(*req*, *res*, *notification_method_id*)

on_get(*req*, *res*, *notification_method_id*)

on_patch(*req*, *res*, *notification_method_id*)

on_post(*req*, *res*)

on_put(*req*, *res*, *notification_method_id*)

The monasca_api.api.notificationstype_api_v2 Module

class monasca_api.api.notificationstype_api_v2.**NotificationsTypeV2API**

Bases: object

on_delete(*req*, *res*)

on_get(*req*, *res*)

on_patch(*req*, *res*)

on_post(*req*, *res*)

on_put(*req*, *res*)

The monasca_api.api.server Module

monasca_api.api.server.**get_wsgi_app**(*config_base_path=None*, ***kwargs*)

monasca_api.api.server.**launch**(*conf*)

monasca_api.api.server.**launch_log_api**(*app*)

monasca_api.api.server.**launch_metrics_api**(*app*)

The monasca_api.api.versions_api Module

class monasca_api.api.versions_api.**VersionsAPI**

Bases: object

on_get(*req*, *res*, *id*)

The monasca_api.cmd.monasca_db Module

CLI interface for monasca database management.

monasca_api.cmd.monasca_db.**add_command_parsers**(*subparsers*)

monasca_api.cmd.monasca_db.**do_detect_revision**()

monasca_api.cmd.monasca_db.**do_fingerprint**()

monasca_api.cmd.monasca_db.**do_stamp**()

monasca_api.cmd.monasca_db.**do_upgrade**()

monasca_api.cmd.monasca_db.**do_version**()

monasca_api.cmd.monasca_db.**main**()

The monasca_api.cmd.status Module

CLI interface for monasca status commands. https://governance.openstack.org/tc/goals/stein/upgrade-checkers.html

class monasca_api.cmd.status.**Checks**
Bases: oslo_upgradecheck.upgradecheck.UpgradeCommands

Various upgrade checks should be added as separate methods in this class and added to _upgrade_checks tuple.

monasca_api.cmd.status.**main**()

The monasca_api.common.messaging.exceptions Module

exception monasca_api.common.messaging.exceptions.**MessageQueueException**
Bases: Exception

The monasca_api.common.messaging.fake_publisher Module

class monasca_api.common.messaging.fake_publisher.**FakePublisher**(*topic*)
Bases: *monasca_api.common.messaging.publisher.Publisher*

send_message(*message*)

The monasca_api.common.messaging.kafka_publisher Module

class monasca_api.common.messaging.kafka_publisher.**KafkaPublisher**(*topic*)
Bases: *monasca_api.common.messaging.publisher.Publisher*

close()

send_message(*message*)

The monasca_api.common.messaging.message_formats.metrics Module

monasca_api.common.messaging.message_formats.metrics.**transform**(*metrics*, *tenant_id*, *region*)

The monasca_api.common.messaging.publisher Module

class monasca_api.common.messaging.publisher.**Publisher**
Bases: object

abstract send_message(*message*)

The monasca_api.common.policy.i18n Module

oslo.i18n integration module.

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

monasca_api.common.policy.i18n.**get_available_languages**()

monasca_api.common.policy.i18n.**translate**(*value*, *user_locale*)

The monasca_api.common.policy.policy_engine Module

class monasca_api.common.policy.policy_engine.**IsAdminCheck**(*kind*, *match*)
Bases: oslo_policy._checks.Check

An explicit check for is_admin.

monasca_api.common.policy.policy_engine.**authorize**(*context*, *action*, *target*, *do_raise=True*)
Verify that the action is valid on the target in this context.

Parameters

- **context** (*object*) monasca project context
- **action** (*str*) String representing the action to be checked. This should be colon separated for clarity.
- **target** (*dict*) Dictionary representing the object of the action for object creation. This should be a dictionary representing the location of the object e.g. `{'project_id': 'context.project_id'}`
- **do_raise** (*bool*) if True (the default), raises PolicyNotAuthorized, if False returns False

Returns returns a non-False value (not necessarily True) if authorized, and the False if not authorized and do_raise if False

Raises **oslo_policy.policy.PolicyNotAuthorized** if verification fails

monasca_api.common.policy.policy_engine.**check_is_admin**(*context*)
Check if roles contains admin role according to policy settings.

monasca_api.common.policy.policy_engine.**get_enforcer**()

monasca_api.common.policy.policy_engine.**get_rules**()

monasca_api.common.policy.policy_engine.**init**(*policy_file=None*, *rules=None*, *default_rule=None*, *use_conf=True*)
Init an Enforcer class.

Parameters

- **policy_file** Custom policy file to use, if none is specified, *CONF.policy_file* will be used.
- **rules** Default dictionary / Rules to use. It will be considered just in the first instantiation.

- **default_rule** Default rule to use, CONF.default_rule will be used if none is specified.
- **use_conf** Whether to load rules from config file.

monasca_api.common.policy.policy_engine.**register_rules**(*enforcer*)
Register default policy rules.

monasca_api.common.policy.policy_engine.**reset**()
Reset Enforcer class.

monasca_api.common.policy.policy_engine.**set_rules**(*rules*, *overwrite=True*, *use_conf=False*)
Set rules based on the provided dict of rules.

Note: Used in tests only.

Parameters

- **rules** New rules to use. It should be an instance of dict
- **overwrite** Whether to overwrite current rules or update them with the new rules.
- **use_conf** Whether to reload rules from config file.

monasca_api.common.policy.policy_engine.**verify_deprecated_policy**(*old_policy*, *new_policy*, *default_rule*, *context*)
Check the rule of the deprecated policy action

If the current rule of the deprecated policy action is set to a non-default value, then a warning message is logged stating that the new policy action should be used to dictate permissions as the old policy action is being deprecated.

Parameters

- **old_policy** policy action that is being deprecated
- **new_policy** policy action that is replacing old_policy
- **default_rule** the old_policy action default rule value
- **context** the monasca context

The monasca_api.common.repositories.alarm_definitions_repository Module

class monasca_api.common.repositories.alarm_definitions_repository.**AlarmDefinitionsRepository**
Bases: object

abstract create_alarm_definition(*tenant_id*, *name*, *expression*, *sub_expr_list*, *description*, *severity*, *match_by*, *alarm_actions*, *undetermined_actions*, *ok_action*)

abstract delete_alarm_definition(*tenant_id*, *alarm_definition_id*)

abstract get_alarm_definition(*tenant_id*, *id*)

abstract get_alarm_definitions(*tenant_id*, *name*, *dimensions*, *severity*, *sort_by*, *offset*, *limit*)

abstract get_alarm_metrics(*tenant_id*, *alarm_definition_id*)

abstract get_sub_alarm_definitions(*alarm_definition_id*)

abstract get_sub_alarms(*tenant_id*, *alarm_definition_id*)

abstract update_or_patch_alarm_definition(*tenant_id*, *id*, *name*, *expression*, *sub_expr_list*, *actions_enabled*, *description*, *alarm_actions*, *ok_actions*, *undetermined_actions*, *match_by*, *severity*, *patch*)

The monasca_api.common.repositories.alarms_repository Module

class monasca_api.common.repositories.alarms_repository.**AlarmsRepository**
Bases: object

abstract delete_alarm(*tenant_id*, *id*)

abstract get_alarm(*tenant_id*, *id*)

abstract get_alarm_metrics(*alarm_id*)

abstract get_alarms(*tenant_id*, *query_parms*, *offset*, *limit*)

abstract get_alarms_count(*tenant_id*, *query_parms*, *offset*, *limit*)

abstract get_sub_alarms(*tenant_id*, *alarm_id*)

abstract update_alarm(*tenant_id*, *alarm_id*, *state*, *lifecycle_state*, *link*)

The monasca_api.common.repositories.cassandra.metrics_repository Module

The monasca_api.common.repositories.constants Module

The monasca_api.common.repositories.exceptions Module

exception monasca_api.common.repositories.exceptions.**AlreadyExistsException**
Bases: *monasca_common.repositories.exceptions.RepositoryException*

exception monasca_api.common.repositories.exceptions.**DoesNotExistException**
Bases: *monasca_common.repositories.exceptions.RepositoryException*

exception monasca_api.common.repositories.exceptions.**InvalidUpdateException**
Bases: *monasca_common.repositories.exceptions.RepositoryException*

exception monasca_api.common.repositories.exceptions.**MultipleMetricsException**
Bases: *monasca_common.repositories.exceptions.RepositoryException*

exception monasca_api.common.repositories.exceptions.**RepositoryException**
Bases: Exception

exception
monasca_api.common.repositories.exceptions.**UnsupportedDriverException**
Bases: Exception

