



RBMS User Guides

Version 24.8.1, 11 Sept 2024

Table of Contents

1. RBMS Overview	1
2. Managing Inventory	6
3. Managing Software Images	26
4. Managing Logs	31
5. Metric Management	34
6. Metric Sampling and Monitoring	55
6.1. Metric sampling Overview	55
6.2. Temperature Monitoring	57
6.3. CPU Utilization	63
6.4. PPPoE Session Count	70
6.5. Count of Received IPv4 Unicast Prefixes	73
6.6. Metric Management	74
6.7. Grafana Dashboards	75
6.8. Summary	76
6.9. References	76
7. Administering RBMS	78
7.1. Administration	78
7.2. Managing Jobs	92
8. RBMS Configuration Store	99
9. RBMS Template Engine	115
10. Auto-DNS Provisioning	121
10.1. DNS Introduction	121
10.2. DNS Overview	121
10.3. DNS Record Management	124

1. RBMS Overview

RtBrick's Management System (RBMS) provides network level workflows such as image lifecycle management, network upgrades, event and log Management. RBMS actions are available through REST APIs making them easy to integrate into existing Operations Support Systems (OSS) systems. RBMS provides a single point of interaction for operations staff – from provisioning and management to monitoring and debugging.

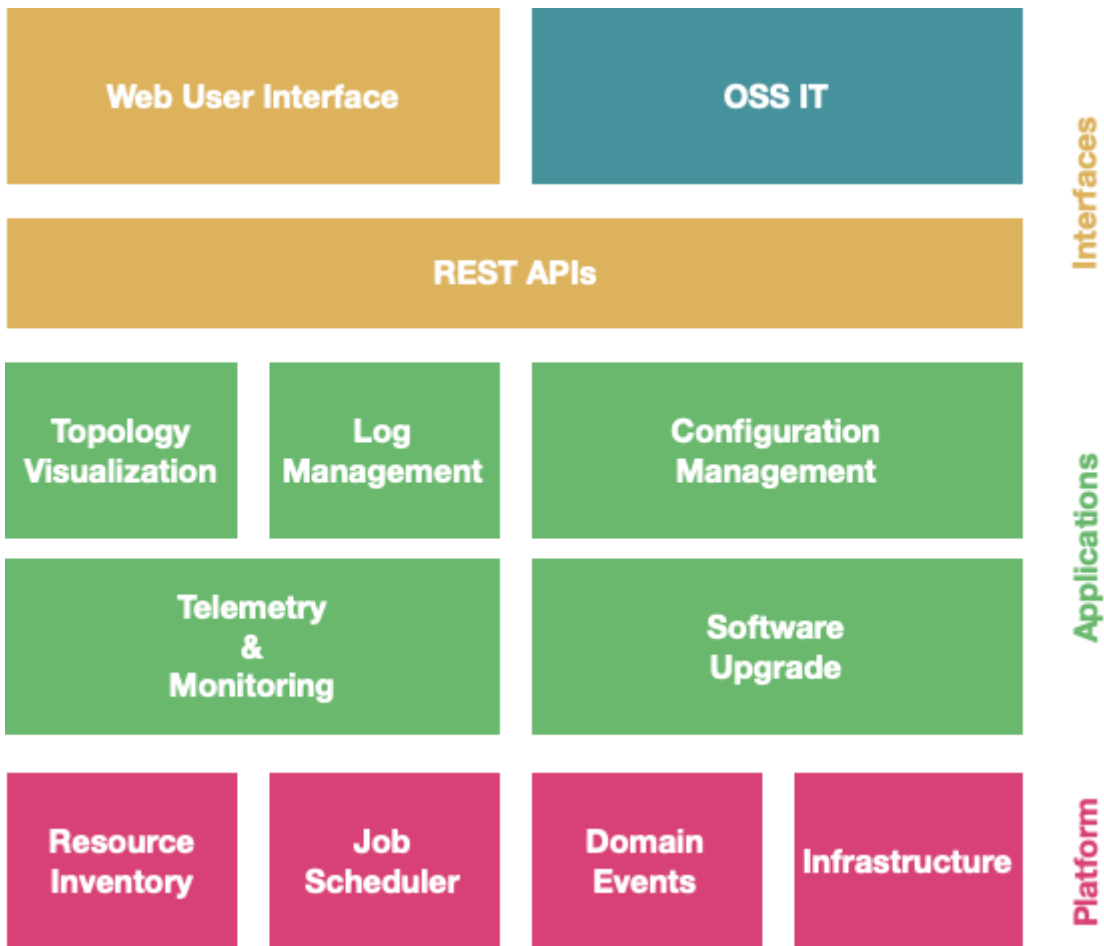
The RBMS user interface provides the following capabilities:

- **Image:** Manage software images
- **Inventory:** Manage network elements
- **Metrics:** Manage the metrics sampled from network devices
- **Jobs:** Review the currently active or scheduled jobs
- **Logs:** View logs messages from the switches
- **Administration:** Manage users, roles, and access tokens.

System Architecture

RBMS follows a *service-oriented architecture* and defines three categories of services:

1. Platform Services
2. Applications
3. Connectors



Platform Services provide the foundation for all network management functions available in RBMS. The platform services include the resource inventory, job scheduler, authentication and authorization and event subscription.

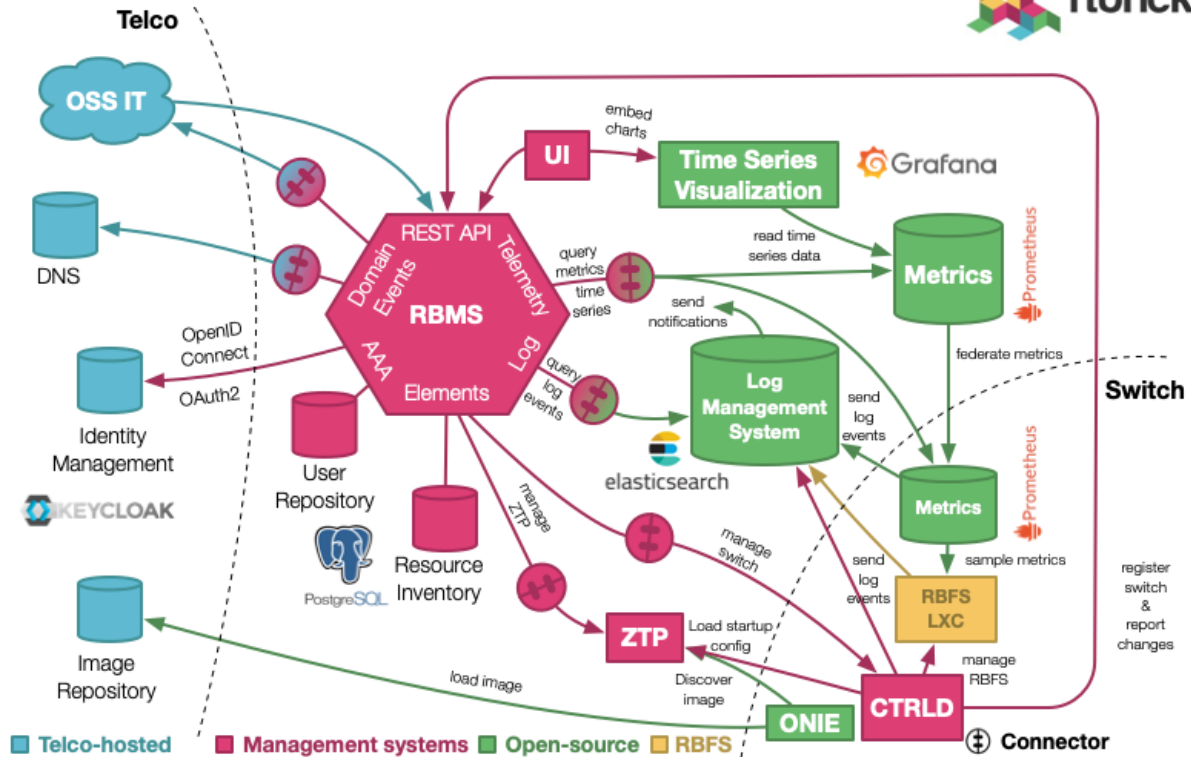
Applications provide a specific network management function. Applications use the platform services and also services provided by other applications to implement network management functions. For example, the topology application visualizes the link-state graph of a fabric.

Connectors interface RBMS with open-source solutions like log management systems or time series databases.

All services provide a REST API to interface with the service. The RBMS User Interface uses these REST APIs. Consequently, all UI actions can be automated through REST API calls. Services can contribute views to the RBMS UI to make the provided service functions accessible from the RBMS UI.

The figure below shows a real-world RBMS deployment.

RBMS Architectural Overview



OSS-IT and RBMS UI manage the switches installed in the network. This includes to inspect the state of a switch but also to manage the switch configurations or to install a new software image.

OSS-IT can also subscribe for *domain events*. A domain event reports when a switch inventory record has changed.

RBMS maintains the switch configurations and publishes the switch startup configuration on the ZTP service. This configuration also includes the software image to be installed on the switch.

ONIE, the open network install environment reads the installer image information from the ZTP service and installs RBFS. Next CTRLD reads the startup configuration from the ZTP service.

RBFS reports log messages to Graylog, an open-source enterprise log management. Graylog uses elasticsearch to store log messages. RBMS is connected to elasticsearch to query the logs and display them in RBMS.

Prometheus, an open-source monitoring system with built-in timeseries database, runs on the switch to sample metrics and to monitor metrics for alert conditions. A centralized Prometheus federates the metrics from the switches. RBMS is

connected to this Prometheus instance to show the metric values in the RBMS user interfaces.

Grafana, an open-source visualization platform, provides dashboards to visualize the time series data. The RBMS UI provides quick-access to the dashboards and embeds dashboard panels in RBMS views.

Authentication and authorization is either delegated to an OpenID/Connect and OAuth2-compatible authorization service or done by the built-in user repository shipped with RBMS.

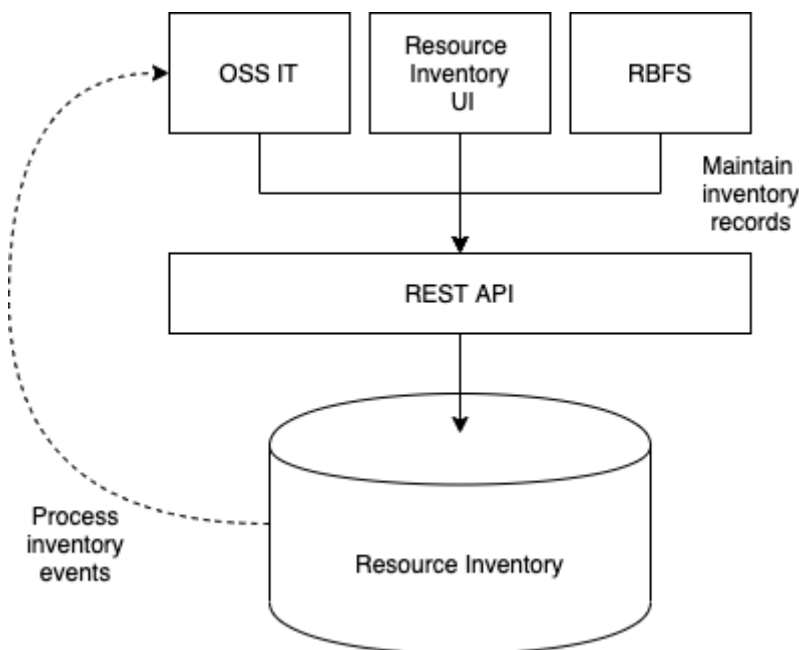
Registered Address	Support	Sales
40268, Dolerita Avenue Fremont CA 94539		
+1-650-351-2251		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

©Copyright 2024 RtBrick, Inc. All rights reserved. The information contained herein is subject to change without notice. The trademarks, logos and service marks ("Marks") displayed in this documentation are the property of RtBrick in the United States and other countries. Use of the Marks are subject to RtBrick's Term of Use Policy, available at <https://www.rtbrick.com/privacy>. Use of marks belonging to other parties is for informational purposes only.

