RBFS Time Series Database Configuration Guide

Version 20.9.1, 25 September 2020

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

Table of Contents

1. Introduction

Operational-state visibility is key for troubleshooting, testing, monitoring and capacity management. This requires to sample router metrics periodically. Ingestion of time-series data allows to ask interesting operational queries.

Examples:

- A slightly increasing memory consumption over time while overall PPPoE session count has not changed, for example, is an indication for a memory leak.
- If the 5 Minute chassis temperature is too high, this might be an indication for an imminent hardware breakdown and the switch hardware must be replaced.
- If utilization of all fabric interfaces is constantly touching the 80% saturation levels then new fabric links must be commissioned.
- High input traffic with degradation of optical receive levels might be an indication of running very close to optical budget.

The challenge is to sample all these information efficiently in terms of disk, memory and CPU utilization while providing comprehensive query and reporting functionality.

1.1. Architectural Overview

The RBFS telemetry architecture is based on Prometheus as an open-source systems monitoring and alerting toolkit. Prometheus is designed to pull metrics periodically, and save them efficiently. It allows to analyze the metrics with a powerful query language called PromQL. Also an optional alert management is available. There is opportunity to tie it together with own services to integrate it into the system landscape. Data should have short retention times (default 15d).

This fits perfectly to the needs in BDS. The figure below shows how it fits in an overall architecture.

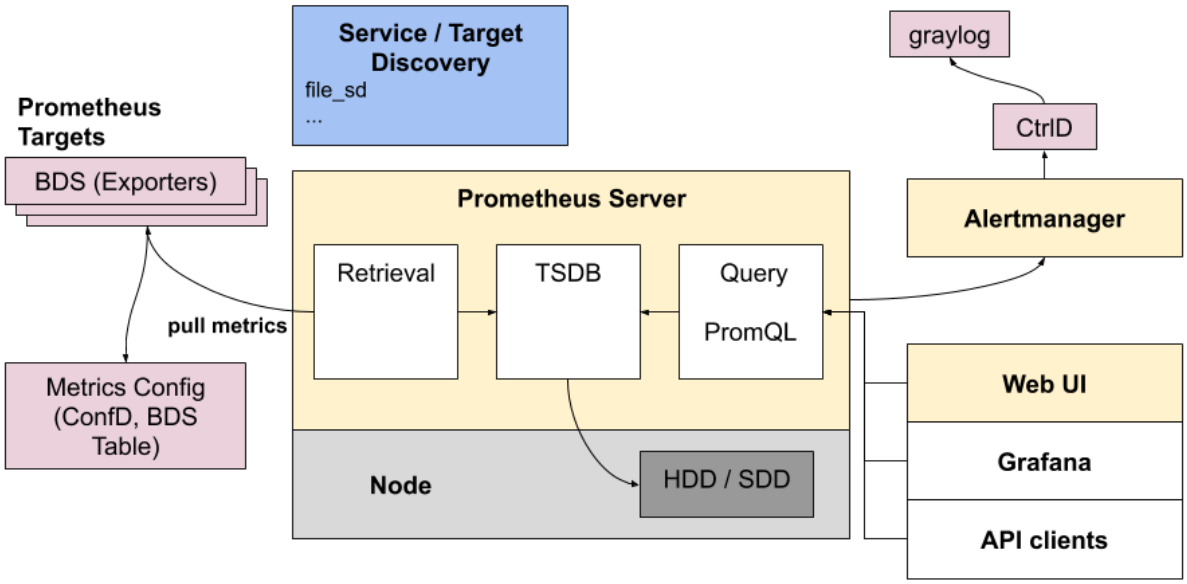


Figure 1. Prometheus in RBFS

To mitigate the short retention times, which fits to BDS but not in an overall telemetry process, the data can be stored in a centralized storage database (for example, Influx) this can be done by federation or via remote storage adapters. To distribute the alert messages from prometheus, CTRLD functions as "alertmanager webhook receiver", which takes the alert and distributes it to a log management tool (graylog).