The monasca_api.common.repositories.fake.metrics_repository Module

class
monasca_api.common.repositories.fake.metrics_repository.**MetricsRepository**
Bases: *monasca_api.common.repositories.metrics_repository.AbstractMetricsRepository*

list_metrics(*tenant_id*, *name*, *dimensions*, *offset*, *limit*)

The monasca_api.common.repositories.influxdb.metrics_repository Module

The monasca_api.common.repositories.metrics_repository Module

class
monasca_api.common.repositories.metrics_repository.**AbstractMetricsRepository**
Bases: object

MULTIPLE_METRICS_MESSAGE = "Found multiple metrics matching metric name and dimensions. Please refine your search criteria using a unique metric name or additional dimensions. Alternatively, you may specify 'merge_metrics=True' as a query parameter to combine all metrics matching search criteria into a single series."

abstract alarm_history(*tenant_id*, *alarm_id_list*, *offset*, *limit*, *start_timestamp*, *end_timestamp*)

abstract static check_status()

abstract list_metric_names(*tenant_id*, *region*, *dimensions*)

abstract list_metrics(*tenant_id*, *region*, *name*, *dimensions*, *offset*, *limit*)

abstract measurement_list(*tenant_id*, *region*, *name*, *dimensions*, *start_timestamp*, *end_timestamp*, *offset*, *limit*, *merge_metrics_flag*, *group_by*)

abstract metrics_statistics(*tenant_id*, *region*, *name*, *dimensions*, *start_timestamp*, *end_timestamp*, *statistics*, *period*, *offset*, *limit*, *merge_metrics_flag*, *group_by*)

The monasca_api.common.repositories.model.sub_alarm_definition Module

class monasca_api.common.repositories.model.sub_alarm_definition.**SubAlarmDefinition**(*row=Non*
sub_expr=

Bases: object

Holds sub alarm definition

Used for comparing sub alarm definitions for equality.

property expression
Build the entire expressions as a string with spaces.

same_key_fields(*other*)

The monasca_api.common.repositories.notification_method_type_repository Module

class monasca_api.common.repositories.notification_method_type_repository.**NotificationMethodTypeRepository**
Bases: object

abstract list_notification_method_types()

The monasca_api.common.repositories.notifications_repository Module

class monasca_api.common.repositories.notifications_repository.**NotificationsRepository**
Bases: object

abstract create_notification(*tenant_id*, *name*, *notification_type*, *address*, *period*)

abstract delete_notification(*tenant_id*, *notification_id*)

abstract list_notification(*tenant_id*, *notification_id*)

abstract list_notifications(*tenant_id*, *sort_by*, *offset*, *limit*)

abstract update_notification(*notification_id*, *tenant_id*, *name*, *notification_type*, *address*, *period*)

The monasca_api.common.repositories.sqla.alarm_definitions_repository Module

class monasca_api.common.repositories.sqla.alarm_definitions_repository.**AlarmDefinitionsRepository**
Bases: *monasca_api.common.repositories.sqla.sql_repository.SQLRepository*, *monasca_api.common.repositories.alarm_definitions_repository.AlarmDefinitionsRepository*

create_alarm_definition(***kwargs*)

delete_alarm_definition(***kwargs*)

get_alarm_definition(***kwargs*)

get_alarm_definitions(***kwargs*)

get_alarm_metrics(***kwargs*)

get_sub_alarm_definitions(***kwargs*)

get_sub_alarms(***kwargs*)

update_or_patch_alarm_definition(***kwargs*)

The monasca_api.common.repositories.sqla.alarms_repository Module

class monasca_api.common.repositories.sqla.alarms_repository.**AlarmsRepository**

Bases: *monasca_api.common.repositories.sqla.sql_repository.SQLRepository*, *monasca_api.common.repositories.alarms_repository.AlarmsRepository*

delete_alarm(***kwargs*)

get_alarm(***kwargs*)

get_alarm_definition(***kwargs*)

get_alarm_metrics(***kwargs*)

get_alarms(***kwargs*)

get_alarms_count(***kwargs*)

get_sub_alarms(***kwargs*)

update_alarm(***kwargs*)

The monasca_api.common.repositories.sqla.models Module

monasca_api.common.repositories.sqla.models.**create_a_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_aa_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_ad_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_am_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_md_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_mdd_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_mde_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_nm_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_nmt_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_sa_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_sad_model**(*metadata=None*)

monasca_api.common.repositories.sqla.models.**create_sadd_model**(*metadata=None*)

class monasca_api.common.repositories.sqla.models.**field_sort**(*column*, *fields*)

Bases: sqlalchemy.sql.elements.ColumnElement

column = None

fields = []

name = 'field_sort'

monasca_api.common.repositories.sqla.models.**get_all_metadata**()

Return metadata for full data model

This metadata is used for autogenerating a complete (works on an empty database) schema migration. To ensure this mechanism keeps working please invoke any new model creation methods you add from this function as well.

class monasca_api.common.repositories.sqla.models.**group_concat**(*columns*, *separator=','*, *order_by=None*)

Bases: sqlalchemy.sql.elements.ColumnElement

columns = []

name = 'group_concat'

order_by = None

separator = ','

The monasca_api.common.repositories.sqla.notification_method_type_repository Module

class monasca_api.common.repositories.sqla.notification_method_type_repository.**NotificationMethodTypeRepository**

Bases: *monasca_api.common.repositories.sqla.sql_repository.SQLRepository*, *monasca_api.common.repositories.notification_method_type_repository.NotificationMethodTypeRepository*

list_notification_method_types(***kwargs*)

The monasca_api.common.repositories.sqla.notifications_repository Module

class monasca_api.common.repositories.sqla.notifications_repository.**NotificationsRepository**

Bases: *monasca_api.common.repositories.sqla.sql_repository.SQLRepository*, *monasca_api.common.repositories.notifications_repository.NotificationsRepository*

create_notification(*tenant_id*, *name*, *notification_type*, *address*, *period*)

delete_notification(***kwargs*)

find_notification_by_name(***kwargs*)

list_notification(***kwargs*)

list_notifications(***kwargs*)

update_notification(***kwargs*)

The monasca_api.common.repositories.sqla.sql_repository Module

class monasca_api.common.repositories.sqla.sql_repository.**SQLRepository**

Bases: object

monasca_api.common.repositories.sqla.sql_repository.**create_context_manager**(*connection=None*)

Create a database context manager object.

Parameters **connection** The database connection string

monasca_api.common.repositories.sqla.sql_repository.**get_engine**(*use_slave=False*, *connection=None*)

Get a database engine object.

Parameters

- **use_slave** Whether to use the slave connection
- **connection** The database connection string

monasca_api.common.repositories.sqla.sql_repository.**sql_try_catch_block**(*fun*)

The monasca_api.common.rest.exceptions Module

exception monasca_api.common.rest.exceptions.**DataConversionException**

Bases: Exception

Exception thrown if data transformation fails

:py:class‘.DataConversionException‘ may be thrown if data was impossible to transform into target representation according to content_type classifier.

exception monasca_api.common.rest.exceptions.**UnreadableContentError**

Bases: OSError

Exception thrown if reading data fails

:py:class‘.UnreadableContentError‘ may be thrown if data was impossible to read from input

exception monasca_api.common.rest.exceptions.**UnsupportedContentTypeException**

Bases: Exception

Exception thrown if content type is not supported.

The monasca_api.common.rest.utils Module

monasca_api.common.rest.utils.**as_json**(*data*, ***kwargs*)

Writes data as json.

Parameters

- **data** (*dict*) data to convert to json
- **kwargs** (*kwargs*) kwargs for json dumps

Returns json string

Return type str

monasca_api.common.rest.utils.**from_json**(*data*, ***kwargs*)

Reads data from json str.

Parameters

- **data** (*str*) data to read
- **kwargs** (*kwargs*) kwargs for json loads

Returns read data

Return type dict

monasca_api.common.rest.utils.**read_body**(*payload*, *content_type='application/json'*)
Reads HTTP payload according to given content_type.

Function is capable of reading from payload stream. Read data is then processed according to content_type.

Note: Content-Type is validated. It means that if read_body body is not capable of reading data in requested type, it will throw an exception.

If read data was empty method will return false boolean value to indicate that.

Note: There is no transformation if content type is equal to text/plain. What has been read is returned.

Parameters

- **payload** (*stream*) payload to read, payload should have read method
- **content_type** (*str*) payload content type, default to application/json

Returns read data, returned type depends on content_type or False if empty

Exception UnreadableBody - in case of any failure when reading data

The monasca_api.config Module

monasca_api.config.**get_config_files**()
Get the possible configuration files accepted by oslo.config

This also includes the deprecated ones

monasca_api.config.**parse_args**(*argv=None*)
Loads application configuration.