2. Managing Inventory

Introduction

The resource inventory stores information about the elements installed in the network. The resource inventory provides a REST API to maintain the resource inventory records. In addition, the resource inventory contributes views to the RBMS UI. The resource inventory UI only uses the REST API to maintain the resource inventory records. Typically three consumers of the resource REST API exists as illustrated below:

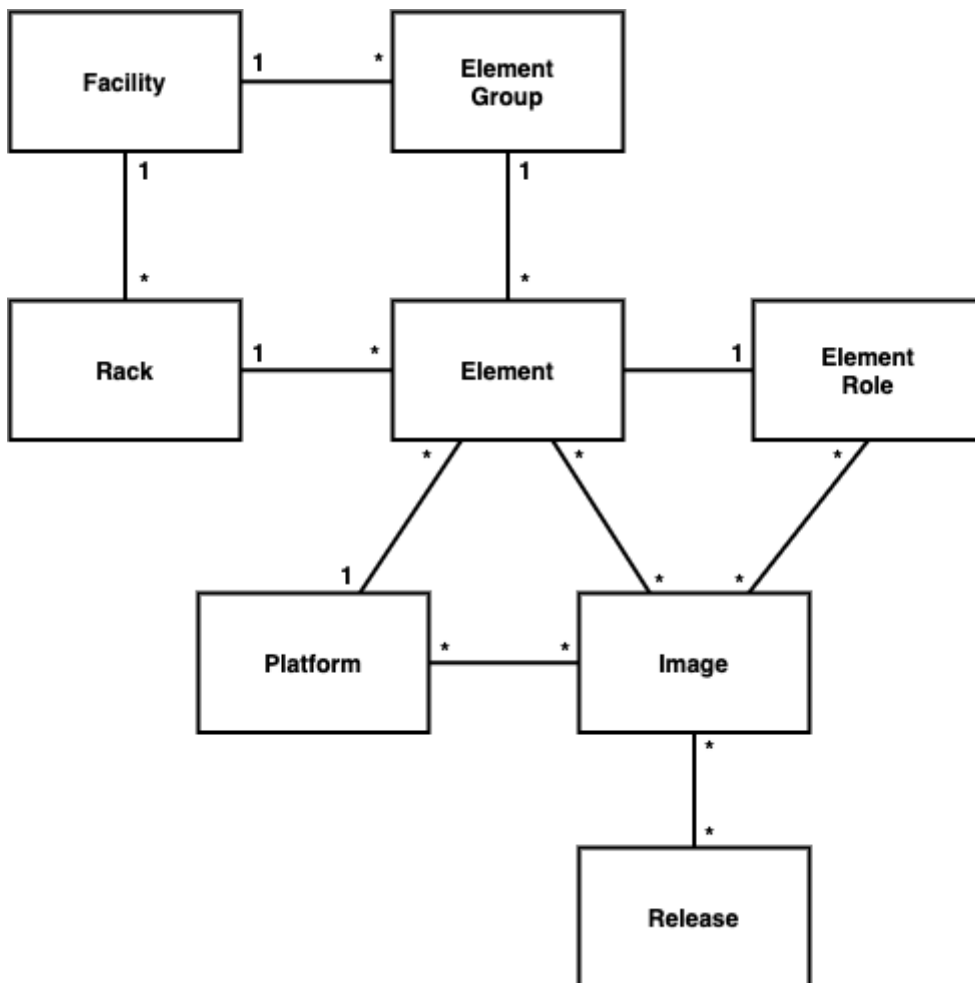


1. RBFS uses the resource inventory API to register itself in RBMS. RBFS reports the running services, physical interfaces, discovered neighbors, logical interfaces, hardware module information, information about the installed software image and the current configuration, including configuration changes, to RBMS.
2. The RBMS UI uses the resource inventory API to enable an operator to view and manage resource inventory records.
3. The OSS IT, or any other customer IT system can use the REST API to manage resource inventory records. It is also feasible to subscribe for events to get notified when a resource inventory record changes.

Data model

The resource inventory data is also the basis for the network management applications provided by RBMS. The data model is generalized allowing you to define additional roles to store custom elements, for example, to simplify automation of network management operations by storing the required data in the same database.

The main entities of the resource inventory and their relationships are shown below:



Element, element role and platform

The element forms the cornerstone of the resource inventory. Every element has a certain role, like a spine or a leaf switch of a fabric for example. The element role describes the function of an element in the network.

The resource inventory stores the following information per element:

- General settings including the element role, element name, an optional element alias, the operational and administrative state, the serial number, the hardware platform, an optional asset ID and management interfaces like SSH or REST API access.
- List of physical interfaces and their operational state
- List of logical interfaces and their operational state
- Information about the installed software image
- Installed services including their current state
- Element hardware modules
- Element configurations including a configuration history
- Environments containing custom data used to generate the element configuration.

The platform provides information about the hardware platform vendor, model and chipset. This information is needed to discover the software images that can be installed on this element.

The element has one of the following administrative states:

Administrative State	Description
NEW	A new element that has not yet been installed or enabled in the network.
ACTIVE	An element that has been installed in the network and is supposed to be up and running.
RETIRED	An element that is about to being removed from the network.

The following operational states exist:

Operational State	Description
DOWN	The element is down.
UP	The element is up.

Operational State	Description
DETACHED	RBMS did not receive heartbeats from the element and declares it as detached from RBMS.
MAINTENANCE	The element is currently in maintenance.

Image

The image provides information about the images being eligible for deployment. RBMS supports propose-build image that can be installed on certain element roles and platforms only. RBMS includes a image lifecycle management with the following states:

State	Description
NEW	A new image that is not yet eligible for deployment.
CANDIDATE	A candidate image to become the next release. The image is eligible for deployment but not installed by default.
RELEASE	The image that get installed by default.
SUPERSEDED	A former release image that has been replaced by another release image.
REVOKED	An image that has been revoked and must not be installed any more. A revoked image retains in the intenvory for the sake of documentation.

Element Group

Each element belongs to an element group. Elements of the same element group form a logical unit inside the network. For example, a pod of an access network is modelled as element group.

Facility and racks

The facility describes a network facility including its physical location. Moreover the racks installed at a facility can be specified. This also includes the locations of elements in a rack.

Managing Pods

Filtering Pod list by Pod name

To filter Pod list by Pod name

1. Click the **Inventory** tab. The Pods page appears by default.
2. In the **Filter** text box, specify the filter criteria and then click **Filter**. The Pods that match the filter criteria appear.

The screenshot shows the 'Inventory' page in the rtbrick application. The main heading is 'Inventory' with the subtitle 'Manage network elements'. There are tabs for 'Images', 'Inventory', 'Metrics', 'Jobs', 'Logs', and 'Administration'. A 'Logout' button is in the top right. A left sidebar contains a menu with 'Inventory' (expanded), 'Pods', 'Elements', 'Interfaces', 'Facilities', 'Racks', 'DNS Zones', and 'Administration'. The 'Pods' section is active, showing an overview of all registered pods. A filter input field is present with a 'Filter' button. Below the filter, a table lists pods with the following data:

Pod	Elements	Tags	Description
blr	4 active, 4 in total	BLR, lab	BLR lab environment
chris	1 active, 4 in total		
karthik	1 active, 1 in total		blr
pod1	6 active, 6 in total	NBG, lab	Pod1 in NBG lab
pod2	6 active, 6 in total	NBG, lab	Pod2 in NBG lab.

At the bottom right of the table, it says '5 pod(s) found.'

Creating Pods

To create a Pod

1. Click the **Inventory** tab. The list of Pods appears by default.
2. In the **Pods** page, click the **Add pod**.

Inventory

Manage network elements



Images Inventory Metrics Jobs Administration

Logout

Inventory –

- Pods**
- Elements
- Interfaces
- Facilities
- Racks
- DNS Zones

Administration +

Pods > New Pod

New Pod

Add a new pod

Settings

Pod Name

The name of the pod

Description

A brief description of this pod

3. In the **Pod Name** field, specify the name of the Pod.
4. In the **Description** field, specify a brief description for the Pod.
5. In the **Tags** field (optional), specify the tags that can be used to categorise the Pod.
6. Click **Add Pod**.

Modifying Pod settings

To modify the Pod settings

1. Click the **Inventory** tab. The list of Pods appear by default.
2. In the **Pods** group box, click the Pod that you want to modify. The **Pod Settings** page appears.

Pod Settings

Pod Location

Elements

Link State Graph

Administration +

Name

pod1

The name of the pod

Facility Name

Change facility

The main facility where this pod is installed at

Description

Pod1 in NBG lab

An optional description of the pod

Tags

NBG - lab - +

Optional tags to categorize the pod

Remove pod Save settings

3. Make the necessary updates and click **Save settings**.

Removing Pods

To remove a Pod

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to remove. The **Pod Settings** page appears.

Pod Settings

[Pod Location](#)

[Elements](#)

[Link State Graph](#)

Administration +

Name

The name of the pod

Facility Name

 Change facility

The main facility where this pod is installed at

Description

An optional description of the pod

Tags

+

Optional tags to categorize the pod

Remove pod
Save settings

Viewing Pod location

To view the facility where a Pod is installed

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod whose facility you want to view.
3. Click **Pod location** in the left navigation pane. The **Pod Facility** page appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

blr

Pod -

[Pod Settings](#)

Pod Location

[Elements](#)

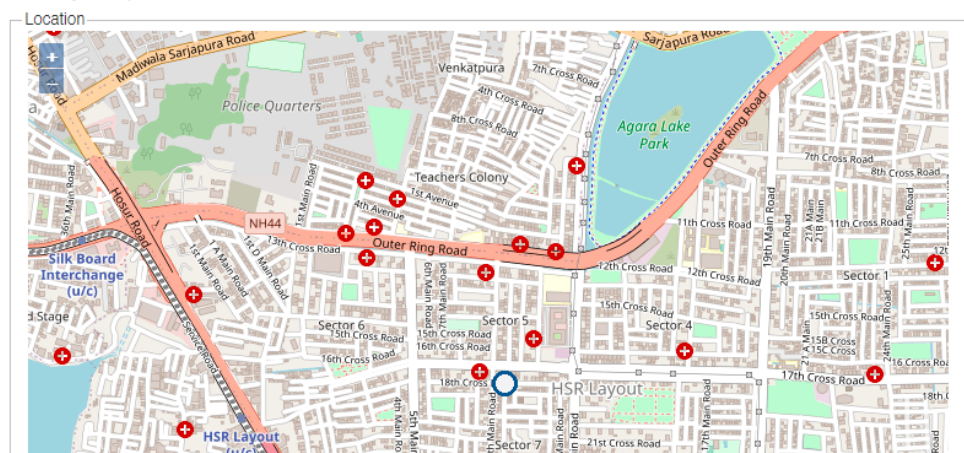
[Link State Graph](#)

Administration +

Pods > blr > Facility

Pod Facility

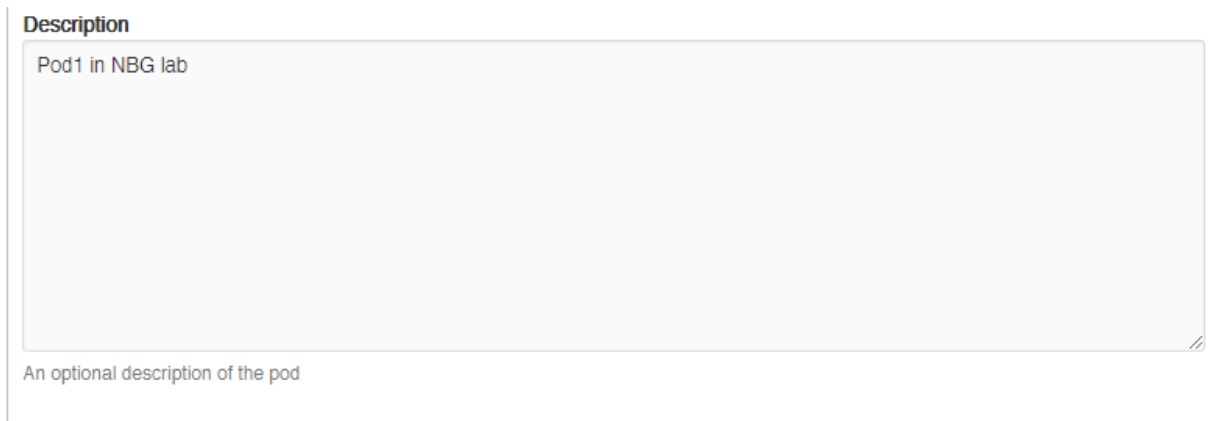
The facility where pod blr is installed at.