1.1.1. Router deployment model

Prometheus DB is run on the router as a dedicated process. It ships with a package-time configuration to poll each BDS capable speaker at periodic intervals. Initially the periodic interval is 1 second. The Prometheus Exposition format is a very simple HTTP based GET query which asks a given BD speaker "Give me all your metrics". Each BD subscribes to the *global.time-series.metric.config* table, which contains an operator-configurable list of BDS targets. Only the BDS which is master of a table responds. Next Prometheus polls the BD using the */metrics* URL.

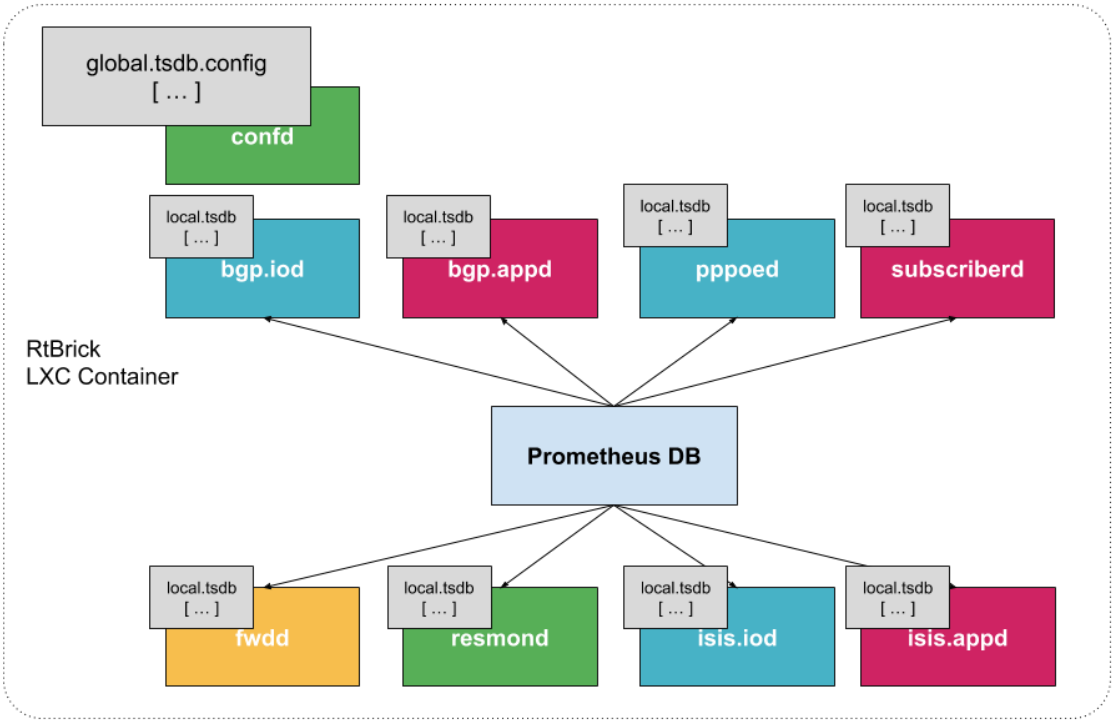


Figure 2. Prometheus in RBFS with the different scrape target

1.1.2. Storage efficiency

On an average Prometheus uses only around 1-2 bytes per sample. Thus, to plan the capacity of a Prometheus server, you can use the rough formula:

```
needed_disk_space = retention_time_seconds * ingested_samples_per_second *
bytes_per_sample
```

The single binaries disk space:

```
-rwxr-xr-x 1 root root 27M Sep 2 22:51 alertmanager +
-rwxr-xr-x 1 root root 81M Sep 2 22:51 prometheus +
-rwxr-xr-x 1 root root 49M Sep 3 19:55 promtool
```

Promtool is needed to test the configurations before set them to prometheus.

1.1.3. Alerting

The alerting is configured through Prometheus. For more information, see alertmanager.

1.1.4. Role of CTRLD

Figure-4 provides an overview of the role of CTRLD.

Prometheus and Alertmanager register themself in CTRLD, so that CTRLD is aware of these two services.