Loads entire application configuration just once.

The monasca_api.db.alembic.env Module

monasca_api.db.alembic.env.**run_migrations_online**()
Run migrations in online mode.

In this scenario we need to create an Engine and associate a connection with the context.

The monasca_api.db.alembic.versions.00597b5c8325_initial Module

Initial migration for full schema (Git revision 00597b5c8325664c2c534625525f59232d243d66).

Revision ID: 00597b5c8325 Revises: N/A Create Date: 2018-04-12 09:09:48.212206

monasca_api.db.alembic.versions.00597b5c8325_initial.**downgrade**()

monasca_api.db.alembic.versions.00597b5c8325_initial.**upgrade**()

The monasca_api.db.alembic.versions.0cce983d957a_deterministic_alarms Module

Add flag for deterministic alarms (Git revision 0cce983d957a3d780b6d206ad25df1271a812b4a).

Revision ID: 0cce983d957a Revises: 00597b5c8325 Create Date: 2018-04-23 13:57:32.951669

monasca_api.db.alembic.versions.0cce983d957a_deterministic_alarms.**downgrade**()

monasca_api.db.alembic.versions.0cce983d957a_deterministic_alarms.**upgrade**()

The monasca_api.db.alembic.versions.26083b298bb7_remove_builtin_notification_types Module

Remove builtin notification types

Revision ID: 26083b298bb7 Revises: f69cb3152a76 Create Date: 2018-09-18 13:52:02.170226

monasca_api.db.alembic.versions.26083b298bb7_remove_builtin_notification_types.**downgrade**()

monasca_api.db.alembic.versions.26083b298bb7_remove_builtin_notification_types.**upgrade**()

The monasca_api.db.alembic.versions.30181b42434b_remove_event_and_migration_tables Module

Remove event related tables and schema_migrations table (git revision 30181b42434bdde0c40abd086e903600b24e9684)

Revision ID: 30181b42434b Revises: c2f85438d6f3 Create Date: 2018-04-24 09:54:50.024470

monasca_api.db.alembic.versions.30181b42434b_remove_event_and_migration_tables.**downgrade**()

monasca_api.db.alembic.versions.30181b42434b_remove_event_and_migration_tables.**upgrade**()

The monasca_api.db.alembic.versions.6b2b88f3cab4_add_sub_alarm_state Module

Add sub alarm state (Git revision 6b2b88f3cab46cd442369b22da3624611b871169)

Revision ID: 6b2b88f3cab4 Revises: 30181b42434b Create Date: 2018-04-24 12:16:15.812274

monasca_api.db.alembic.versions.6b2b88f3cab4_add_sub_alarm_state.**downgrade**()

monasca_api.db.alembic.versions.6b2b88f3cab4_add_sub_alarm_state.**upgrade**()

The monasca_api.db.alembic.versions.8781a256f0c1_add_inhibited_and_silenced_to_alarms Module

Add inhibited and silenced to alarms (Git revision 8781a256f0c19662b81f04b014e2b769e625bd6b)

Revision ID: 8781a256f0c1 Revises: d8b801498850 Create Date: 2018-04-24 13:16:04.157977

monasca_api.db.alembic.versions.8781a256f0c1_add_inhibited_and_silenced_to_alarms.**downgrade**

monasca_api.db.alembic.versions.8781a256f0c1_add_inhibited_and_silenced_to_alarms.**upgrade**()

The monasca_api.db.alembic.versions.c2f85438d6f3_period_notifications Module

Add period to notifications (Git revision c2f85438d6f3b0fd2e1f86d84eee6e9967025eb6)

Revision ID: c2f85438d6f3 Revises: 0cce983d957a Create Date: 2018-04-23 14:47:49.413502

monasca_api.db.alembic.versions.c2f85438d6f3_period_notifications.**downgrade**()

monasca_api.db.alembic.versions.c2f85438d6f3_period_notifications.**upgrade**()

The monasca_api.db.alembic.versions.d8b801498850_remove_stream_action_types Module

Remove stream action types (Git revision d8b80149885016ede0ee403cf9bb07f9b7253297)

Revision ID: d8b801498850 Revises: 6b2b88f3cab4 Create Date: 2018-04-24 12:53:02.342849

monasca_api.db.alembic.versions.d8b801498850_remove_stream_action_types.**downgrade**()

monasca_api.db.alembic.versions.d8b801498850_remove_stream_action_types.**upgrade**()

The monasca_api.db.alembic.versions.f69cb3152a76_remove_inhibited_silenced_from_alarms Module

Remove inhibited and silenced from alarms (Git revision f69cb3152a76e7c586dcc9a03600d1d4ed32c4e6)

Revision ID: f69cb3152a76 Revises: 8781a256f0c1 Create Date: 2018-04-24 13:16:04.157977

monasca_api.db.alembic.versions.f69cb3152a76_remove_inhibited_silenced_from_alarms.**downgrade**()

monasca_api.db.alembic.versions.f69cb3152a76_remove_inhibited_silenced_from_alarms.**upgrade**()

The monasca_api.db.fingerprint Module

class monasca_api.db.fingerprint.**Fingerprint**(*engine*)

Bases: object

The monasca_api.expression_parser.alarm_expr_parser Module

class monasca_api.expression_parser.alarm_expr_parser.**AlarmExprParser**(*expr*)

Bases: object

property sub_expr_list

class monasca_api.expression_parser.alarm_expr_parser.**AndSubExpr**(*tokens*)

Bases: *monasca_api.expression_parser.alarm_expr_parser.BinaryOp*

Expand later as needed.

class monasca_api.expression_parser.alarm_expr_parser.**BinaryOp**(*tokens*)

Bases: object

property operands_list

class monasca_api.expression_parser.alarm_expr_parser.**OrSubExpr**(*tokens*)
Bases: *monasca_api.expression_parser.alarm_expr_parser.BinaryOp*

Expand later as needed.

class monasca_api.expression_parser.alarm_expr_parser.**SubExpr**(*tokens*)
Bases: object

property deterministic

property dimensions
Get the dimensions.

property dimensions_as_list
Get the dimensions as a list.

property dimensions_str
Get all the dimensions as a single comma delimited string.

property fmtd_sub_expr_str
Get the entire sub expressions as a string with spaces.

property func
Get the function as it appears in the orig expression.

property id
Get the id used to identify this sub expression in the repo.

property metric_name
Get the metric name as it appears in the orig expression.

property normalized_func
Get the function upper-cased.

property normalized_metric_name
Get the metric name lower-cased.

property normalized_operator
Get the operator as one of LT, GT, LTE, or GTE.

property operands_list
Get this sub expression as a list.

property operator
Get the operator.

property period
Get the period. Default is 60 seconds.

property periods
Get the periods. Default is 1.

property threshold
Get the threshold value.

monasca_api.expression_parser.alarm_expr_parser.**main**()
Used for development and testing.

monasca_api.expression_parser.alarm_expr_parser.**periodValidation**(*instr*, *loc*, *tokens*)

monasca_api.expression_parser.alarm_expr_parser.**periodsValidation**(*instr*, *loc*, *tokens*)

The monasca_api.hacking.checks Module

The monasca_api.healthcheck.alarms_db_check Module

class monasca_api.healthcheck.alarms_db_check.**AlarmsDbHealthCheck**
Bases: *monasca_api.healthcheck.base.BaseHealthCheck*, *monasca_api.common.repositories.sqla.sql_repository.SQLRepository*

Evaluates alarm db health

Healthcheck verifies if: * database is up and running, it is possible to establish connection * sample sql query can be executed

If following conditions are met health check return healthy status. Otherwise unhealthy status is returned with explanation.

check_db_status()

health_check()
Evaluate health of given service

The monasca_api.healthcheck.base Module

class monasca_api.healthcheck.base.**BaseHealthCheck**
Bases: object

Abstract class implemented by the monasca-api healthcheck classes

abstract health_check()
Evaluate health of given service

class monasca_api.healthcheck.base.**CheckResult**(*healthy*, *message*)
Bases: *monasca_api.healthcheck.base.CheckResult*

Result for the health check

healthy - boolean message - string

The monasca_api.healthcheck.kafka_check Module

class monasca_api.healthcheck.kafka_check.**KafkaHealthCheck**
Bases: *monasca_api.healthcheck.base.BaseHealthCheck*

Evaluates kafka health

Healthcheck verifies if:

- kafka server is up and running
- there is a configured topic in kafka

If following conditions are met health check returns healthy status. Otherwise unhealthy status is returned with message.