Describing Pod racks

To describe the Pod racks

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to modify. The **Pod Settings** page appears.
3. Specify the description for Pod rack.



The screenshot shows a form titled "Description" with a text input field. The field contains the text "Pod1 in NBG lab". Below the field, there is a small icon of a pencil. At the bottom of the form, there is a placeholder text: "An optional description of the pod".

Adding element to Pod

To add element to Pod

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to modify. The **Pod Settings** page appears.
3. Click **Elements** from the left navigation pane. The **Elements** page appears.
4. Click **Add element**.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Administration Logout

Inventory +

pod1

Pod -

- Pod Settings
- Pod Location
- Elements**
- Link State Graph

Administration +

[Elements](#) > [New Element](#)

New Element

Add a new element to pod1

Element Settings

Element Name

The unique name of the element

Element Alias

An optional unique element alias

Element Role

Select the role of the element

Hardware Platform

Select the hardware platform of the element

Listing elements of a Pod

To view the list of element in Pod

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to modify. The **Pod Settings** page appears.
3. Click **Elements** from the left navigation pane. The **Elements** page appears and the list of elements in the selected Pod appear.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Administration Logout

Inventory +

pod1

Pod -

- Pod Settings
- Pod Location
- Elements**
- Link State Graph

Administration +

[Pods](#) > [Elements](#)

Elements

Elements of pod pod1

Elements

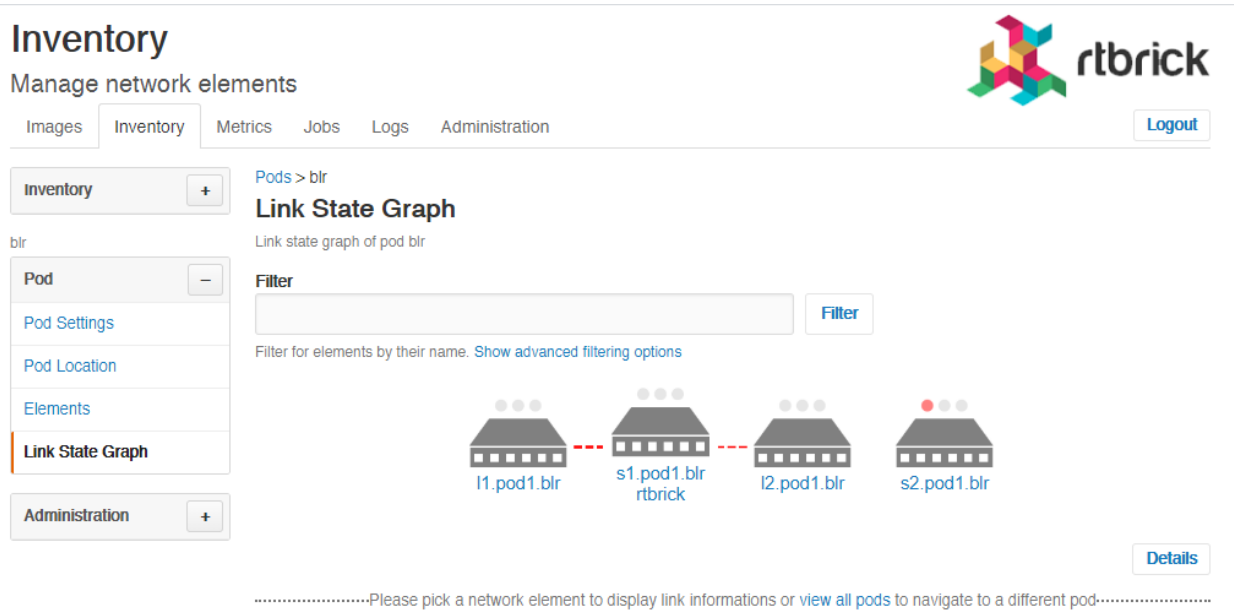
Name	Alias	Adm.-State	Op.-State	Tags	Description
bl1.pod1.nbg2	bl1.pod1	ACTIVE	DETACHED	NBG lab pod1	Border leaf 1 in pod 1 in NBG lab.
bl2.pod1.nbg2	bl2.pod1	ACTIVE	DETACHED	NBG lab pod1	Border leaf 2 in pod 1 in NBG lab
l1.pod1.nbg2	l1.pod1	ACTIVE	DETACHED	NBG lab pod1	Access leaf 1 in pod 1 in NBG lab
l2.pod1.nbg2	l2.pod1	ACTIVE	DETACHED	NBG lab pod1	Access leaf 2 in pod1 in NBG lab.
s1.pod1.nbg2	s1.pod1	ACTIVE	DETACHED	NBG lab pod1	Spine 1 in pod 1 in NBG lab.
s2.pod1.nbg2	s2.pod1	ACTIVE	DETACHED	NBG lab pod1	Spine 2 in pod1 in NBG lab.

[Add element](#)

Viewing Link State Graph

To view the link state graph of Pod

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to view. The **Pod Settings** page appears.
3. Click **Link State Graph** in the left navigation pane. The **Link State Graph** page appears.



4. On the **Link State Graph** page, select a network element to display link information.
5. Click **Details** to view the detailed information of network elements.

Inventory

Manage network elements



Images Inventory Metrics Jobs Logs Administration Logout

Inventory + Pods > blr

Link State Graph

Link state graph of pod blr

blr

Pod -

Pod Settings

Pod Location

Elements

Link State Graph

Administration +

Filter Filter

Filter for elements by their name. [Show advanced filtering options](#)

DETACHED
Access Leaf
f1.pod1.blr
EC5916-54XK

DETACHED
Access Leaf
s1.pod1.blr rtbrick
EC5916-54XK

DETACHED
Access Leaf
i2.pod1.blr
EC5916-54XK

DOWN
Access Leaf
s2.pod1.blr
EC5916-54XK

Overview

.....Please pick a network element to display link informations or [view all pods](#) to navigate to a different pod.....

Viewing Pod Topology

The topology view provides the following functionalities:

- Visualizes the Pod link-state graph
- Visualizes connected Pods
- Provides quick access to most relevant information
 - Alerts
 - Events
 - Established links

To view the Pod topology

1. Click the **Inventory** tab. The list of Pods appear.
2. In the **Pods** group box, click the Pod that you want to view. The **Pod Settings** page appears.
3. Click **Link State Graph** in the left navigation pane. The **Link State Graph** page appears.
4. Click the **view all pods** link that is displayed along the bottom of the page. The **Topology** page appears.

Inventory



Manage network elements

Images Inventory Metrics Jobs Logs Administration

Logout

Inventory +

Pod -

- Pod Settings
- Pod Location
- Elements
- Link State Graph**

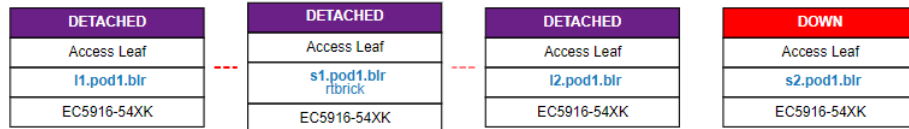
Administration +

Link State Graph

Link state graph of pod blr

Filter Filter

Filter for elements by their name. [Show advanced filtering options](#)



Overview

.....Please pick a network element to display link informations or [view all pods](#) to navigate to a different pod.....

The **Topology** page displays the overview of the Pod topology in a graphical interface along with a table that shows the list of Pods deployed in the network.

Managing Elements

To view the list of all registered elements

1. Click the **Inventory** tab, and then click **Elements** in the left navigation pane. The list of registered elements appear.

Inventory



Manage network elements

Images Inventory Metrics Jobs Logs Administration

Logout

Inventory -

- Pods
- Elements**
- Interfaces
- Facilities
- Racks
- DNS Zones

Administration +

Elements

Overview of all registered elements.

Expression Filter

Select the property to filter elements for. [Show advanced filtering options](#)

Pod	Element	Alias	Adm.-State	Tags	Description
blr	i1.pod1.blr		ACTIVE	BLR RBMS lab	
blr	i2.pod1.blr		ACTIVE	BLR lab	
blr	s1.pod1.blr		ACTIVE	BLR lab	
blr	s2.pod1.blr		ACTIVE	BLR lab	
chris	chris		NEW		
chris	cspine		ACTIVE		
chris	demo	demo	ACTIVE		

Assigning roles to elements

1. Click the **Inventory** tab, and then click **Elements** in the left navigation pane. The list of registered elements appear.
2. Click the name of the element that you want view.
3. In the **Element Role** box, select the role you want to use. The following roles are available:

Access Leaf- Access leaf fabric switch

Border Leaf - Border leaf fabric switch

Spine - Spine fabric switch

Viewing Physical Interfaces

The physical interface summary shows the information of a physical interface and its logical interfaces. The view might be augmented with telemetry data (for example, current interface utilization, log messages).

To view the list of physical interfaces

1. Click the **Inventory** tab.
2. In the left navigation pane, click **Interfaces**. The **Physical Interfaces** page appears.

Inventory
Manage network elements

Images | **Inventory** | Metrics | Jobs | Logs | Administration Logout

Inventory [-]

- Pods
- Elements
- Interfaces**
- Facilities
- Racks
- DNS Zones

Administration [+]

Physical Interfaces

Search for physical interfaces

Filter Filter

Filter physical interfaces by element name or description

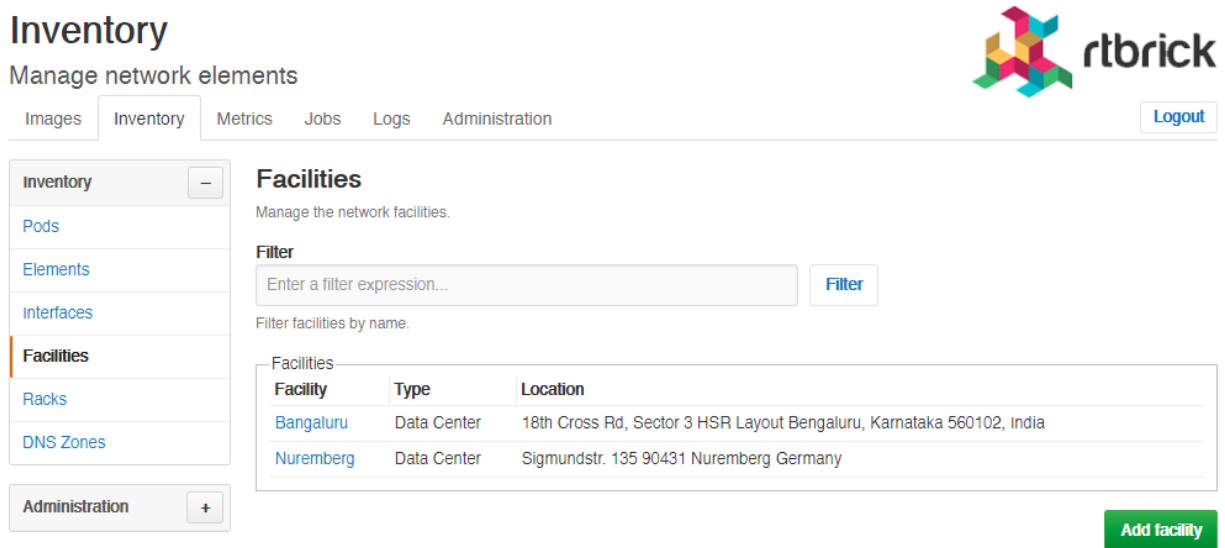
Element	Interface	Description	Adm.-State	Op.-State
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/17	ifp-0/0/17	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/15	ifp-0/0/15	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/26	ifp-0/0/26	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/41	ifp-0/0/41	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/54	ifp-0/0/54	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/11	ifp-0/0/11	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/14	ifp-0/0/14	UP	DOWN
bl1_pod1.nbg2 (bl1.pod1)	ifp-0/0/24	ifp-0/0/24	UP	DOWN

Managing Facilities

Viewing list of facilities

To view the list of facilities

1. Click the **Inventory** tab.
2. In the left navigation pane, click **Facilities**. The list of available facilities appear.



The screenshot shows the 'Inventory' section of the rtbrick interface. The 'Facilities' tab is selected in the left navigation pane. The main content area displays the 'Facilities' section, which includes a filter input field and a table of facilities.