1.1.4.1. Service state and Proxy

The registration of the services gives 2 advantages:

1. The operational state is an indicator if the service is up and running.
2. The proxy functionality of CTRLD can be used for prometheus and alertmanager.

The proxy functionality is used for querying prometheus directly:

```
curl
'http://192.168.202.125:19091/api/v1/rbfs/elements/rtbrick/services/PROMETHEU
S/proxy/api/v1/query?query=up' | jq .
```

But it is also used for federation and therefore the following URL is used:

```
http://192.168.202.125:19091/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS
/proxy/federate
```

1.1.4.2. Alert distribution

CTRLD can forward the alerts from the alertmanager to graylog or any other REST endpoint.

1.1.4.3. REST API for Configuration

CTRLD provides a REST API Endpoint for configuration of alerts and metrics.

1.1.5. Federation deployment model

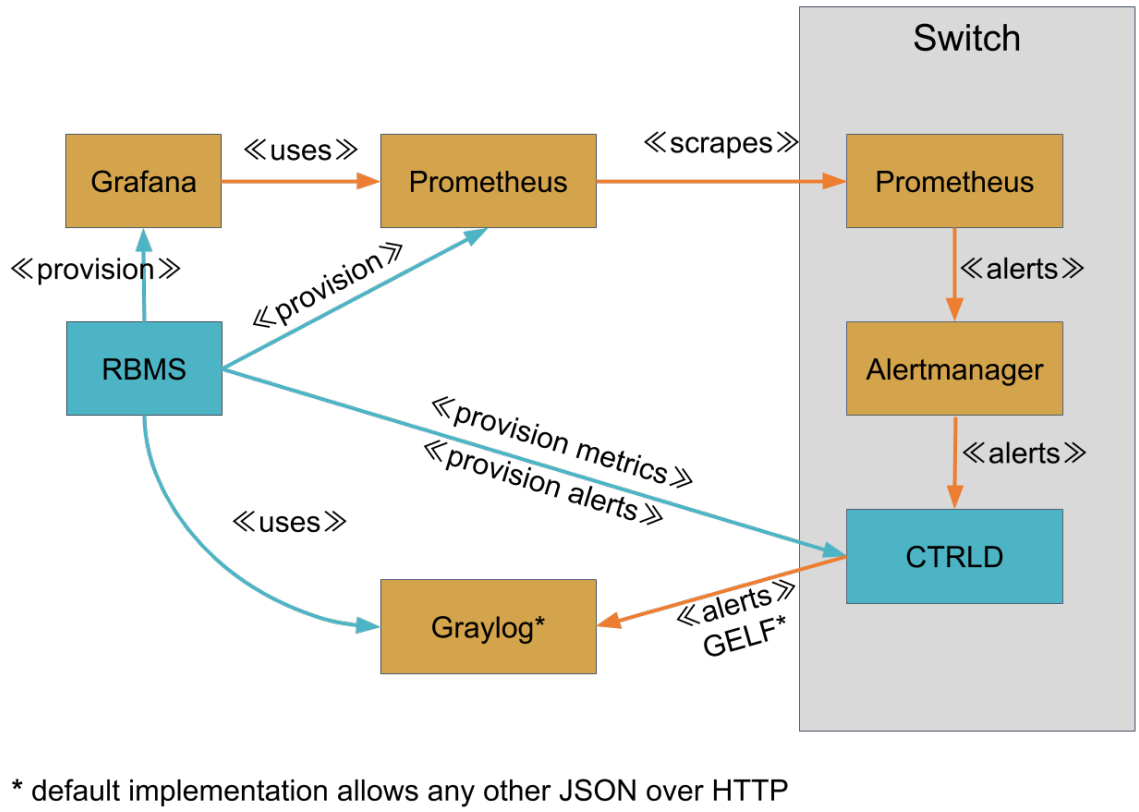


Figure 3. Federation of Prometheus, Alertmanager and graylog target

Prometheus is intended to have at least one instance per datacenter usually; also with a global Prometheus for global graphing or alerting. Federation allows for pulling metrics and aggregations up the hierarchy.