Note: Healthcheck checks 3 type of topics given in configuration: metrics_topic, events_topic and alarm_state_transition_topic.

health_check()
Evaluate health of given service

The monasca_api.healthcheck.keystone_protocol Module

class monasca_api.healthcheck.keystone_protocol.**SkippingAuthProtocol**(*app*, *conf*)
Bases: keystonemiddleware.auth_token.AuthProtocol

SkippingAuthProtocol to reach healthcheck endpoint

Because healthcheck endpoints exists as endpoint, it is hidden behind keystone filter thus a request needs to authenticated before it is reached.

Note: SkippingAuthProtocol is lean customization of keystonemiddleware.auth_token.AuthProtocol that disables keystone communication if request is meant to reach healthcheck

process_request(*request*)
Process request.

Evaluate the headers in a request and attempt to authenticate the request. If authenticated then additional headers are added to the request for use by applications. If not authenticated the request will be rejected or marked unauthenticated depending on configuration.

monasca_api.healthcheck.keystone_protocol.**filter_factory**(*global_conf*, ***local_conf*)
Return factory function for *SkippingAuthProtocol*

Parameters

- **global_conf** global configuration
- **local_conf** local configuration

Returns factory function

Return type function

The monasca_api.healthcheck.metrics_db_check Module

class monasca_api.healthcheck.metrics_db_check.**MetricsDbCheck**
Bases: *monasca_api.healthcheck.base.BaseHealthCheck*

Evaluates metrics db health

Healthcheck what type of database is used (InfluxDB, Cassandra) and provide health according to the given db.

If following conditions are met health check return healthy status. Otherwise unhealthy status is returned with explanation.

health_check()
Evaluate health of given service

The monasca_api.healthchecks Module

class monasca_api.healthchecks.**HealthChecks**

Bases: *monasca_api.api.healthcheck_api.HealthCheckApi*

CACHE_CONTROL = ['must-revalidate', 'no-cache', 'no-store']

HEALTHY_CODE_GET = '200 OK'

HEALTHY_CODE_HEAD = '204 No Content'

NOT_HEALTHY_CODE = '503 Service Unavailable'

on_get(*req*, *res*)

Complex healthcheck report on GET

Returns complex report regarding API health and all dependent services

Parameters

- **req** (*falcon.Request*) current request
- **res** (*falcon.Response*) current response

on_head(*req*, *res*)

Simple healthcheck report on HEAD.

In opposite to *HealthCheckApi.on_get()*, this method is supposed to execute ASAP to inform user that API is up and running.

Parameters

- **req** (*falcon.Request*) current request
- **res** (*falcon.Response*) current response

The monasca_api.policies.alarms Module

monasca_api.policies.alarms.**list_rules**()

The monasca_api.policies.delegate Module

monasca_api.policies.delegate.**list_rules**()

The monasca_api.policies.healthcheck Module

monasca_api.policies.healthcheck.**list_rules**()

The monasca_api.policies.logs Module

monasca_api.policies.logs.**list_rules**()

The monasca_api.policies.metrics Module

monasca_api.policies.metrics.**list_rules**()

The monasca_api.policies.notifications Module

monasca_api.policies.notifications.**list_rules**()

The monasca_api.policies.versions Module

monasca_api.policies.versions.**list_rules**()

The monasca_api.v2.common.bulk_processor Module

class monasca_api.v2.common.bulk_processor.**BulkProcessor**(*logs_in_counter=None*, *logs_rejected_counter=None*)

Bases: *monasca_api.api.core.log.log_publisher.LogPublisher*

BulkProcessor for effective log processing and publishing.

BulkProcessor is customized version of monasca_log_api.app.base.log_publisher.LogPublisher that utilizes processing of bulk request inside single loop.

send_message(*logs*, *global_dimensions=None*, *log_tenant_id=None*)

Sends bulk package to kafka

Parameters

- **logs** (*list*) received logs
- **global_dimensions** (*dict*) global dimensions for each log
- **log_tenant_id** (*str*) tenant who sent logs

The monasca_api.v2.common.exceptions Module

exception monasca_api.v2.common.exceptions.**HTTPBadRequestError**(*title*, *description*, ***kwargs*)

Bases: falcon.http_error.HTTPError

code

description

headers

link

status

title

exception monasca_api.v2.common.exceptions.**HTTPUnprocessableEntityError**(*title*, *description*, ***kwargs*)

Bases: falcon.http_error.HTTPError

code

description

headers

link

status

title

The monasca_api.v2.common.helpers Module

monasca_api.v2.common.helpers.**get_global_dimensions**(*request_body*)
Get the top level dimensions in the HTTP request body.

monasca_api.v2.common.helpers.**get_logs**(*request_body*)
Get the logs in the HTTP request body.

monasca_api.v2.common.helpers.**read_json_msg_body**(*req*)
Read the json_msg from the http request body and return them as JSON.

Parameters **req** HTTP request object.

Returns Returns the metrics as a JSON object.

Raises **falcon.HTTPBadRequest**

The monasca_api.v2.common.schemas.alarm_definition_request_body_schema Module

monasca_api.v2.common.schemas.alarm_definition_request_body_schema.**validate**(*msg*, *require_all=False*)

monasca_api.v2.common.schemas.alarm_definition_request_body_schema.**validate_action_list**(*notif*, *action*,

monasca_api.v2.common.schemas.alarm_definition_request_body_schema.**validate_alarm_action_lis**

monasca_api.v2.common.schemas.alarm_definition_request_body_schema.**validate_ok_action_list**(

monasca_api.v2.common.schemas.alarm_definition_request_body_schema.**validate_undetermined_act**

The monasca_api.v2.common.schemas.alarm_update_schema Module

monasca_api.v2.common.schemas.alarm_update_schema.**validate**(*msg*)

The monasca_api.v2.common.schemas.exceptions Module

exception monasca_api.v2.common.schemas.exceptions.**ValidationException**
Bases: Exception

The monasca_api.v2.common.schemas.notifications_request_body_schema Module

monasca_api.v2.common.schemas.notifications_request_body_schema.**parse_and_validate**(*msg*, *valid_perio* *re-* *quire_all=*

The monasca_api.v2.common.utils Module

monasca_api.v2.common.utils.**date_handler**(*obj*)

The monasca_api.v2.common.validation Module

monasca_api.v2.common.validation.**validate_alarm_definition_severity**(*severity*)

monasca_api.v2.common.validation.**validate_alarm_state**(*state*)

monasca_api.v2.common.validation.**validate_email_address**(*email*)

monasca_api.v2.common.validation.**validate_severity_query**(*severity_str*)

monasca_api.v2.common.validation.**validate_sort_by**(*sort_by_list*, *allowed_sort_by*)

The monasca_api.v2.reference.alarm_definitions Module

class monasca_api.v2.reference.alarm_definitions.**AlarmDefinitions**
Bases: *monasca_api.api.alarm_definitions_api_v2.AlarmDefinitionsV2API*, *monasca_api.v2.reference.alarming.Alarming*

on_delete(***kwargs*)

on_get(***kwargs*)

on_patch(***kwargs*)

on_post(***kwargs*)

on_put(***kwargs*)

monasca_api.v2.reference.alarm_definitions.**get_comma_separated_str_as_list**(*comma_separated_st*

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_actions_enabled**(*alarm_*
re-
quired=
re-
turn_n

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_alarm_actions**(*alarm_de*
re-
turn_none

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_description**(*alarm_defin*
re-
turn_none=

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_expression**(*alarm_definit*
re-
turn_none=F

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_match_by**(*alarm_definition*
re-
turn_none=Fals

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_name**(*alarm_definition*,
re-
turn_none=False)

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_severity**(*alarm_definition*
re-
turn_none=Fals

monasca_api.v2.reference.alarm_definitions.**get_query_alarm_definition_undetermined_actions**(*a*
r
t

monasca_api.v2.reference.alarm_definitions.**get_query_ok_actions**(*alarm_definition*,
re-
turn_none=False)

monasca_api.v2.reference.alarm_definitions.**is_definition_deterministic**(*expression*)

Evaluates if found expression is deterministic or not.

In order to do that expression is parsed into sub expressions. Each sub expression needs to be deterministic in order for entity expression to be such.

Otherwise expression is non-deterministic.

Parameters **expression** (*str*) expression to be evaluated

Returns true/false

Return type bool

The monasca_api.v2.reference.alarming Module

class monasca_api.v2.reference.alarming.**Alarming**
Bases: object

Super class for Alarms and AlarmDefinitions.