Facility	Type	Location
Bangaluru	Data Center	18th Cross Rd, Sector 3 HSR Layout Bengaluru, Karnataka 560102, India
Nuremberg	Data Center	Sigmundstr. 135 90431 Nuremberg Germany

An 'Add facility' button is visible in the bottom right corner of the facilities section.

Creating a Facility

To create a facility

1. Click the **Inventory** tab.
2. In the left navigation pane, click **Facilities**. The list of available facilities appear.
3. Click **Add facility**.
4. Specify the facility settings such as facility name, type, and description.
5. Click **Add facility**.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory -

- Pods
- Elements
- Interfaces
- Facilities**
- Racks
- DNS Zones

Administration +

Facilities > New Facility

New Facility

Manage facility settings.

Facility Settings

Facility Name

The name of the facility

Facility Type

Data Center ▾

Select the type of this facility

Description

Managing Racks

To view the list of racks

1. Click the **Inventory** tab.
2. In the left navigation pane, click **Racks**. The list of available racks appear.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory -

- Pods
- Elements
- Interfaces
- Facilities
- Racks**
- DNS Zones

Administration +

Racks

Manage the racks containing the elements.

Filter

 Filter

Filter racks by name.

Rack	Facility	State	Description
nbg1	Nuremberg	UNKNOWN	Hetzner NBG

Add rack

Creating a Rack

To create a rack

1. Click the **Inventory** tab.
2. In the left navigation pane, click **Racks**. The list of available racks appear.
3. Click **Add rack**.

Inventory

Manage network elements



Images Inventory Metrics Jobs Logs Administration Logout

Inventory -

Pods

Elements

Interfaces

Facilities

Racks

DNS Zones

Administration +

Racks > New Rack

New Rack

Describe new rack.

General Settings

Rack Name

The name of the rack

Administrative State

Active ▾

The administrative state of this rack

Description

4. Specify the rack settings such as rack name, administration state, description, rack dimensions, and other additional information.
5. Click **Add rack**.

Managing DNS Zones

To view the list of DNS zones

1. Click the **Inventory** tab.
2. In the left navigation pane, click **DNS Zones**. The list of available DNS Zones appear.

Creating a DNS Zone

To create a DNS zone

1. Click the **Inventory** tab.
2. In the left navigation pane, click **DNS Zones**. The list of available DNS zones appear.
3. Click **Add DNZ Zone**.
4. Specify the rack settings such as canonical DNS zone name and description.
5. Click **Add zone**.

Inventory Administration

To configure the inventory administration settings in the **Inventory** page, click the plus sign (+) next to **Administration**.

Managing element platforms

To list down the known hardware platforms

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**.
3. Click **Platforms**. The list of hardware platforms appear.

Creating element platforms

To create a hardware platform

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**.
3. Click **Platforms**. The list of hardware platforms appear.
4. Click **Add platform**.
5. Specify the settings and dimensions of the hardware platform.
6. Click **Save platform**.

Removing element platforms

To remove an element platform

1. Click the **Inventory** tab.
2. Click the plus sign (+) next to **Administration**, and then click **Platforms**. The list of hardware platforms appear.
3. Click the name of the element platform that you want delete. The **Platform Settings** page displays the settings of the element platform.
4. Click **Remove**.

Managing Element Roles

To list down the existing element roles

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**.
3. Click **Roles**. The list of existing element roles appear.

Creating Element Roles

To create an element role

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**. The list of existing element roles appear.
3. Click **Roles**. The list of existing element roles appear.
4. Click **Add role**
5. Specify the role settings.
6. Click **Save role**.

Removing Element Roles

To remove an element role

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**. The list of existing element roles appear.
3. Click **Roles**. The list of existing element roles appear.
4. Click the name of the element role that you want delete. The next page displays the settings of the element role.
5. Click **Remove role**.

Managing Templates

To list down the existing templates

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**.
3. Click **Templates**. The list of existing element templates appear.

Creating Templates

To create a template

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**. The list of existing element templates appear.
3. Click **Add templates**
4. Specify the template settings such as template name and description.
5. Click **Save template**.

Removing Templates

To remove a template

1. Click the **Inventory** tab.
2. click the plus sign (+) next to **Administration**. The list of existing element templates appear.
3. Click **Templates**. The list of existing element templates appear.
4. Click the name of the template that you want delete. The next page displays the settings of the template.
5. Click **Remove template**.

3. Managing Software Images

RBMS manages a list of software images that are currently installed on the switches in the network or eligible for being installed. The image lifecycle encompasses the following states:

- **NEW**, a new image that has been uploaded to RBMS but is not yet eligible for deployment. Typically the integrity of a new image needs to be verified before it becomes eligible for deployment.
- **CANDIDATE**, a candidate image is eligible for deployment. It is a candidate to become the new release image if it passes all tests in the network.
- **RELEASE**, the release image gets installed by default.
- **SUPERSEDED**, a superseded image is a former release image which has been replaced by a newer release.
- **REVOKED**, a revoked image must not be installed on switches in the network.

Superseded and revoked images are kept in the resource inventory for the sake of documentation and to track on how many switches superseded or revoked images are installed.

The `rtb-image` tool provides an option to register new image in RBMS. The new image can be installed on a switch via Zero-Touch Provisioning (ZTP).

Viewing all registered images

To view all registered images

- Click the **Images** tab. The **Images** page appears, showing the list of registered images.

Images
Manage software images

Images Inventory Metrics Jobs Administration Logout

Images –

Images
Releases

Images
Explore available images

Filter
Enter a filter expression Filter

Search for images by adding a regular expression to filter for image name, element role or platform name. [Show advanced filtering options](#)

Name	State	Type	Chipset	Version	Roles	Element
leer	CANDIDATE	LXC	105	20.5.0-g8daily.20200602165356-Bmaster.C7179e422		*
rtbrick-onl-installer/rtbrick-onl-installer-accessleaf-qmx-20.5.0-g8daily.20200602171027+Bmaster.Ce64bd03c	NEW	onl-installer	qmx	20.5.0-g8daily.20200602171027+Bmaster.Ce64bd03c	spine accessleaf	*
rtbrick-onl-installer/rtbrick-onl-installer-accessleaf-qmx-20.5.0-g8daily.20200603030630+Bmaster.Ce64bd03c	NEW	onl-installer	qmx	20.5.0-g8daily.20200603030630+Bmaster.Ce64bd03c	spine accessleaf	*

The **Images** page displays the following fields:

Field	Description
Name	Name of the software image
State	Indicates the image state such as New, Candidate, Release, Suspended, Revoked. For more information, see the Image Lifecycle Management section.
Type	Type of the software image such as LXC or ONL
Chipset	Specifies the supported hardware name
Version	Version of the software image
Roles	The image is deployed on the specified element roles.
Element	Name of the registered element

Viewing image details

To view the details of an image

1. Click the **Images** tab. The list of registered images appear.
2. In the **Images** group box, click the name of the image that you want to view. The image details page appears.

Images



Manage software images

Images Inventory Metrics Jobs Administration Logout

Images
+

Images > Image rtbrick-onl-installer/rtbrick-onl-installer-accessleaf-qmx-20.5.0-g8daily.20200602171027+Bmaster.Ce64bd03c onl-installer image

rtbrick-onl-installer/rtbrick-onl-installer-accessleaf-qmx-20.5.0-g8daily.20200602171027+Bmaster.Ce64bd03c

Review image metadata

General Settings

Image ID

Image UUID (readonly).

Organization

The organization that issued the image

Image Type

The image type (readonly)

Image Name

Image Information -

- Metadata
- State
- Applications
- Packages
- Utilization

This page displays the following fields:

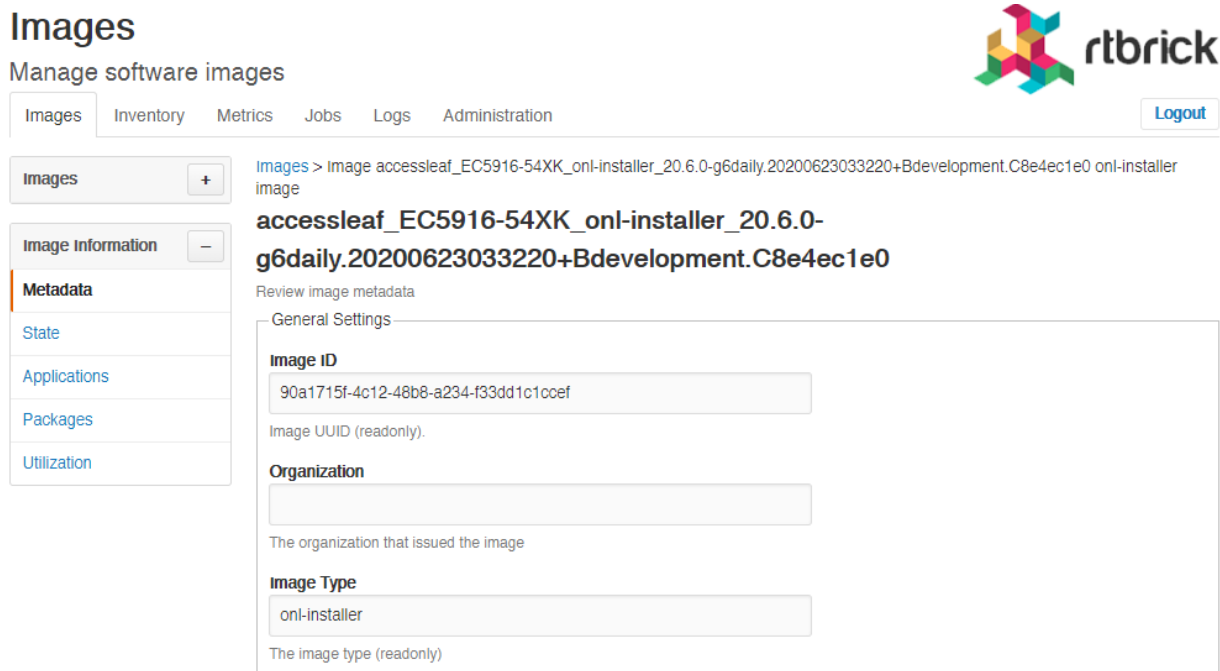
Field	Description
Image ID	Image UUID (read-only)
Organization	The organization that issued the image
Image Type	The image type (read-only)
Image Name	The unique image name
Image Version	Image version (read-only)
Category	Optional image category
Description	Optional description of the image
Platform Chipset	The chipset this image is build for (read-only)
Element Roles	This image can be deployed on the selected element roles
Supported Platforms	The list of platforms on which the image can be installed

Updating image metadata

To update the metadata of an image

1. Click the **Images** tab. The list of registered images appear.

2. In the **Images** group box, click the name of the image whose metadata you want to update. The image details page appears.
3. On the left navigation pane, click **Metadata**.



The screenshot shows the 'Images' management interface. At the top, there's a navigation bar with 'Images', 'Inventory', 'Metrics', 'Jobs', 'Logs', and 'Administration'. A 'Logout' button is on the right. Below the navigation bar, there's a breadcrumb trail: 'Images > Image accessleaf_EC5916-54XK_onl-installer_20.6.0-g6daily.20200623033220+Bdevelopment.C8e4ec1e0 onl-installer image'. The main content area displays the image name: 'accessleaf_EC5916-54XK_onl-installer_20.6.0-g6daily.20200623033220+Bdevelopment.C8e4ec1e0'. Below the name, it says 'Review image metadata'. The 'General Settings' section contains three fields: 'Image ID' with the value '90a1715f-4c12-48b8-a234-f33dd1c1ccef', 'Image UUID (readonly)', 'Organization' (empty), and 'Image Type' with the value 'onl-installer'. The left sidebar has a 'Metadata' tab selected, along with 'State', 'Applications', 'Packages', and 'Utilization'.

4. Update image metadata such as image ID, type, name, and version.
5. Click **Save settings**.

Image Lifecycle Management

The image lifecycle defines five different image states. A new image is an image that has not yet been approved to be eligible for a deployment. The release image is deployed by default. A candidate image is intended to become the new default image, once the candidate has passed all necessary tests. The existing default image is superseded by the new default image. A revoked image must not be used any longer because it contains severe errors.

1. Click the **Images** tab. The list of registered images appear.
2. Click the name of the image that you want to view.
3. Click **State** in the left navigation pane.
4. From the following list of states, select the state that you want to apply.