In the global Prometheus config, this timeseries is pulled:

prometheus.yml:

```
global:
  scrape_interval: 60s # By default, scrape targets every 15 seconds.
  # A scrape configuration containing exactly one endpoint to scrape:
scrape_configs:
  - job_name: "federate"

    honor_labels: true
    metrics_path: '/federate'
    params:
      'match[]':
        - '{job="bds"}'
    scrape_interval: 15s
    # Patterns for files from which target groups are extracted.
    file_sd_configs:
      - files:
          - ./bds.target.yml
        refresh_interval: 5m
```

The match[] here requests all BDS job time series. By following this job naming

convention, you do not have to adjust the config every time when there is a new aggregating rule.

The targets itself can be configured in a separate file.

bds.target.yml:

```
- targets: ['192.168.202.125:19091']
  labels:
    __metrics_path__:
"/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS/proxy/federate"
    box: 125_rtbrick
```

2. Installation

The RtBrick fullstack comes with a ready to use tsdb instance. So no more installation on RBFS has to be done.

For federation of metrics, a global prometheus instance is needed. To visualize the metrics a Grafana instance has to be installed, and to get the alert messages, a graylog instance has to be set up. This document does not contain an installation guide for that systems.

The information about configuring a federation Pprometheus to scrape metrics from a RBFS installation is described in the Federation deployment model section.

3. Configuring Time Series Database

The following section describes how to configure the system to gather metrics and alerts out of the system.

3.1. Metric

To better understand the Data Model have a look at the Prometheus Data Model.

3.1.1. Metric Data Model

In RBFS it is possible to turn each table attribute into a metric.

When you export the time-series metric data for an attribute which has more than 50 label values (user-defined, default labels), you may see truncated data in the exported metric.

The following table describes the configuration model:

Metric	
metric_name	Name of the metric (metric name conventions). That is the unique identifier for the metric.
table_name	Table Name for which the metric is designed, could also be a regular expression.
bds_metric_type	• object-metric: if the metric should be gathered from regular table attributes • index-metric: if the metric should be gathered out of an attribute of an index table
index_name	Name of the index, if the bds_metric_type is index-metric.

metric_type	• gauge: is a metric that represents a single numerical value that can arbitrarily go up and down. Gauges are typically used for measured values like temperatures or current memory usage, but also "counts" that can go up and down, like the number of concurrent requests. • counter: is a cumulative metric that represents a single monotonically increasing counter whose value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors. Do not use a counter to expose a value that can decrease. For example, do not use a counter for the number of currently running processes; instead use a gauge.
metric_description	Description of the metric.
attributes	List of Attributes (see Attribute Table) that will be streamed as metric.
filters	List of AttributeFilters (see AttributeFilter Table) that filters the table rows which should be considered for metric generation. Each filter in this list has to match in order to generate the metric, so the list implies an implicit AND.

Attribute	
attribute_name	Name of the attribute that should be streamed as metric. This Attribute has to be a numeric type, or a type that has a numeric converter.
filters	List of AttributeFilters (see the [AttributeFilter] table) that filters the table rows which should be considered for metric generation. Each filter in this list has to match in order to generate the metric, so the list implies an implicit AND.
labels	List of AttributeLabels (see the [AttributeLabel] table) that are attached to that metric.

AttributeFilter	
match_attribute_name	Attribute of the Table which is used to match against.

match_type	• exact: so the attribute has to match exactly the value • regex: the value is of type regex and the attribute has to match against the regular expression
match_value	The value that attribute has to match against.

AttributeLabel **CAUTION**: Remember that every unique combination of key-value label pairs represents a new time series, which can dramatically increase the amount of data stored. Do not use labels to store dimensions with high cardinality (many different label values), such as user IDs, email addresses, or other unbounded sets of values.	
label_name	Name of the Label (label name conventions).
dynamic	bool: If the label is dynamic, the label_value is treated as attribute_name, so the value of the attribute is used as the label value, otherwise the label value is used directly.
label_value	The value of the label or the attribute which should be used as label value.
filters	List of AttributeFilters (see [AttributeFilter] Table) that filters the table rows which should be considered for label generation. Each filter in this list has to match in order to generate the label, so the list implies an implicit AND.