Shared attributes and methods for classes Alarms and AlarmDefinitions.

send_event(*message_queue*, *event_msg*)

The monasca_api.v2.reference.alarms Module

class monasca_api.v2.reference.alarms.**Alarms**
Bases: *monasca_api.api.alarms_api_v2.AlarmsV2API*, *monasca_api.v2.reference.alarming.Alarming*

on_delete(***kwargs*)

on_get(***kwargs*)

on_patch(***kwargs*)

on_put(***kwargs*)

class monasca_api.v2.reference.alarms.**AlarmsCount**
Bases: *monasca_api.api.alarms_api_v2.AlarmsCountV2API*, *monasca_api.v2.reference.alarming.Alarming*

on_get(***kwargs*)

class monasca_api.v2.reference.alarms.**AlarmsStateHistory**
Bases: *monasca_api.api.alarms_api_v2.AlarmsStateHistoryV2API*, *monasca_api.v2.reference.alarming.Alarming*

on_get(***kwargs*)

The monasca_api.v2.reference.helpers Module

monasca_api.v2.reference.helpers.**add_links_to_resource**(*resource*, *uri*, *rel='self'*)
Adds links to the given resource dictionary.

Parameters

- **resource** the resource dictionary you wish to add links.
- **uri** the http request.uri.

monasca_api.v2.reference.helpers.**add_links_to_resource_list**(*resourcelist*, *uri*)
Adds links to the given resource dictionary list.

Parameters

- **resourcelist** the list of resources you wish to add links.
- **uri** the http request.uri.

monasca_api.v2.reference.helpers.**build_base_uri**(*parsed_uri*)

monasca_api.v2.reference.helpers.**create_alarms_count_next_link**(*uri*, *offset*, *limit*)

monasca_api.v2.reference.helpers.**from_json**(*req*)
Read the json_msg from the http request body and return them as JSON.

Parameters **req** HTTP request object.

Returns Returns the metrics as a JSON object.

Raises **falcon.HTTPBadRequest**

monasca_api.v2.reference.helpers.**get_link**(*uri*, *resource_id*, *rel='self'*)
Returns a link dictionary containing href, and rel.

Parameters

- **uri** the http request.uri.
- **resource_id** the id of the resource

monasca_api.v2.reference.helpers.**get_query_dimensions**(*req*, *param_key='dimensions'*)
Gets and parses the query param dimensions.

Parameters

- **req** HTTP request object.
- **dimensions_param** param name for dimensions, default=dimensions

Returns Returns the dimensions as a JSON object

Raises **falcon.HTTPBadRequest** If dimensions are malformed.

monasca_api.v2.reference.helpers.**get_query_endtime_timestamp**(*req*, *required=True*)

monasca_api.v2.reference.helpers.**get_query_group_by**(*req*)

monasca_api.v2.reference.helpers.**get_query_name**(*req*, *name_required=False*)
Returns the query param name if supplied.

Parameters **req** HTTP request object.

monasca_api.v2.reference.helpers.**get_query_param**(*req*, *param_name*, *required=False*, *default_val=None*)

monasca_api.v2.reference.helpers.**get_query_period**(*req*)

monasca_api.v2.reference.helpers.**get_query_starttime_timestamp**(*req*, *required=True*)

monasca_api.v2.reference.helpers.**get_query_statistics**(*req*)

monasca_api.v2.reference.helpers.**get_x_tenant_or_tenant_id**(*http_request*, *delegate_authorized_rules_list*)

monasca_api.v2.reference.helpers.**paginate**(*resource*, *uri*, *limit*)

monasca_api.v2.reference.helpers.**paginate_alarming**(*resource*, *uri*, *limit*)

monasca_api.v2.reference.helpers.**paginate_dimension_values**(*dimvals*, *uri*, *offset*, *limit*)

monasca_api.v2.reference.helpers.**paginate_measurements**(*measurements*, *uri*, *limit*)

monasca_api.v2.reference.helpers.**paginate_statistics**(*statistics*, *uri*, *limit*)

monasca_api.v2.reference.helpers.**paginate_with_no_id**(*dictionary_list*, *uri*, *offset*, *limit*)
This method is to paginate a list of dictionaries with no id in it. For example, metric name list, directory name list and directory value list.

monasca_api.v2.reference.helpers.**raise_not_found_exception**(*resource_name*, *resource_id*, *tenant_id*)
Provides exception for not found requests (update, delete, list).

Parameters

- **resource_name** the name of the resource.
- **resource_id** id of the resource.
- **tenant_id** id of the tenant

monasca_api.v2.reference.helpers.**str_2_bool**(*s*)

monasca_api.v2.reference.helpers.**to_json**(*data*)
Converts data to JSON string.

Parameters **data** (*dict*) data to be transformed to JSON

Returns JSON string

Return type str

Raises Exception

monasca_api.v2.reference.helpers.**validate_authorization**(*http_request*, *authorized_rules_list*)
Validates whether is authorized according to provided policy rules list.

If authorization fails, 401 is thrown with appropriate description. Additionally response specifies WWW-Authenticate header with Token value challenging the client to use different token (the one with different set of roles which can access the service).

monasca_api.v2.reference.helpers.**validate_json_content_type**(*req*)

monasca_api.v2.reference.helpers.**validate_payload_size**(*content_length*)
Validates payload size.

Method validates payload size, this method used req.content_length to determinate payload size

[service] max_log_size = 1048576

max_log_size refers to the maximum allowed content length. If it is exceeded `falcon.HTTPRequestEntityTooLarge` is thrown.

Parameters **content_length** size of payload

Exception `falcon.HTTPLengthRequired`

Exception `falcon.HTTPRequestEntityTooLarge`

monasca_api.v2.reference.helpers.**validate_query_dimensions**(*dimensions*)
Validates the query param dimensions.

Parameters **dimensions** Query param dimensions.

Raises **falcon.HTTPBadRequest** If dimensions are not valid.

monasca_api.v2.reference.helpers.**validate_query_name**(*name*)
Validates the query param name.

Parameters **name** Query param name.

Raises **falcon.HTTPBadRequest** If name is not valid.

monasca_api.v2.reference.helpers.**validate_start_end_timestamps**(*start_timestamp*, *end_timestamp=None*)

The monasca_api.v2.reference.logs Module

class monasca_api.v2.reference.logs.**Logs**

Bases: *monasca_api.api.logs_api.LogsApi*

SUPPORTED_CONTENT_TYPES = {'application/json'}

VERSION = 'v2.0'

on_post(*req*, *res*)

Accepts sent logs as text or json.

Accepts logs sent to resource which should be sent to kafka queue.

Parameters

- **req** current request
- **res** current response

process_on_post_request(*req*, *res*)

The monasca_api.v2.reference.metrics Module

class monasca_api.v2.reference.metrics.**DimensionNames**

Bases: *monasca_api.api.metrics_api_v2.DimensionNamesV2API*

on_get(***kwargs*)

class monasca_api.v2.reference.metrics.**DimensionValues**

Bases: *monasca_api.api.metrics_api_v2.DimensionValuesV2API*

on_get(***kwargs*)

class monasca_api.v2.reference.metrics.**Metrics**

Bases: *monasca_api.api.metrics_api_v2.MetricsV2API*

on_get(***kwargs*)

on_post(***kwargs*)

class monasca_api.v2.reference.metrics.**MetricsMeasurements**

Bases: *monasca_api.api.metrics_api_v2.MetricsMeasurementsV2API*

on_get(***kwargs*)

class monasca_api.v2.reference.metrics.**MetricsNames**

Bases: *monasca_api.api.metrics_api_v2.MetricsNamesV2API*

on_get(***kwargs*)

class monasca_api.v2.reference.metrics.**MetricsStatistics**

Bases: *monasca_api.api.metrics_api_v2.MetricsStatisticsV2API*

on_get(***kwargs*)

monasca_api.v2.reference.metrics.**get_merge_metrics_flag**(*req*)
Return the value of the optional metrics_flag

Returns False if merge_metrics parameter is not supplied or is not a string that evaluates to True, otherwise True

The monasca_api.v2.reference.notifications Module

class monasca_api.v2.reference.notifications.**Notifications**
Bases: *monasca_api.api.notifications_api_v2.NotificationsV2API*

on_delete(***kwargs*)

on_get(***kwargs*)

on_patch(***kwargs*)

on_post(***kwargs*)

on_put(***kwargs*)

The monasca_api.v2.reference.notificationstype Module

class monasca_api.v2.reference.notificationstype.**NotificationsType**
Bases: *monasca_api.api.notificationstype_api_v2.NotificationsTypeV2API*

on_get(***kwargs*)

The monasca_api.v2.reference.resource Module

monasca_api.v2.reference.resource.**resource_try_catch_block**(*fun*)

The monasca_api.v2.reference.version_2_0 Module

class monasca_api.v2.reference.version_2_0.**Version2**
Bases: object

on_get(*req*, *res*)

The monasca_api.v2.reference.versions Module

class monasca_api.v2.reference.versions.**Versions**
Bases: *monasca_api.api.versions_api.VersionsAPI*

on_get(*req*, *res*, *version_id=None*)

The monasca_api.version Module

CHAPTER

THREE

FOR OPERATORS

3.1 Administrating

3.1.1 Administration guide

Schema Setup

For setting up the Monasca configuration database, we provide `monasca_db`, an Alembic based database migration tool. Historically, the schema for the configuration database was created by a SQL script. This SQL was changed a couple of times, so `monasca_db` comes with a mechanism to detect the SQL script revision being used to create it and stamp the database with the matching Alembic revision.