Images



Manage software images

Images Inventory Metrics Jobs Logs Administration

Logout

Images +

images > accessleaf_EC5916-54XK_lxd_20.5.0-g8daily.20200602165356+Bmaster.C7179e422 lxd image

Image accessleaf_EC5916-54XK_lxd_20.5.0-g8daily.20200602165356+Bmaster.C7179e422

Image Information -

Metadata

State

Applications

Packages

Utilization

Image State

The image lifecycle defines five different image states. A *new* image is an image that has not yet been approved to be eligible for a deployment. The *release* image is deployed by default. A *candidate* image is intended to become the *new default* image, once the candidate has passed all necessary tests. The existing default image is *superseded* by the new default image. A *revoked* image must not be used any longer because it contains severe errors.

- New**
Newly registered image
- Candidate**
Candidate for the next release image
- Release**
The image to be installed by default
- Superseded**
A former release image that has been superseded by a newer image version
- Revoked**
The image must not be used anymore.

Save state

- **New:** Newly registered image
- **Candidate:** Candidate for the next release image
- **Release:** The image to be installed by default
- **Superseded:** A former release image that has been superseded by a newer image version
- **Revoked:** The image must not be used anymore.

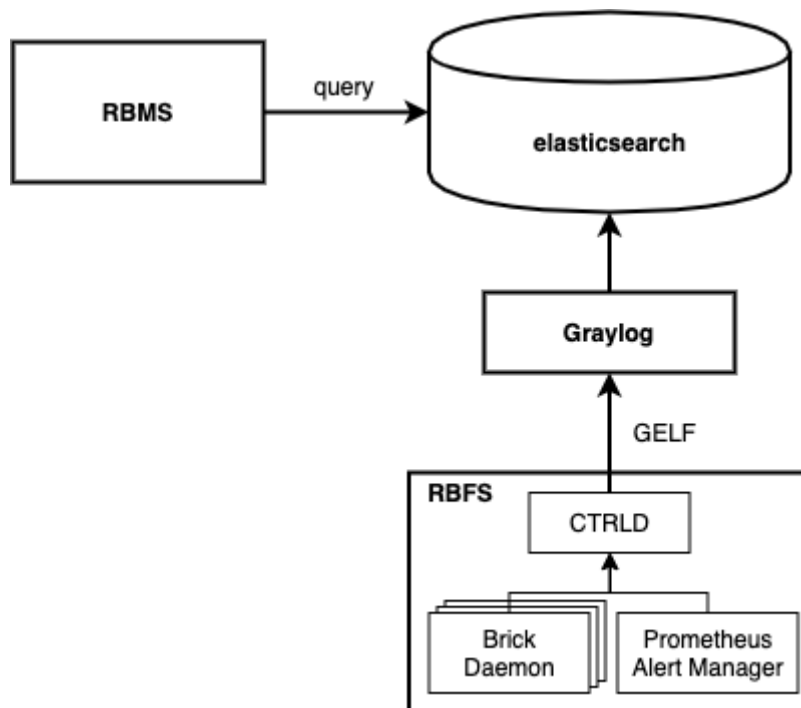
5. Click **Save state**.

4. Managing Logs

Introduction

In order to understand the RBMS log viewer it is key to understand the RBFS logging concept. RBFS stores log information in Brick Data Store (BDS) tables. The BDS is an in-memory database developed by RtBrick and optimized for the networking domain. The BDS log tables contain only the raw data of a log event. Exporters pass the raw data to a template string to create a human friendly log message.

By default RBFS exports log messages in GELF format. The [Graylog Extended Logging Format \(GELF\)](#) is a JSON representation of the Syslog protocol, with the option to add custom fields.



The CTRLD forms the egress node for all GELF messages. CTRLD receives log messages from brick daemons, augments the GELF message with the element name, element role, serial number and pod name and forwards it to the configured GELF endpoint. In addition, CTRLD receives all notification of the Prometheus Alert Manager running on the switch and translates them to GELF messages. Last but not least, CTRLD generates GELF messages to log events.

All messages are send to a configured GELF endpoint. The GELF endpoint stores the data in a central log database. The GELF message is already a structured

message. Thus the endpoint does not have to create a log message into a structured record.

RBMS queries log events from the log management system to provide quick access to log messages. In addition, RBMS links all log messages to the inventory records to quickly inspect the state of an element.

Viewing log events

The log viewer reads log records from the Elasticsearch database. The query is formed from the resource inventory data and can be amended by the operator to fine-tune the result set. You can inspect the details of a log message in the RBMS UI.

To view the list of logs

1. Click the **Logs** tab. The list of all log events occurred in the network within the last five minutes having at least *WARNING* severity appears.

The screenshot shows the RBMS interface for viewing logs. At the top, there is a navigation bar with tabs for Images, Inventory, Metrics, Jobs, Logs, and Administration. The 'Logs' tab is selected. Below the navigation bar, there is a section titled 'Events' with a sub-header 'Query all log events occurred in the network'. A filter interface is displayed, allowing users to search for log events based on Time Range (Search in last 5 minutes), Severity (Warning), and a custom Filter expression. There are also fields for Pod Name, Element Name, Module Name, and a Limit (100 events). Below the filter interface, a table of log events is shown. The table has columns for Issued At, Pod Element, and Module Message. A single event is listed with a red 'ERR' icon, indicating an error.

!	Issued At	Pod Element	Module Message
ERR	28-MAY-2020 11:30:48.643	rtbrick-pod rtbrick	lldpv2 LLDP interface ifp-0/0/26: Failed to handle update of lldp interface object

2. Click the timestamp of the event that you want to view.

Filtering log events

To filter the list of logs

1. Click the **Logs** tab. The list of all log events occurred in the network appear.

Logs

View logs messages from the switches



Images Inventory Metrics Jobs **Logs** Administration [Logout](#)

Filter

Time Range Search in last 15 minutes ▾ <small>Select time range when the event occurred</small>	Severity Warning ▾ <small>Select minimum severity</small>	Pod Name pod1.blr <small>Filter for log events of the specified pod</small>
--	--	--

Filter

`element_name:"l1.pod1.blr"` [Filter](#)

Enter filter query to search for certain elements only.

2. Specify the filter criteria to filter the log events.

5. Metric Management

Overview

In order to understand metric processing in RBMS it is key to understand how metrics are sampled on RBFS.

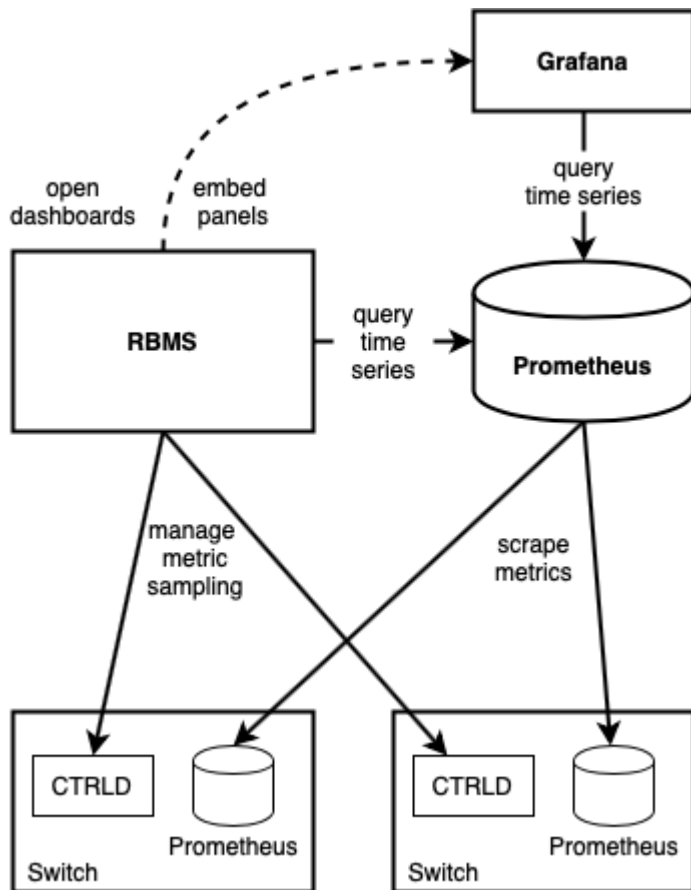
RBFS runs [Prometheus](#), an open-source monitoring tool with built-in time series database, to sample metrics from the Brick Data Store (BDS). The BDS is an in-memory database developed by RtBrick and optimized for networking use cases. All configuration and state information is stored in BDS. BDS can be configured to expose certain BDS attributes as metrics in the [Prometheus Exposition Format](#). Prometheus periodically samples the exposed values to create a time series over them. The [Prometheus Query Language](#) allows querying the time series data and to apply transformations and statistical computations.

Prometheus includes an alert manager to scan time series for alert conditions and to create a notification whenever an alert condition is satisfied. Since Prometheus runs on the switch the alert conditions are also evaluated on the switch.

All metrics and alerts that shall be enabled permanently by default are configured in the running configuration of the switch. CTRLD allows enabling and disabling metrics on-demand.

The main goal of RBMS is to provide quick access to metrics and time series visualization. RBMS runs a Prometheus instance to federate metrics from the Prometheus instances on the switches. [Grafana](#), an open-source visualization platform, is used to visualize time series data.

The figure below summarizes how RBMS interfaces with Grafana and Prometheus.



The RBMS UI provides quick access to Grafana dashboards and embeds dashboard panels in the RBMS UI. In addition, RBMS queries Prometheus based on the configured metrics and the inventory data to provide quick access to the current metric values.



The Metrics Sampling and Monitoring Tutorial provides more details about the concepts outlined in this section.

Working with metrics

Viewing metrics

To view the metrics of an element

1. Click the **Inventory** tab. The list of pods appears.
2. In the pods group box, click on the elements link for pod that contains the element you want to view the metrics for. The list of elements appears.
3. In the elements group box, click on the element you want to view the metrics for.

- Click **Metrics** in the left navigation pane. The list of metrics and their current values appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

LAB-0

Pod +

accessleaf leaf2

- Element -
- Element Settings
- CTRLD
- Configuration
- Environments
- Location
- Dashboards
- Metrics**
- Images
- Services

Pods > LAB-0 > Elements > leaf2 > Metrics

Metrics snapshots from accessleaf leaf2

Metrics sampled from accessleaf leaf2 (leaf2)

Metric	Values from 11-JUL-2020 20:55:31.000
Chassis Temperature	LM75-2-49 24 °C graph
	LM75-3-4A 22.5 °C graph
	LM75-1-48 26.5 °C graph
	LM75-3-4B 22.5 °C graph
PSU Output Power	PSU-1 0 W graph
	PSU-2 139 W graph
CPU Core Temperature	CPU Core 37 °C graph
Interface Utilization	No data points.
Chassis Fan Speed	Chassis Fan - 6 8.7 krpm graph
	Chassis Fan - 1 8.3 krpm graph
	Chassis Fan - 2 8.4 krpm graph
	Chassis Fan - 3 8.5 krpm graph
	Chassis Fan - 4 8.4 krpm graph
	Chassis Fan - 5 8.4 krpm graph

[Configure metrics](#)

- Click on **graph** to view the time series of a metric.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Pods > LAB-0 > Elements > leaf2 > Metrics > chassis_temperature_millicelsius

Chassis Temperature leaf2

Metrics sampled from accessleaf leaf2 (leaf2)

LAB-0

Pod +

accessleaf leaf2

Element -

- Element Settings
- CTRLD
- Configuration
- Environments
- Location
- Dashboards
- Metrics**
- Images
- Services
- Physical Interfaces

	min	max	avg	current
Sensor LM75-1-48	26.000 °C	27.000 °C	26.452 °C	26.500 °C
Sensor LM75-2-49	23.500 °C	24.000 °C	23.857 °C	23.500 °C
Sensor LM75-3-4A	22.000 °C	23.000 °C	22.429 °C	22.500 °C
Sensor LM75-3-4B	22.000 °C	22.500 °C	22.286 °C	22.000 °C

[Show metric list](#)

Enabling a metric

To enable a metric, go to the metric list view

1. Click **Configure metrics**. A list of available metrics appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Pods > LAB-0 > Elements > leaf2 > Metrics > Configure

Manage metrics for element leaf2

Enable or disable metrics for element leaf2 (leaf2)

Metrics

Action	Metric	Unit
Disable	Chassis Temperature	°C
Disable	PSU Output Power	W
Disable	CPU Core Temperature	°C
Disable	Chassis Fan Speed	rpm
Enable	Interface Utilization	bps

2. Click **Enable** for all metrics you want to enable.

Disabling a metric

To disable a metric, go to the metric list view

1. Click **Configure metrics**. A list of available metrics appears.

Inventory
Manage network elements

Images | **Inventory** | Metrics | Jobs | Logs | Administration Logout

Inventory +

LAB-0

Pod +

accessleaf leaf2

Element -

- Element Settings
- CTRLD
- Configuration
- Environments
- Location
- Dashboards
- Metrics**

Pods > LAB-0 > Elements > leaf2 > Metrics > Configure

Manage metrics for element leaf2

Enable or disable metrics for element leaf2 (leaf2)

Action	Metric	Unit
Disable	Chassis Temperature	°C
Disable	PSU Output Power	W
Disable	CPU Core Temperature	°C
Disable	Chassis Fan Speed	rpm
Enable	Interface Utilization	bps

2. Click **Disable** for all metric you want to disable.

Viewing element dashboards

To open a dashboards of an element

1. Click the **Inventory** tab. The list of pods appears.
2. Click on **Elements** in the left navigation pane to show the list of known elements.
3. Click on the element for which you want to access a dashboard. Use the filter to search for a particular element.
4. Click on **Dashboards**. The list of available dashboards appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration [Logout](#)

Inventory +

pod1.blr

Pod +

accessleaf I1.pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration
- Environments
- Location
- Dashboards**
- Metrics

Dashboards

Access the dashboards available for this element.