There are various ways to configure metrics that should be gathered in the system.

3.1.1.1. Allowed Attribute Types (Type Converters)

Normally only attributes are allowed, which are of type numeric, but for some types, there are built-in type converters, which allow also to use attributes of their types.

For the following BDS types, built-in type converters are provided by BDS. As per Prometheus data model, type converter will convert the BDS type into a 64bit float number.

BDS data type	Outcome number represents
unix-wallclock-timestamp	Seconds
unix-usec-wallclock-timestamp	Seconds
unix-usec-monotonic-timestamp	Seconds
unix-usec-coarse-wallclock-timestamp	Seconds

BDS data type	Outcome number represents
bandwidth	bps(bit per second)
temperature	Degree Celsius

3.1.2. Configuring Metrics

The configuration of the Metrics can be done in various ways.

3.1.2.1. Configuring Metrics using Command Line Interface

To configure the Time Series Database, perform the following steps:

1. Define Metric configuration
2. Define Attribute configuration
3. Optional Filters at Metric Level and Attribute level
4. Defining labels to be attached to exported metric

3.1.2.1.1. Metric Configuration

Metric configuration is used to configure the parameters of the metric data being exported.

Syntax

[time-series]

edit metric <metric-name>

[time-series metric <metric-name>]

set metric-description <128 character description about the metric-name >

set prometheus-type <counter / gauge>

set metric-bds-type <object-metric / index-metric>

set table-name <table-name>

set attribute-name <attribute-name>

[time-series-metric <metric-name> metric-type index-metric]
set index-name <index-name>

Command arguments

edit metric <metric-name>	Specifies the name of the metric exported, as would be reflected in Prometheus. Use the naming conventions as recommended by Prometheus
set metric-description <128 character description about the metric-name >	Description of the metric
set prometheus-type <counter / gauge>	Configures the metric data type. Currently the supported Prometheus metric data are: counter and gauge
set metric-bds-type < object-metric / index-metric >	Specifies the type of attribute, that is scraped and exported. There are two types, object-metric and index-metric
set table-name <table-name>	Specifies the target table, from which the data is scraped and exported.
set attribute-name <attribute-name>	Specifies the name of the attribute, in the target table to be scraped and exported
set index-name <index-name>	Specifies the index-name of the index-metric attribute. This config applicable for index-metric alone.

Example

```
root@rtbrick:confd> edit time-series

[ time-series ]

root@rtbrick:confd> edit metric interface_statistics_data

[ time-series metric interface_statistics_data ]

root@rtbrick:confd> set metric-bds-type object-metric

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> set prometheus-type counter

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> set description Interface statistics accounting info

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> set table-name global.interface.physical.statistics

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> set attribute-name port_stat_if_in_ucast_pkts

[ time-series metric interface_statistics_data ]
```

3.1.2.1.2. Metric Filter Configuration

Metric filter configuration is used to configure the parameters of the filter. It is used to filter the exported metric. This is an optional configuration.

Syntax

[time-series]

edit metric <metric-name> **filter** <match-attribute-name>

[time-series metric interface_statistics_data filter <match-attribute-name>]

set match-type < exact / regular-expression >

[time-series metric interface_statistics_data filter <match-attribute-name>]

set match-attribute-value <match-attribute-value>

Command arguments

edit metric <metric-name> filter <match-attribute-name>	Specifies the filter that filters the exported metric, based on specified criteria. This is optional configuration.
set match-type < exact / regular-expression >	Specifies the match type to be used, There are two options, exact and regular-expression.
set match-attribute-value <match-attribute-value>	Specifies the attribute value used for match. Fixed value for exact. Regex pattern for regular-expression

Example

Exact Value

```
[ time-series ]
root@rtbrick:confd> edit metric interface_statistics_data filter
interface_name

[ time-series metric interface_statistics_data filter interface_name ]
root@rtbrick:confd> set match-type exact

[ time-series metric interface_statistics_data filter interface_name ]
root@rtbrick:confd> set match-attribute-value ifp-0/0/50
```