Setting up a new database

If you are deploying Monasca from scratch, database setup is quite straightforward:

1. Create a database and configure access credentials with `ALL PRIVILEGES` permission level on it in the Monasca API configuration files `[database]` section.
2. Run schema migrations: `monasca_db upgrade`. It will run all migrations up to and including the most recent one (`head`) unless a revision to migrate to is explicitly specified.

Upgrading Existing Database from Legacy Schema

If you have been running an older version of Monasca, you can attempt to identify and stamp its database schema:

```
monasca_db stamp --from-fingerprint
```

This command will generate a unique fingerprint for the database schema in question and match that fingerprint with an in-code map of fingerprints to database schema revisions. This should work for all official (shipped as part of the `monasca-api` repository) schema scripts. If you used a custom third-party schema script to set up the database, it may not be listed and youll get an error message similar to this one (the fingerprint hash will vary):

```
Schema fingerprint 3d45493070e3b8e6fc492d2369e51423ca4cc1ac does not match␣
↪any known legacy revision.
```

If this happens to you, please create a Storyboard story against the openstack/monasca-api project. Provide the following alongside the story:

1. A copy of or pointer to the schema SQL script being used to set up the database.
2. The fingerprint shown in the error message.
3. The output of `monasca_db fingerprint --raw`.

Time Series Databases Setup

Enabling InfluxDB Time Series Index in existing deployments

If enabling TSI on an existing InfluxDB install please follow the instructions for migrating existing data here: https://docs.influxdata.com/influxdb/v1.7/administration/upgrading/#upgrading-influxdb-1-3-1-4-no-tsi-preview-to-1-7-x-tsi-enabled

Database Per Tenant

It is envisaged that separate database per tenant will be the default behaviour in a future release of Monasca. Not only would it make queries faster for tenants, it would also allow administrators to define retention policy per tenancy. To enable this, set *influxdb.db_per_tenant* to *True* in *monasca-{api,persister}* config (it defaults to *False* at the moment if not set).

To migrate existing data to database per tenant, refer to README.rst under the following URL which also contains the Python script to facilitate migration: https://opendev.org/openstack/monasca-persister/src/branch/master/monasca_persister/tools/db-per-tenant/

3.2 Configuration

- *Sample Config Files*

3.2.1 Command Line Interface

monasca (python-monascaclient)

This is the main command line interface for working with the Monasca services, including retrieving metrics from storage.

See the https://docs.openstack.org/python-monascaclient/latest/ for details.

monasca_db

CLI for Monasca database management.

```
usage: api [-h] [--config-dir DIR] [--config-file PATH] [--version]
           {fingerprint,detect-revision,stamp,upgrade,version} ...
```

monasca-status

CLI for checking the status of Monasca.

Use the command *monasca-status upgrade check* to check the readiness of the system for an upgrade.

Return Codes

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

History

Introduced in the Stein cycle as part of the OpenStack Community wide goal. https://governance.openstack.org/tc/goals/stein/upgrade-checkers.html

3.2.2 Samples

The following sections show sample configuration files for monasca-api and related utilities. These are generated from the code (apart from the samples for logging and paster) and reflect the current state of code in the monasca-api repository.

Sample Configuration For Application

This sample configuration can also be viewed in monasca-api.conf.sample.

```
[DEFAULT]

#
# From monasca_api
#

#
# Region that API is running in
#  (string value)
#
```

(continues on next page)

(continued from previous page)

```
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#region = RegionOne

# Warning: Failed to format sample for valid_notification_periods
# type object 'int' has no attribute '_formatter'

#
# Enable Metrics api endpoints (boolean value)
#enable_metrics_api = true

#
# Enable Logs api endpoints (boolean value)
#enable_logs_api = false

#
# From oslo.log
#

# If set to true, the logging level will be set to DEBUG instead of the␣
↪default
# INFO level (boolean value)
# Note: This option can be changed without restarting.
#debug = false

# 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) (string value)
# Note: This option can be changed without restarting.
# Deprecated group/name - [DEFAULT]/log_config
#log_config_append = <None>

# Defines the format string for %%(asctime)s in log records. Default:
# %(default)s . This option is ignored if log_config_append is set (string
# value)
#log_date_format = %Y-%m-%d %H:%M:%S

# (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 (string value)
# Deprecated group/name - [DEFAULT]/logfile
#log_file = <None>
```

(continues on next page)

(continued from previous page)

```
# (Optional) The base directory used for relative log_file  paths. This option
# is ignored if log_config_append is set (string value)
# Deprecated group/name - [DEFAULT]/logdir
#log_dir = <None>

# 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
# (boolean value)
#watch_log_file = 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 (boolean value)
#use_syslog = 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 (boolean value)
#use_journal = false

# Syslog facility to receive log lines. This option is ignored if
# log_config_append is set (string value)
#syslog_log_facility = LOG_USER

# Use JSON formatting for logging. This option is ignored if log_config_append
# is set (boolean value)
#use_json = false

# Log output to standard error. This option is ignored if log_config_append is
# set (boolean value)
#use_stderr = false

# Log output to Windows Event Log (boolean value)
#use_eventlog = false

# The amount of time before the log files are rotated. This option is ignored
# unless log_rotation_type is set to "interval" (integer value)
#log_rotate_interval = 1

# 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 (string
# value)
# Possible values:
```

(continues on next page)

(continued from previous page)

```
# Seconds - <No description provided>
# Minutes - <No description provided>
# Hours - <No description provided>
# Days - <No description provided>
# Weekday - <No description provided>
# Midnight - <No description provided>
#log_rotate_interval_type = days

# Maximum number of rotated log files (integer value)
#max_logfile_count = 30

# Log file maximum size in MB. This option is ignored if "log_rotation_type"␣
↪is
# not set to "size" (integer value)
#max_logfile_size_mb = 200

# Log rotation type (string value)
# Possible values:
# interval - Rotate logs at predefined time intervals.
# size - Rotate logs once they reach a predefined size.
# none - Do not rotate log files.
#log_rotation_type = none

# Format string to use for log messages with context. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_context_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [%(request_id)s %(user_identity)s] %(instance)s
↪%(message)s

# Format string to use for log messages when context is undefined. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_default_format_string = %(asctime)s.%(msecs)03d %(process)d
↪%(levelname)s %(name)s [-] %(instance)s%(message)s

# Additional data to append to log message when logging level for the message
# is DEBUG. Used by oslo_log.formatters.ContextFormatter (string value)
#logging_debug_format_suffix = %(funcName)s %(pathname)s:%(lineno)d

# Prefix each line of exception output with this format. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_exception_prefix = %(asctime)s.%(msecs)03d %(process)d ERROR
↪%(name)s %(instance)s

# Defines the format string for %(user_identity)s that is used in
# logging_context_format_string. Used by oslo_log.formatters.ContextFormatter
# (string value)
#logging_user_identity_format = %(user)s %(tenant)s %(domain)s %(user_
↪domain)s %(project_domain)s
```

(continues on next page)

(continued from previous page)

```
# List of package logging levels in logger=LEVEL pairs. This option is ignored
# if log_config_append is set (list value)
#default_log_levels = 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

# Enables or disables publication of error events (boolean value)
#publish_errors = false

# The format for an instance that is passed with the log message (string␣
↪value)
#instance_format = "[instance: %(uuid)s] "

# The format for an instance UUID that is passed with the log message (string
# value)
#instance_uuid_format = "[instance: %(uuid)s] "

# Interval, number of seconds, of log rate limiting (integer value)
#rate_limit_interval = 0

# Maximum number of logged messages per rate_limit_interval (integer value)
#rate_limit_burst = 0

# 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 (string
# value)
#rate_limit_except_level = CRITICAL

# Enables or disables fatal status of deprecations (boolean value)
#fatal_deprecations = false


[cassandra]

#
# From monasca_api
#

#
# Comma separated list of Cassandra node IP addresses
#  (list value)
#contact_points = 127.0.0.1

#
```

(continues on next page)