Dashboards

List of dashboards available for this element.

Name	Category	Description
Chassis Overview	RBFS	Temperatures, Power Supply and Fan Speeds.
CPU Utilization	RBFS	CPU utilization overview

- Click on the **Dashboard** you would like to view. RBMS opens the dashboard in Grafana.

Managing metrics

Adding a new metric

To add a metric

- Click the **Metrics** tab. The list of metrics appears in the **Metrics** page.
- In the **Metrics** page, click the **Add metric** button. The **New Metric** page appears.

Metrics

Manage the metrics sampled from network devices



Images Inventory **Metrics** Jobs Logs Administration Logout

Metrics

[Metrics](#) > [New Metric](#)

New Metric

Add a new metric

General Settings

Metric Name

The name of the metric

Display Name

The optional display name of the metric. Defaults to the metric name if unspecified.

Unit

The unit of the metric

Scope

The scope of the metric

3. In the **Metric Name** field, assign a unique metric name.



The metric name should be in lower case. The name suffix should contain the metric unit. The name prefix should be used to group metrics. Use the `_` as delimiter. For example `psu_powerout_milliwatts` or `psu_voltageout_millivolts`

4. In the **Display Name** field, assign a descriptive metric name.

5. In the **Unit** field, specify the metric unit (e.g. °C, W, bps)

6. In the **Scope** field, select the scope of the metric.

The following metric scopes exist:

Metric Scope	Description
Element	The metric exists for the element (e.g. chassis temperature)
Physical Interface	The metric exists per physical interface (e.g. interface utilization)
Service	The metric exists per service or daemon (e.g. CPU and memory utilization of a daemon)

Logical Interface	The metric exists per logical interface.
-------------------	--

- Optionally restrict the metric to certain element roles and platforms.

Binding

Element Roles

accessleaf
borderleaf
spine
integrationtest

Restrict this metric to the selected element roles.

Platforms

EC5916-54XK
EC5916-54XKS
foo
Wedge-100BF-32X
Wedge-100BF-65X

Restrict this metric to the selected platforms.

Dismiss
Add metric

- Click **Add metric**.

Adding the CTRLD configuration

The *CTRLD* configuration contains the JSON message to enable the metric via CTRLD.

To add the CTRLD configuration

- Click **Configurations** in the left navigation pane. The list of configurations appears.
- Click **Add config**. The New Configuration view appears.
- Add the CTRLD JSON object to the JSON editor.

Metrics

Manage the metrics sampled from network devices



Metrics

Metrics > chassis_temperature_millicelsius > Configurations > New configuration

New chassis_temperature_millicelsius configuration

chassis_temperature_millicelsius

New configuration for metric chassis_temperature_millicelsius

- Metric Settings -
- Metric Settings
- Configurations**

```
Configuration
powered by ace
1  {
2  "metric_name": "chassis_temperature_millicelsius",
3  "table_name": "global.chassis_0.resource.sensor",
4  "bds_metric_type": "object-metric",
5  "metric_type": "gauge",
6  "metric_description": "Chassis temperature in millidegree celsius",
7  "attributes": [
8  {
9    "attribute_name": "temperature",
10   "labels": [
11   {
12     "label_name": "sensor",
13     "label_value": "resource_name",
14     "dynamic": true
15   }
16   ]
17 }
18 ],
19 "filters": [
20 {
21   "match_attribute_name": "resource_type",
22   "match_type": "exact",
23   "match_value": "thermal"
24 },
25 {
26   "match_attribute_name": "resource_name",
27   "match_type": "regular-expression",
28   "match_value": "LM.*"
29 }
30 ]
Ln:31 Col:2
```

General Settings

Configuration Name

The name of the configuration

Category

Optional configuration category.

Description

An optional configuration description



The **Metrics sampling and monitoring tutorial** provides more information about how to use CTRLD API to manage metrics.

4. In the **Configuration Name** field, enter **CTRLD**.

5. In the **Category** field, enter **RBFS**.
6. In the **Description** field, enter an optional description for the configuration.
7. Click **Add config** to add the new configuration.



The value of the `metric_name` attribute in the CTRLD JSON object must be equal to the metric name in RBMS to avoid trouble.

Adding the Prometheus configuration

The *Prometheus* configuration provides the Prometheus queries to fetch the metrics from Prometheus.

To add the Prometheus configuration.

1. Click **Configurations** in the left navigation pane. The list of configurations appears.
2. Click **Add config**. The New Configuration view appears.

Metrics

Manage the metrics sampled from network devices



The screenshot shows the RBMS web interface. At the top, there are navigation tabs: Images, Inventory, Metrics (selected), Jobs, Logs, and Administration. A 'Logout' button is in the top right. Below the tabs, the breadcrumb path is 'Metrics > chassis_temperature_millicelsius > Configurations > New configuration'. The main heading is 'New chassis_temperature_millicelsius configuration'. Below this, it says 'New configuration for metric chassis_temperature_millicelsius'. The main content area is a code editor titled 'Configuration' with a 'powered by ace' label. The code is as follows:

```

1 {"element": "chassis_temperature_millicelsius{element_name=\"{{element_name}}\"}/1000",
2   "labels": [
3     "sensor"
4   ]
5 }

```

At the bottom of the code editor, it shows 'Ln:3 Col:12'. On the left side, there is a sidebar with 'Metrics' selected, and a sub-menu for 'chassis_temperature_millicelsius' with options for 'Metric Settings' and 'Configurations'.

General Settings

Configuration Name

The name of the configuration

Category

Optional configuration category.

Description

Query chassis temperature from Prometheus.

An optional configuration description

Cancel
Add config

3. Add the Prometheus JSON object to the JSON editor.

The Prometheus JSON object contains the following properties:

Attribute	Description
element	The Prometheus query to fetch the metric values for an element. RBMS replaces {{element_name}} with the name of the selected element.
ifp	The Prometheus query to fetch the metric values for a physical interface. RBMS replaces {{element_name}} with the name of the selected element and {{ifp_name}} with the name of the selected physical interface.
ifl	The Prometheus query to fetch the metric values for a logical interface. RBMS replaces {{element_name}} with the name of the selected element and {{ifl_name}} with the name of the selected physical interface.

service	The Prometheus query to fetch the metric values for a physical interfaces. RBMS replaces <code>{{element_name}}</code> with the name of the selected element and <code>{{service_name}}</code> with the name of the selected physical interface.
labels	String array with the names of the labels that shall be displayed in the metrics list.

4. In the **Configuration Name** field, enter **Prometheus**.
5. In the **Category** field, enter **RBFS**.
6. In the **Description** field, enter an optional description for the configuration.
7. Click **Add configuration** to add the new configuration.

Removing a metric

To remove a metric from RBMS

1. Click on the **Metrics** tab. The list of metrics appears.
2. Click on the metric you want to remove.
3. Click on **Remove metric**. A confirmation dialog appears.
4. Click on **Confirm** to remove the metric. Check the **force** option to remove the metric including the metric configurations.

Managing dashboards

Registering Dashboards

Configuring the visualization platform

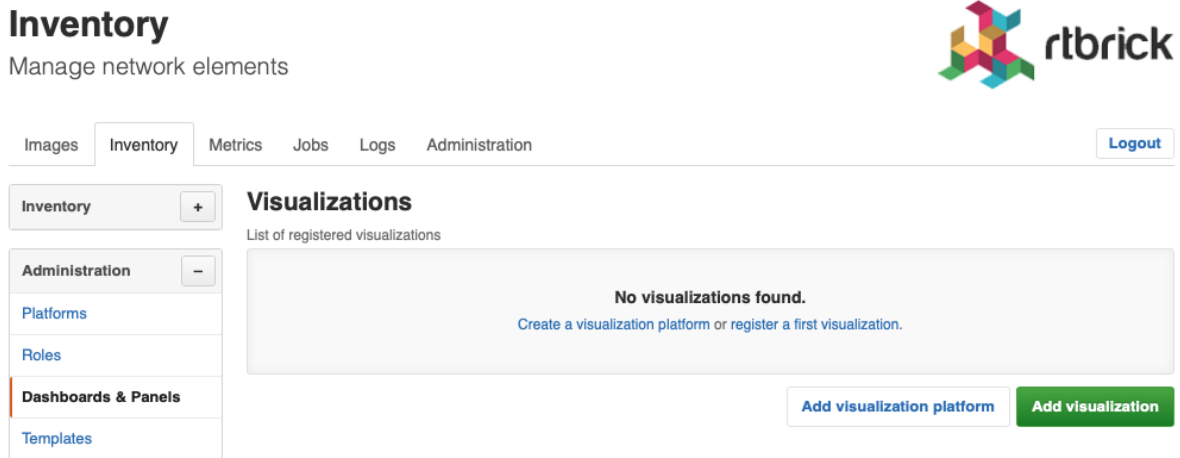


RBMS needs to know the visualization platform hosting the dashboard that shall be accessible from RBMS.

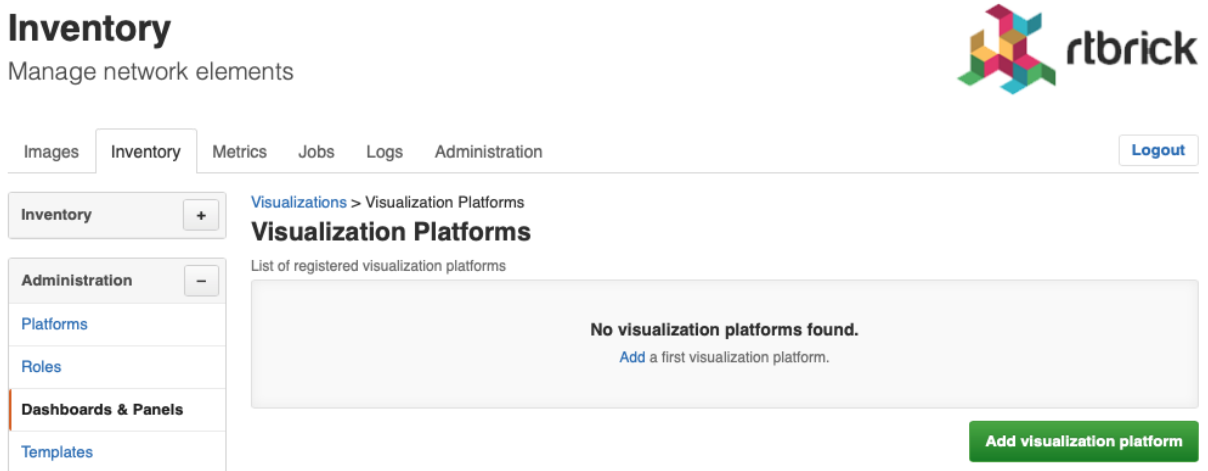
To register a visualization platform

1. Click the **Inventory** tab.

2. Click on **+** to expand the **Administration** menu.
3. Click on **Dashboard & Panels** in the navigation pane. The list of registered dashboards and panels appears.