Regular Expression

```
[ time-series ]
root@rtbrick:confd> edit metric interface_statistics_data filter
interface_name
[ time-series metric interface_statistics_data filter interface_name ]
root@rtbrick:confd> set match-type regular-expression
[ time-series metric interface_statistics_data filter interface_name ]
root@rtbrick:confd> set match-attribute-value ifp-0/0/5[0-9]+
```

3.1.2.1.3. Metric Attribute Label Configuration

Metric attribute config is used to configure the labels to be attached to the exported metric.

Syntax

[time-series metric <metric-name>]

edit attribute-name <metric-name>

[time-series metric <metric-name> attribute-name <attribute-name>]

edit label-key <label-key>

[time-series metric interface_statistics_data attribute-name port_stat_if_in_ucast_pkts label-key <label-key>]

set label-type < dynamic / static >

set label-value <label-value>

Command arguments

edit label-key <label-key>	Specifies the name of label. User definable, Please use naming conventions as recommended by Prometheus
set label-type < dynamic / static>	Specifies the type of labels, a static value or dynamic value to be added.
set label-value <label-value>	Specifies the label-value to be used.

Example

Dynamic Label

```
root@rtbrick:confd> edit time-series

[ time-series ]
root@rtbrick:confd> edit metric interface_statistics_data

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> edit attribute-name port_stat_if_in_ucast_pkts

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts ]
root@rtbrick:confd> edit label-key Attribute

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key Attribute ]
root@rtbrick:confd> set label-type dynamic

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key Attribute ]
root@rtbrick:confd> set label-value interface_name
```

Static Label

```
root@rtbrick:confd> edit time-series

[ time-series ]
root@rtbrick:confd> edit metric interface_statistics_data

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> edit attribute-name port_stat_if_in_ucast_pkts

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts ]
root@rtbrick:confd> edit label-key Attribute

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key Attribute ]
root@rtbrick:confd> set label-type static

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key Attribute ]
root@rtbrick:confd> set label-value leaf1_pod1

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key Attribute ]
```

3.1.2.1.4. Metric Attribute Filter Configuration

Attribute filter config is used to configure the parameters of Attribute filter. It is used to filter the exported metric based on certain fields of the attribute. This is an optional configuration.

Syntax

[time-series]

root@rtbrick:confd> **edit metric** <metric name>

[time-series metric <metric name>]

root@rtbrick:confd> **edit attribute-name** <attribute name>

[time-series metric <metric name> attribute-name <attribute name>]

root@rtbrick:confd> **edit filter** <filter name>

[time-series metric <metric name> attribute-name <attribute name> filter <filter name>]

root@rtbrick:confd> **set match-type** < exact / regular-expression >

[time-series metric <metric name> attribute-name <attribute name> filter <filter name>]

root@rtbrick:confd> **set match-attribute-value** <match-attribute-value>

Command arguments

edit metric <metric-name> attribute-name <attribute name> filter <match-attribute-name>	Specifies the filter that filters the exported metric , based on criteria of the attribute. This is optional config.
set match-type < exact / regular-expression >	Specifies the match type to be used, There are two options, exact and regular-expression.
set match-attribute-value <match-attribute-value>	Specifies the attribute value used for match. Fixed value for exact. Regex pattern for regular-expression

Example

```
The below example shows, the metric attribute will be exported only if  the
port_stat_if_in_discards is exactly 0.

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts ]
root@rtbrick:confd> edit filter port_stat_if_in_discards

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts filter port_stat_if_in_discards ]
root@rtbrick:confd> set match-type exact

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts filter port_stat_if_in_discards ]
root@rtbrick:confd> set match-attribute-value 0
```

3.1.2.1.5. Metric Label Filter Configuration

Label filter configuration is used to set filter parameters that can be used to attach label based on certain criteria. This is an optional configuration.