(continued from previous page)

```
# Cassandra port number
#  (port value)
# Minimum value: 0
# Maximum value: 65535
#port = 9042

#
# keyspace where metric are stored
#  (string value)
#keyspace = monasca

#
# Cassandra user for monasca-api service
#  (string value)
#user =

#
# Cassandra user password for monasca-api service
#  (string value)
#password =

#
# Cassandra timeout in seconds when creating a new connection
#  (integer value)
#connection_timeout = 5

#
# Cassandra local data center name
#  (string value)
#local_data_center =


[database]

#
# From oslo.db
#

# If True, SQLite uses synchronous mode (boolean value)
#sqlite_synchronous = true

# The back end to use for the database (string value)
# Deprecated group/name - [DEFAULT]/db_backend
#backend = sqlalchemy

# The SQLAlchemy connection string to use to connect to the database (string
# value)
# Deprecated group/name - [DEFAULT]/sql_connection
# Deprecated group/name - [DATABASE]/sql_connection
```

(continues on next page)

(continued from previous page)

```
# Deprecated group/name - [sql]/connection
#connection = <None>

# The SQLAlchemy connection string to use to connect to the slave database
# (string value)
#slave_connection = <None>

# 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=
# (string value)
#mysql_sql_mode = TRADITIONAL

# If True, transparently enables support for handling MySQL Cluster (NDB)
# (boolean value)
#mysql_enable_ndb = false

# 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 (integer value)
#connection_recycle_time = 3600

# Maximum number of SQL connections to keep open in a pool. Setting a value of
# 0 indicates no limit (integer value)
#max_pool_size = 5

# Maximum number of database connection retries during startup. Set to -1 to
# specify an infinite retry count (integer value)
# Deprecated group/name - [DEFAULT]/sql_max_retries
# Deprecated group/name - [DATABASE]/sql_max_retries
#max_retries = 10

# Interval between retries of opening a SQL connection (integer value)
# Deprecated group/name - [DEFAULT]/sql_retry_interval
# Deprecated group/name - [DATABASE]/reconnect_interval
#retry_interval = 10

# If set, use this value for max_overflow with SQLAlchemy (integer value)
# Deprecated group/name - [DEFAULT]/sql_max_overflow
# Deprecated group/name - [DATABASE]/sqlalchemy_max_overflow
#max_overflow = 50

# Verbosity of SQL debugging information: 0=None, 100=Everything (integer
# value)
# Minimum value: 0
# Maximum value: 100
# Deprecated group/name - [DEFAULT]/sql_connection_debug
#connection_debug = 0
```

(continues on next page)

(continued from previous page)

```
# Add Python stack traces to SQL as comment strings (boolean value)
# Deprecated group/name - [DEFAULT]/sql_connection_trace
#connection_trace = false

# If set, use this value for pool_timeout with SQLAlchemy (integer value)
# Deprecated group/name - [DATABASE]/sqlalchemy_pool_timeout
#pool_timeout = <None>

# Enable the experimental use of database reconnect on connection lost␣
↪(boolean
# value)
#use_db_reconnect = false

# Seconds between retries of a database transaction (integer value)
#db_retry_interval = 1

# If True, increases the interval between retries of a database operation up␣
↪to
# db_max_retry_interval (boolean value)
#db_inc_retry_interval = true

# If db_inc_retry_interval is set, the maximum seconds between retries of a
# database operation (integer value)
#db_max_retry_interval = 10

# Maximum retries in case of connection error or deadlock error before error␣
↪is
# raised. Set to -1 to specify an infinite retry count (integer value)
#db_max_retries = 20

# Optional URL parameters to append onto the connection URL at connect time;
# specify as param1=value1&param2=value2& (string value)
#connection_parameters =


[dispatcher]

#
# From monasca_api
#

# Versions controller (string value)
#versions = monasca_api.v2.reference.versions:Versions

# Version 2.0 controller (string value)
#version_2_0 = monasca_api.v2.reference.version_2_0:Version2

# Metrics controller (string value)
#metrics = monasca_api.v2.reference.metrics:Metrics
```

(continues on next page)

(continued from previous page)

```
# Metrics measurements controller (string value)
#metrics_measurements = monasca_api.v2.reference.metrics:MetricsMeasurements

# Metrics statistics controller (string value)
#metrics_statistics = monasca_api.v2.reference.metrics:MetricsStatistics

# Metrics names controller (string value)
#metrics_names = monasca_api.v2.reference.metrics:MetricsNames

# Alarm definitions controller (string value)
#alarm_definitions = monasca_api.v2.reference.alarm_
↪definitions:AlarmDefinitions

# Alarms controller (string value)
#alarms = monasca_api.v2.reference.alarms:Alarms

# Alarms Count controller (string value)
#alarms_count = monasca_api.v2.reference.alarms:AlarmsCount

# Alarms state history controller (string value)
#alarms_state_history = monasca_api.v2.reference.alarms:AlarmsStateHistory

# Notification Methods controller (string value)
#notification_methods = monasca_api.v2.reference.notifications:Notifications

# Dimension Values controller (string value)
#dimension_values = monasca_api.v2.reference.metrics:DimensionValues

# Dimension Names controller (string value)
#dimension_names = monasca_api.v2.reference.metrics:DimensionNames

# Notifications Type Methods controller (string value)
#notification_method_types = monasca_api.v2.reference.
↪notificationstype:NotificationsType

# Logs controller (string value)
#logs = monasca_api.v2.reference.logs:Logs

# Health checks endpoint controller (string value)
#healthchecks = monasca_api.healthchecks:HealthChecks


[influxdb]

#
# From monasca_api
#
```

(continues on next page)

(continued from previous page)

```
#
# Database name where metrics are stored
#  (string value)
#database_name = mon

#
# Whether to use a separate database per tenant
#  (boolean value)
#db_per_tenant = false

#
# IP address to Influxdb server
#  (host address value)
#ip_address = 127.0.0.1

# Port to Influxdb server (port value)
# Minimum value: 0
# Maximum value: 65535
#port = 8086

#
# Influxdb user
#  (string value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#user = monasca-api

#
# Influxdb password
#  (string value)
#
# This option has a sample default set, which means that
# its actual default value may vary from the one documented
# below.
#password = password


[kafka]

#
# From monasca_api
#

# Comma separated list of Kafka broker host:port (list value)
#uri = 127.0.0.1:9092

# The topic that metrics will be published to (string value)
```

(continues on next page)

(continued from previous page)

```
#metrics_topic = metrics

# The topic that events will be published to (string value)
#events_topic = events

# The topic that logs will be published to (multi valued)
#logs_topics = log

# The topic that alarm state will be published to (string value)
#alarm_state_transitions_topic = alarm-state-transitions

# The group name that this service belongs to (string value)
#group = api

# The ack time back to kafka. (NOT USED) (integer value)
#ack_time = 20

# Number of retries in case of connection error (NOT USED) (integer value)
#max_retry = 3

# Whether posting is asynchronous or not (NOT USED) (boolean value)
#is_async = true

# Specify if the message received should be parsed. If True, message will not
# be parsed, otherwise messages will be parsed (NOT USED) (boolean value)
#compact = true

# Warning: Failed to format sample for partitions
# type object 'int' has no attribute '_formatter'

# Specify if received data should be simply dropped. This parameter is only␣
↪for
# testing purposes. (NOT USED) (boolean value)
#drop_data = false

# The maximum number of metrics per payload sent to Kafka. Posts to the␣
↪Monasca
# API which exceed this will be chunked into batches not exceeding this number
# (integer value)
#queue_buffering_max_messages = 1000

# The wait time when no messages on kafka queue (NOT USED) (integer value)
# Minimum value: 1
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#wait_time = 1

# Whether the message is automatically committed (NOT USED) (boolean value)
```

(continues on next page)

(continued from previous page)

```
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#auto_commit = false

# Enable legacy Kafka client. When set old version of kafka-python library is
# used. Message format version for the brokers should be set to 0.9.0.0 to
# avoid performance issues until all consumers are upgraded (boolean value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#legacy_kafka_client_enabled = false


[keystone_authtoken]

#
# From keystonemiddleware.auth_token
#

# 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 (string
# value)
# Deprecated group/name - [keystone_authtoken]/auth_uri
#www_authenticate_uri = <None>

# DEPRECATED: 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 (string value)
# 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_uri = <None>

# API version of the Identity API endpoint (string value)
#auth_version = <None>
```

(continues on next page)