4. Click on **Add visualization platform**. The list of known visualization platforms appears.



5. Click on **Add visualization platform** to add a new visualization platform. The New visualization platform appears.

Inventory

Manage network elements



Images Inventory Metrics Jobs Logs Administration Logout

Inventory +

Administration -

Platforms

Roles

Dashboards & Panels

Templates

Visualizations > Visualization Platforms > New Visualization Platform

New Visualization Platform

Add new visualization platform

Visualization Platform Settings

Visualization Platform Name

Grafana

The name of the visualization platform.

Origin

http://192.168.202.46:3000/

The location of the visualization server. This is the prefix for all registered visualizations.

Description

RBMS Grafana Instance

An optional description of the visualization platform.

Cancel
Add visualization platform

In the **Name** field, provide a unique visualization platform name. In the **Origin** field, provide the origin of the visualization. In the **Description** field, provide an optional description of the visualization platform.

Click **Add visualization platform** to add the visualization platform.

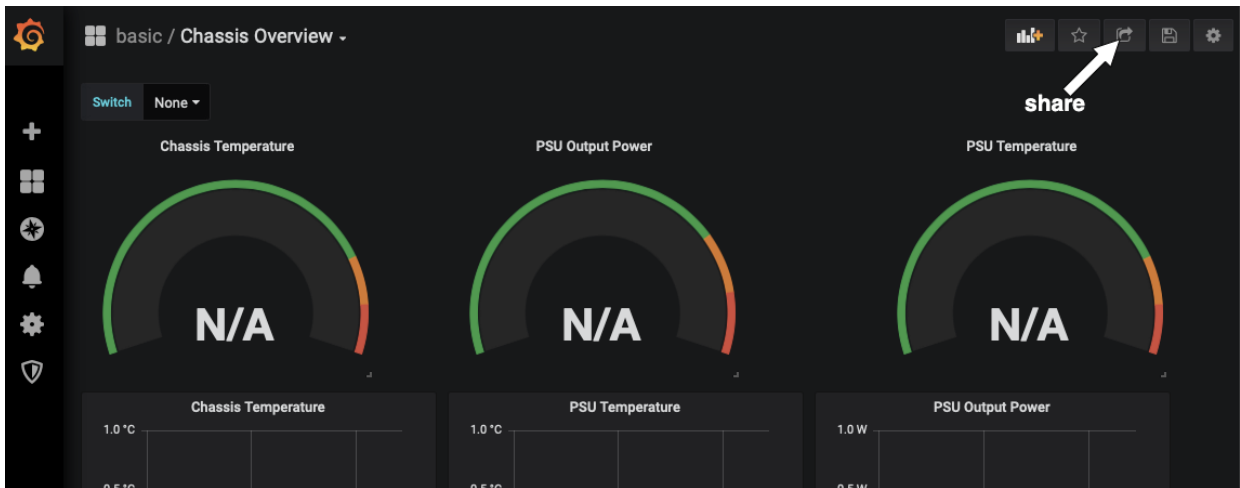
Registering a dashboard



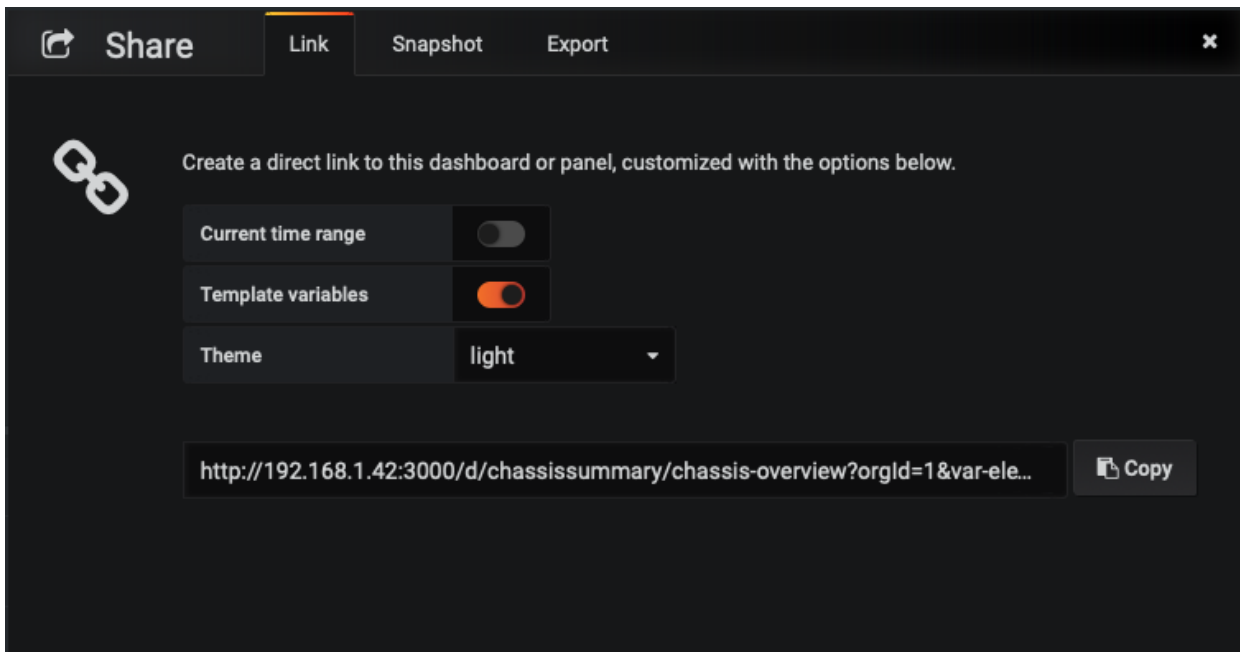
The dashboard must accept an element query parameter specifying the element for which the metrics shall be displayed.

To add a Grafana dashboard to RBMS

1. Open the dashboard in Grafana.
2. Click the **Share** icon.



- 3. Uncheck the **Current time** option.
- 4. Select the *light* theme.
- 5. Click on **Copy** to copy the displayed link.



- 6. Proceed to RBMS.
- 7. Click on the **Inventory** tab.
- 8. Click on **+** to expand the **Administration** menu.
- 9. Click on **Dashboard & Panels** in the left navigation pane. The list of registered dashboard and panels appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels**
- Templates

Visualizations

List of registered visualizations

No visualizations found.

Create a visualization platform or register a first visualization.

Add visualization platform
Add visualization

10. Click **Add visualization**. The New Visualization page appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels**
- Templates

Visualizations > New Visualization

New Visualization

Create a new visualization.

Visualization Settings

Visualization Type

Grafana

Select the visualization type

Visualization Name

Chassis Overview

The name of the visualization.

Category

Optional visualization category.

Panel

This visualization is a panel to be embedded in other views.

Path

```
/d/chassissummary/chassis-overview?orgId=1&var-element={{element_name}}&theme=light
```

The path template to compute the visualization URI. All general element settings can be processed by the template. For example, use to add the element name to the path.

11. From the **Visualization Platform** select box, select the Grafana instance that hosts the dashboard.
12. In the **Visualization Name** field, specify a unique descriptive name. The name is displayed in the dashboard list of an element.

13. In the **Description** field, add an optional dashboard description. The description is displayed in the element dashboard list.
14. Paste the link into the **URL** field. Assign `{{element_name}}` as value for the element variable declared in the dashboard. RBMS replaces `{{element_name}}` with the name of the currently displayed element.
15. Optionally restrict the dashboards to certain element roles or platforms.

Visualization Binding
Select the element roles and hardware platforms for which this visualization exists.

Element Roles

accessleaf
borderleaf
spine
integrationtest

Restrict visualization availability to the selected element roles. By default the visualizations is available for all element roles.

Platforms

EC5916-54XK
EC5916-54XKS
foo
Wedge-100BF-32X
Wedge-100BF-65X

Restrict visualization availability to the selected platforms. By default the visualizations is available for all platforms.

Cancel
Add visualization

16. Click **Add visualization** to add the dashboard.

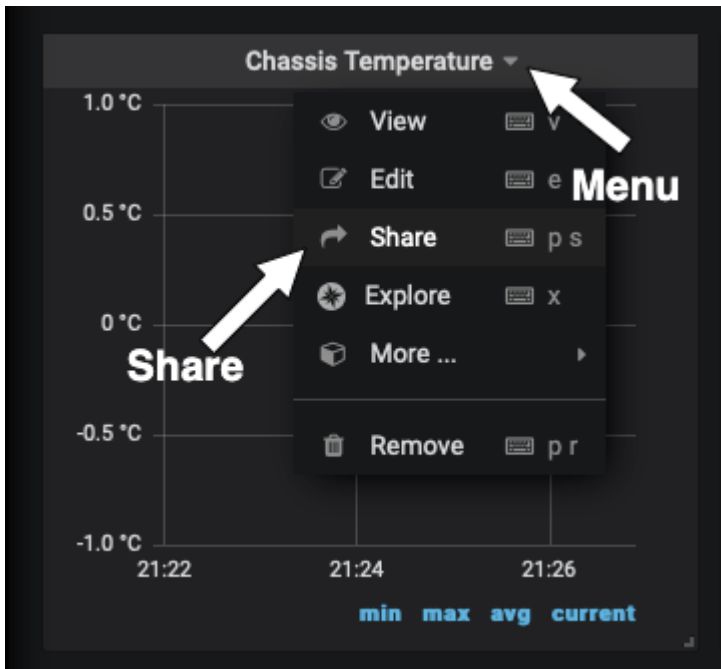
Registering a panel for a configured metric



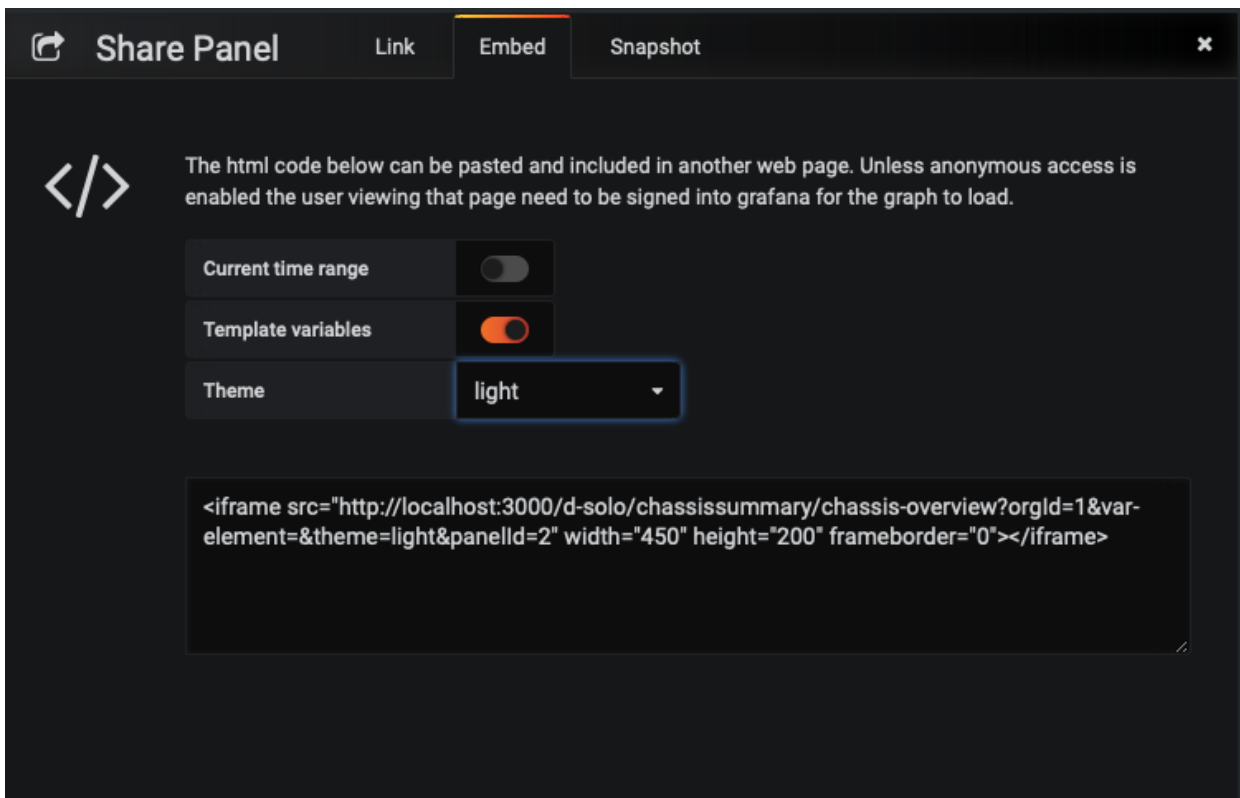
The panel must accept URL query parameters specifying the element for which the metric shall be displayed.