Syntax

[time-series metric <metric name> attribute-name <Attribute name>]

root@rtbrick:confd> **edit label-key** <label name>

[time-series metric <metric name> attribute-name <Attribute name> label-key <label name>] root@rtbrick:confd> **edit filter** <filter name>

[time-series metric <metric name> attribute-name <Attribute name> label-key <label name> filter <filter name>]

root@rtbrick:confd> **set match-type** < exact / regular-expression >

[time-series metric <metric name> attribute-name <Attribute name> label-key <label name> filter <filter name>]

root@rtbrick:confd> **set match-attribute-value** <match-attribute-value>

Command arguments

edit metric <metric-name> attribute-name <Attribute name> label <lable-key> filter <match-attribute-name>	Specifies the filter that filters the exported metric, based on some attribute value.This is optional config.
set match-type < exact / regular-expression >	Specifies the match type to be used, There are two options, exact and regular-expression.

set match-attribute-value <match-attribute-value>	Specifies the attribute value used for match. Fixed value for exact. Regex pattern for regular-expression

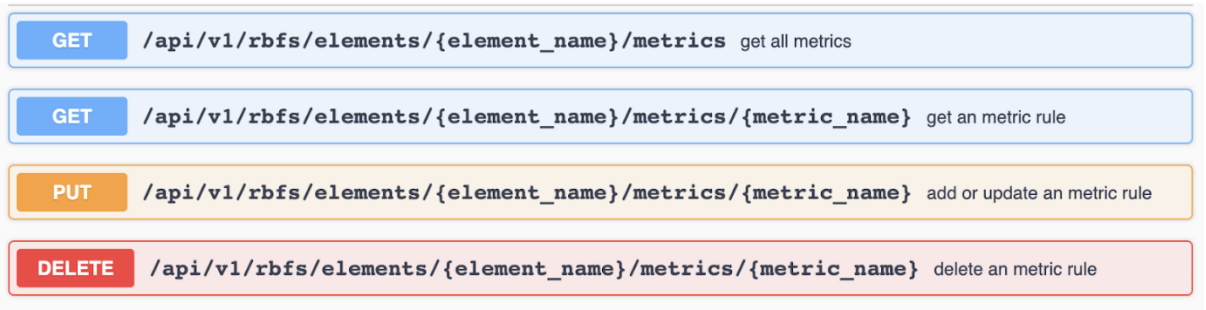
Example

The below example sets label, interface_orientation to the exported data, only if the interface_name matches ifp-0/0/50.

```
[ time-series ]
root@rtbrick:confd> edit metric interface_statistics_data

[ time-series metric interface_statistics_data ]
root@rtbrick:confd> edit attribute-name port_stat_if_in_ucast_pkts

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts ]
root@rtbrick:confd> edit label-key interface_orientation

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key interface_orientation ]
root@rtbrick:confd> edit filter interface_name

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key interface_orientation filter
interface_name ]
root@rtbrick:confd> set match-type exact

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key interface_orientation filter
interface_name ]
root@rtbrick:confd> set match-attribute-value ifp-0/0/50

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key interface_orientation ]
root@rtbrick:confd> set label-type static

[ time-series metric interface_statistics_data attribute-name
port_stat_if_in_ucast_pkts label-key interface_orientation ]
root@rtbrick:confd> set label-value backbone_facing_interface
```

3.1.2.2. Configuing Metrics Using CTRLD

The configuration can be done via the metrics endpoint of CTRLD.



For further information, refer to the Swagger API documentation of CTRLD.

3.2. Alert

RBFS uses the prometheus alerting feature to generate alerts. These alerts are forwarded to an alertmanager instance inside the rbfs container. The alertmanager instance sends the alert to CTRLD which distributes the alert to an HTTP Endpoint.

Alerts are also configured in a BDS table, and they are exported to Prometheus by the system.