(continued from previous page)

```
# Interface to use for the Identity API endpoint. Valid values are "public",
# "internal" (default) or "admin" (string value)
#interface = internal

# Do not handle authorization requests within the middleware, but delegate the
# authorization decision to downstream WSGI components (boolean value)
#delay_auth_decision = false

# Request timeout value for communicating with Identity API server (integer
# value)
#http_connect_timeout = <None>

# How many times are we trying to reconnect when communicating with Identity
# API Server (integer value)
#http_request_max_retries = 3

# 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 (string value)
#cache = <None>

# Required if identity server requires client certificate (string value)
#certfile = <None>

# Required if identity server requires client certificate (string value)
#keyfile = <None>

# A PEM encoded Certificate Authority to use when verifying HTTPs connections.
# Defaults to system CAs (string value)
#cafile = <None>

# Verify HTTPS connections (boolean value)
#insecure = false

# The region in which the identity server can be found (string value)
#region_name = <None>

# Optionally specify a list of memcached server(s) to use for caching. If left
# undefined, tokens will instead be cached in-process (list value)
# Deprecated group/name - [keystone_authtoken]/memcache_servers
#memcached_servers = <None>

# 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 (integer value)
#token_cache_time = 300
```

(continues on next page)

(continued from previous page)

```
# (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 (string value)
# Possible values:
# None - <No description provided>
# MAC - <No description provided>
# ENCRYPT - <No description provided>
#memcache_security_strategy = None

# (Optional, mandatory if memcache_security_strategy is defined) This string␣
↪is
# used for key derivation (string value)
#memcache_secret_key = <None>

# (Optional) Number of seconds memcached server is considered dead before it␣
↪is
# tried again (integer value)
#memcache_pool_dead_retry = 300

# (Optional) Maximum total number of open connections to every memcached␣
↪server
# (integer value)
#memcache_pool_maxsize = 10

# (Optional) Socket timeout in seconds for communicating with a memcached
# server (integer value)
#memcache_pool_socket_timeout = 3

# (Optional) Number of seconds a connection to memcached is held unused in the
# pool before it is closed (integer value)
#memcache_pool_unused_timeout = 60

# (Optional) Number of seconds that an operation will wait to get a memcached
# client connection from the pool (integer value)
#memcache_pool_conn_get_timeout = 10

# (Optional) Use the advanced (eventlet safe) memcached client pool (boolean
# value)
#memcache_use_advanced_pool = 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 (boolean value)
#include_service_catalog = true
```

(continues on next page)

(continued from previous page)

```
# 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
# (string value)
#enforce_token_bind = permissive

# 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 (list value)
#service_token_roles = service

# For backwards compatibility reasons we must let valid service tokens pass
# that don't 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
# (boolean value)
#service_token_roles_required = false

# 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 (string value)
#service_type = <None>

# Authentication type to load (string value)
# Deprecated group/name - [keystone_authtoken]/auth_plugin
#auth_type = <None>

# Config Section from which to load plugin specific options (string value)
#auth_section = <None>


[log_publisher]

#
# From monasca_api
#


#
# Message max size that can be sent to kafka, default to 1048576 bytes
#  (integer value)
```

(continues on next page)

(continued from previous page)

```
#max_message_size = 1048576

#
# Region
#  (string value)
#region = Region;

#
# Refers to payload/envelope size. If either is exceeded API will throw an
# error
#  (integer value)
#max_log_size = 1048576


[messaging]

#
# From monasca_api
#

#
# The message queue driver to use
#  (string value)
#driver = <None>

# DEPRECATED:
# The type of metrics message format to publish to the message queue
#  (string value)
# This option is deprecated for removal since 2.1.0.
# Its value may be silently ignored in the future.
# Reason:
# Option is not used anywhere in the codebase
#metrics_message_format = reference

# DEPRECATED:
# The type of events message format to publish to the message queue
#  (string value)
# This option is deprecated for removal since 2.1.0.
# Its value may be silently ignored in the future.
# Reason:
# Option is not used anywhere in the codebase
#events_message_format = reference


[oslo_policy]

#
# From oslo.policy
#
```

(continues on next page)

(continued from previous page)

```
# 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 (boolean value)
#enforce_scope = 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 OR'd with the new policy check
# string, allowing for a graceful upgrade experience between releases with new
# policies, which is the default behavior (boolean value)
#enforce_new_defaults = false

# 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 (string value)
#policy_file = policy.json

# Default rule. Enforced when a requested rule is not found (string value)
#policy_default_rule = default

# 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 (multi
# valued)
#policy_dirs = policy.d

# Content Type to send and receive data for REST based policy check (string
# value)
# Possible values:
# application/x-www-form-urlencoded - <No description provided>
# application/json - <No description provided>
#remote_content_type = application/x-www-form-urlencoded

# server identity verification for REST based policy check (boolean value)
#remote_ssl_verify_server_crt = false

# Absolute path to ca cert file for REST based policy check (string value)
#remote_ssl_ca_crt_file = <None>
```

(continues on next page)

(continued from previous page)

```
# Absolute path to client cert for REST based policy check (string value)
#remote_ssl_client_crt_file = <None>

# Absolute path client key file REST based policy check (string value)
#remote_ssl_client_key_file = <None>


[repositories]

#
# From monasca_api
#

#
# The repository driver to use for metrics
#  (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#metrics_driver = monasca_api.common.repositories.influxdb.metrics_
↪repository:MetricsRepository

#
# The repository driver to use for alarm definitions
#  (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#alarm_definitions_driver = monasca_api.common.repositories.sqla.alarm_
↪definitions_repository:AlarmDefinitionsRepository

#
# The repository driver to use for alarms
#  (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#alarms_driver = monasca_api.common.repositories.sqla.alarms_
↪repository:AlarmsRepository

#
# The repository driver to use for notifications
#  (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#notifications_driver = monasca_api.common.repositories.sqla.notifications_
↪repository:NotificationsRepository
```

(continues on next page)

(continued from previous page)

```
#
# The repository driver to use for notifications
#  (string value)
# Advanced Option: intended for advanced users and not used
# by the majority of users, and might have a significant
# effect on stability and/or performance.
#notification_method_type_driver = monasca_api.common.repositories.sqla.
↪notification_method_type_repository:NotificationMethodTypeRepository


[security]

#
# From monasca_api
#

#
# Roles that are allowed to check the health
#  (list value)
#healthcheck_roles = @

#
# Roles that are allowed to check the versions
#  (list value)
#versions_roles = @

#
# Roles that are allowed full access to the API
#  (list value)
#default_authorized_roles = monasca-user

#
# Roles that are only allowed to POST to the API
#  (list value)
#agent_authorized_roles = monasca-agent

#
# Roles that are only allowed to GET from the API
#  (list value)
#read_only_authorized_roles = monasca-read-only-user

#
# Roles that are allowed to POST metrics on behalf of another tenant
#  (list value)
#delegate_authorized_roles = admin
```

Sample Configuration For Logging

This sample configuration can also be viewed in api-logging.conf.

```
[loggers]
keys = root, sqlalchemy, kafka, kafkalib

[handlers]
keys = console, file

[formatters]
keys = context

[logger_root]
level = DEBUG
handlers = console, file

[logger_sqlalchemy]
qualname = sqlalchemy.engine
# "level = INFO" logs SQL queries.
# "level = DEBUG" logs SQL queries and results.
# "level = WARN" logs neither.  (Recommended for production systems.)
level = DEBUG
handlers = console, file
propagate=0

[logger_kafka]
qualname = kafka
level = DEBUG
handlers = console, file
propagate = 0

[logger_kafkalib]
qualname = monasca_common.kafka_lib
level = INFO
handlers = console, file
propagate = 0

[handler_console]
class = logging.StreamHandler
args = (sys.stderr,)
level = DEBUG
formatter = context

[handler_file]
class = logging.handlers.RotatingFileHandler
level = DEBUG
formatter = context
# store up to 5*100MB of logs
args = ('/var/log/monasca/api/monasca-api.log', 'a', 104857600, 5)
```

(continues on next page)

(continued from previous page)

```
[formatter_context]
class = oslo_log.formatters.ContextFormatter
```

Sample Configuration For Paster

This sample configuration can also be viewed in api-config.ini.

```
[DEFAULT]
name = monasca_api

[pipeline:main]
pipeline = request_id auth api

[app:api]
paste.app_factory = monasca_api.api.server:launch

[filter:auth]
paste.filter_factory = monasca_api.healthcheck.keystone_protocol:filter_
↪factory

[filter:request_id]
paste.filter_factory = oslo_middleware.request_id:RequestId.factory

[server:main]
use = egg:gunicorn#main
host = 127.0.0.1
port = 8070
workers = 9
worker-connections = 2000
worker-class = eventlet
timeout = 30
backlog = 2048
keepalive = 2
proc_name = monasca-api
loglevel = DEBUG
```