3.2.1. Alert Data Model

Alert	
alert_rule_name	The name of the alert rule. That is the unique identifier for the rule.
alert_group	Name of the alert group the alert belongs to. The alert group helps to structure the alerts.
interval	How often the rule should be evaluated. Pattern:"[0-9]+(ms |[smhdwy]" Example:"5s" In Prometheus the the interval can specified per alert group. So the alert alert group for Prometheus is calculated via {alert_group}_{interval}.
expr	Alert evaluation expression in promql
labels	Key, Value pairs of labels that should be applied. The labels clause allows specifying a set of additional labels to be attached to the alert. Any existing conflicting labels will be overwritten. The label values can be templated (see templating).

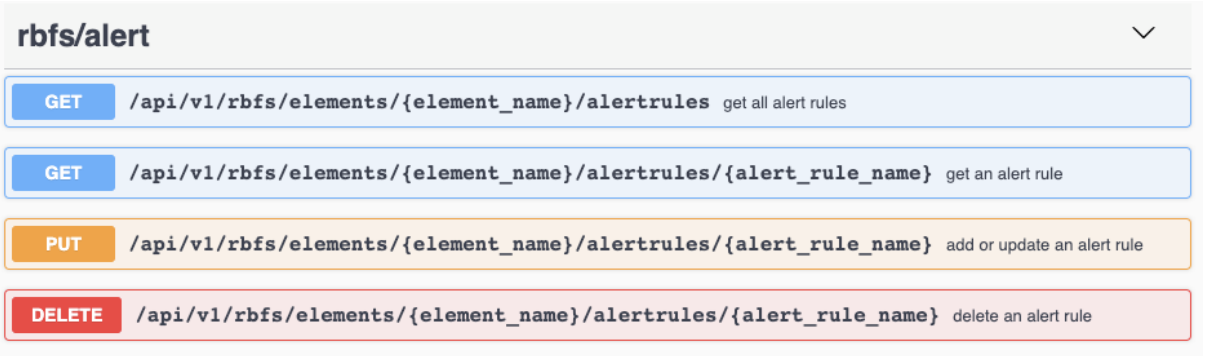
annotations	Key, Value pairs of annotations that should be applied. The annotations clause specifies a set of informational labels that can be used to store longer additional information such as alert descriptions or runbook links. The annotation values can be templated (see templating)
for	Alerts are considered firing once they have been returned for this long. Alerts which have not yet fired for long enough are considered pending. Pattern:"[0-9]+(ms |[smhdwy]" Example:"30s"
level	This is an explicit annotation label with the label name level. This is used to specify the severity: 1.Alert The annotation value can be templated (see templating)
summary	This is an explicit annotation label with the label name summary. The annotation values can be templated (see templating).
description	This is an explicit annotation label with the label name description. The annotation values can be templated (see templating).

3.2.2. Configuration

The configuration of the Metrics can be done in various ways.

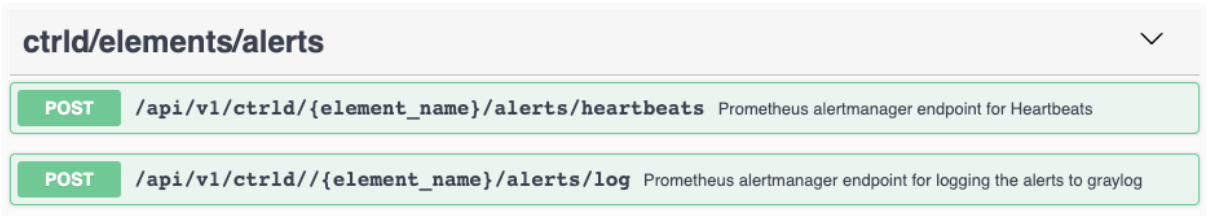
3.2.2.1. Configuing Alert Using CTRLD

The configuration can be done via the alert endpoint of CTRLD.



3.2.3. Graylog Alert Distribution

The alertmanager on RBFS is configured to send alerts to CTRLD.



CTRLD therefore has an endpoint where the alerts are sent to. CTRLD distributes that to a GRAYLOG instance.

The configuration is done in the CTRLD configuration:

```
{ +
"graylog_enable": true, +
"graylog_url": "http://localhost:12201/gelf", +
"graylog_heart_beat_interval": 120 +
}
```