To add a Grafana panel to RBMS

1. Open the dashboard in Grafana.
2. Open the context menu for the panel you want to embed in RBMS.
3. Click the **Share** menu.



4. Uncheck the **Current time** option.
5. Select the *light* theme.
6. Click on **Copy** to copy the displayed link.



7. Proceed to RBMS.
8. Click on the **Inventory** tab.

9. Click on **+** to expand the **Administration** menu.
10. Click on **Dashboard & Panels** in the left navigation pane. The list of registered dashboard and panels appears.

Inventory
Manage network elements

Images Inventory Metrics Jobs Logs Administration [Logout](#)

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels**
- Templates

Visualizations
List of registered visualizations

No visualizations found.
[Create a visualization platform or register a first visualization.](#)

[Add visualization platform](#) [Add visualization](#)

11. Click **Add visualization**. The New Visualization page appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels**
- Templates

Visualizations > New Visualization

New Visualization

Create a new visualization.

Visualization Settings

Visualization Type
 Grafana

Select the visualization type

Visualization Name
 Chassis Temperature

The name of the visualization.

Category

Optional visualization category.

Panel
 This visualization is a panel to be embedded in other views.

Panel Name
 chassis_temperature_millicelsius

Width
 100%

Specify the width of the embedded visualization

Height
 400px

Specify the height of the embedded visualization

Path
 /d-solo/chassissummary/chassis-overview?orgId=1&var-element={{element_name}}&panelId=2

The path template to compute the visualization URI. All general element settings can be processed by the template. For example, use to add the element name to the path.

12. From the **Visualization Platform** select box, select the Grafana instance that hosts the dashboard.
13. In the **Visualization Name** field, specify a unique descriptive name.
14. Check the **Panel** checkbox.
15. In the **Panel Name** field, add the name of the metric displayed in the panel.
16. In the **Width** field, add the width of the panel as absolute value (px or em) or in percent.
17. In the **Height** field, add the height of the panel as absolute value (px or em) or in percent.

18. In the **Description** field, add an optional dashboard description. The description is displayed in the element dashboard list.
19. Paste the link into the **URL** field. Assign `{{element_name}}` as value for the element variable declared in the dashboard. RBMS replaces `{{element_name}}` with the name of the currently displayed element.
20. Optionally restrict the panel to certain element roles or platforms.

Visualization Binding
 Select the element roles and hardware platforms for which this visualization exists.

Element Roles

accessleaf
 borderleaf
 spine
 integrationtest

Restrict visualization availability to the selected element roles. By default the visualizations is available for all element roles.

Platforms

EC5916-54XK
 EC5916-54XKS
 foo
 Wedge-100BF-32X
 Wedge-100BF-65X

Restrict visualization availability to the selected platforms. By default the visualizations is available for all platforms.

Cancel
Add visualization

21. Click **Add visualization** to add the panel.

Support for other visualization platforms

RBMS supports all visualization platforms accepting the element for which data shall be displayed as URL query parameter.

6. Metric Sampling and Monitoring

6.1. Metric sampling Overview

Metric sampling is configured through the CTRLD/RESTCONFD API. The sampled data is stored in Prometheus, an open source monitoring tool with a built-in time series database (TSDB), and can be queried from the switch using PromQL, the Prometheus Query Language. The CTRLD API also supports programming alert conditions in Prometheus Alert Manager.

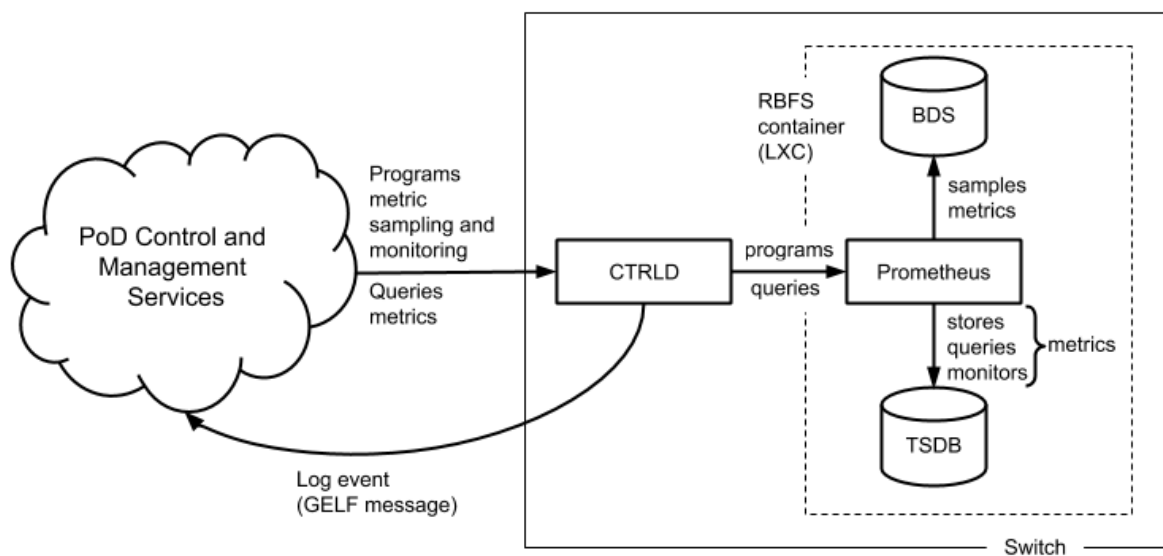


Figure 1. Metric sampling and monitoring overview.

Brick daemons feeding data into BDS are not depicted to keep the drawing simple.

BDS as Single Point of Truth

The Brick Data Store (BDS) is an object-oriented in-memory database that stores the switch configuration and operational state. BDS objects are typed objects, which means that every object and object attribute is of a certain type. BDS objects are described in schemas and organized in tables. One or more indexes per table exists to query objects. BDS supports sampling values from

- BDS object attributes and from
- BDS table indexes.

Every *numeric* BDS object attribute can be periodically sampled to create a time

series of the attribute value. In addition, BDS provides built-in converters for some attribute types that can be converted to numeric values. The bandwidth type is a good example. The bandwidth is stored as a string and consists of a numeric value and a data rate unit, for example, 100.000 Gbps. The built-in converter translates the bandwidth to a numeric value in bits per seconds.

BDS indexes are sampled if the number of objects in a table is of interest. This tutorial includes examples for object- and index-based metrics.

Metric Types

There exist two types of metrics:

- gauge
- counters

A gauge metric values are within a certain range and can basically be visualized as is or after applying a simple linear transformation. The value of a gauge metric can increase and decrease. A temperature value is an example for a metric of type gauge.

Counters increase until they are reset manually or by a restart (or by an overflow which is very unlikely to happen because of the length of the data word to store the counter value). The interesting aspect when working with counters is the delta of the count value between two samples, i.e. the derivation of the counter value over time. The derivation of the counter value is still an absolute value that needs to be put into perspective to the available resources to compute the resource utilization in percent. The CPU tick counters are examples for counter metrics.

Metric Labels

Metric labels separate metric instances from each other. Metric labels have either a static value or is read from a BDS object attribute.

The byte counters, for example, exist for each physical interface. The *ifp_name* label assigns the sampled counter values to the physical interface and is read from the *interface_name* attribute.

Sampling Rate and Retention Period

The sampling rate is 5 seconds and the retention period is five days. The configuration is built-in to the image and cannot be changed through the CTRLD API.

Metric Monitoring

Metric monitoring relies on the *Prometheus Alert Manager*. The alert manager notifies CTRLD about all satisfied alert conditions. CTRLD translates the notification and forwards the message to the configured log management system. CTRLD exposes an API for programming alert conditions and in turn programs the Prometheus Alert Manager based on the specified alert rules.

6.2. Temperature Monitoring

The first example in this tutorial samples and monitors temperature values to outline how to work with gauge metrics. Open the RBFS CLI and run `show sensor temperature`` to list all available temperature sensors.

Listing 1 - CLI output of temperature values.

```
supervisor@rtbrick>BNG: op> show sensor temperature
Name                Temperature      Status
CPU Core            49°C            PRESENT
LM75-1-48           34°C            PRESENT
LM75-2-49           33°C            PRESENT
LM75-3-4A           29°C            PRESENT
LM75-3-4B           31°C            PRESENT
PSU-1 Thermal Sensor 1 28°C            PRESENT
```

This switch has four chassis temperature sensors (LM75), a CPU temperature sensor (CPU Core) and a power supply unit (PSU) temperature sensor (PSU-1 Thermal Sensor 1). A switch typically has two independent power supply units. The second PSU of this switch was not attached in the lab environment.

The temperature is read from the temperature attribute of the *sensor_object* stored in the *global.chassis_0.resource.sensor* BDS table. The sensor object also includes a type (*resource_type* attribute) and a name (*resource_name* attribute). The unit of the temperature is millidegree celsius. An excerpt of the sensor schema definition is listed below:

Listing 2 - Excerpt from BDS sensor object schema definition.

```

    {
      "codepoint": 2,
      "name": "resource_name",
      "type": "string",
      "description": "Name of the resource"
    },
    ...
    {
      "codepoint": 4,
      "name": "resource_type",
      "type": "string",
      "description": "resource type"
    },
    ...
    {
      "codepoint": 33,
      "name": "temperature",
      "type": "uint32",
      "description": "temperature in millidegree celsius"
    }
  }

```



Contact RtBrick professional services if you need help in finding the BDS table and attribute names.

Sampling Temperature Sensors

Based on the available sensors it makes sense to create three temperature metrics:

- *chassis_temperature_millicelsius* to sample the chassis temperature
- *cpu_temperature_millicelsius* to sample the CPU temperature and
- *psu_temperature_millicelsius* to sample the PSU temperature.

The `CTRLD/RESTCONFD` API exposes the `/api/v1/rbfs/elements/{element}/services/restconfd/proxy/restconf/data/rtbrick-config:time-series/metric={metric_name}` endpoint. A HTTP PUT request to this endpoint configures a metric by either creating a new metric or replacing an existing metric with the specified `{metric_name}`. `{element}` contains the name of the element assigned in the element configuration file and defaults to the container name if no element name was specified. The default container name is `rtbrick`.

All metrics need to be labeled with the sensor name. In addition, a filter is needed to sample only the sensors for the respective type of temperature. The listings

below show the JSON objects to sample the chassis temperature as an example:

Listing 3 - JSON object to configure chassis temperature sampling.

```
{
  "rtbrick-config:metric": [
    {
      "name": "chassis_temperature_millicelsius",
      "table-name": "global.chassis_0.resource.sensor",
      "bds-type": "object-metric",
      "prometheus-type": "gauge",
      "description": "Chassis temperature in millidegree celsius",
      "filter": [
        {
          "match-attribute-name": "resource_name",
          "match-attribute-value": "LM.*",
          "match-type": "regular-expression"
        },
        {
          "match-attribute-name": "resource_type",
          "match-attribute-value": "thermal",
          "match-type": "exact"
        }
      ],
      "attribute": [
        {
          "attribute-name": "temperature",
          "label": [
            {
              "label-key": "sensor",
              "label-value": "resource_name",
              "label-type": "dynamic"
            }
          ]
        }
      ]
    }
  ]
}
```

The temperature metric is of type gauge (**metric_type**) and sample from a BDS object (**bds_metric_type**). The temperature value shall be sampled, which is of numeric type (uint32, see the schema definition above). The filter section makes sure that only thermal sensors are sampled and also filters for the LM sensors that measure the chassis temperature.

Querying the Chassis Temperature

The following PromQL query returns the chassis temperature in degree Celsius from the Prometheus running on the switch.

```
chassis_temperature_millicelsius / 1000
```

The next query converts the chassis_temperature from degree Celsius to Fahrenheit:

```
(chassis_stemperature_millicelsius / 1000) * 9 / 5 + 32
```

Both expressions are examples for simple linear transformations of a gauge metric. The queries can be used in Grafana to visualize the chassis temperature time series. The screenshot below shows a chassis temperature panel of a Grafana dashboard:

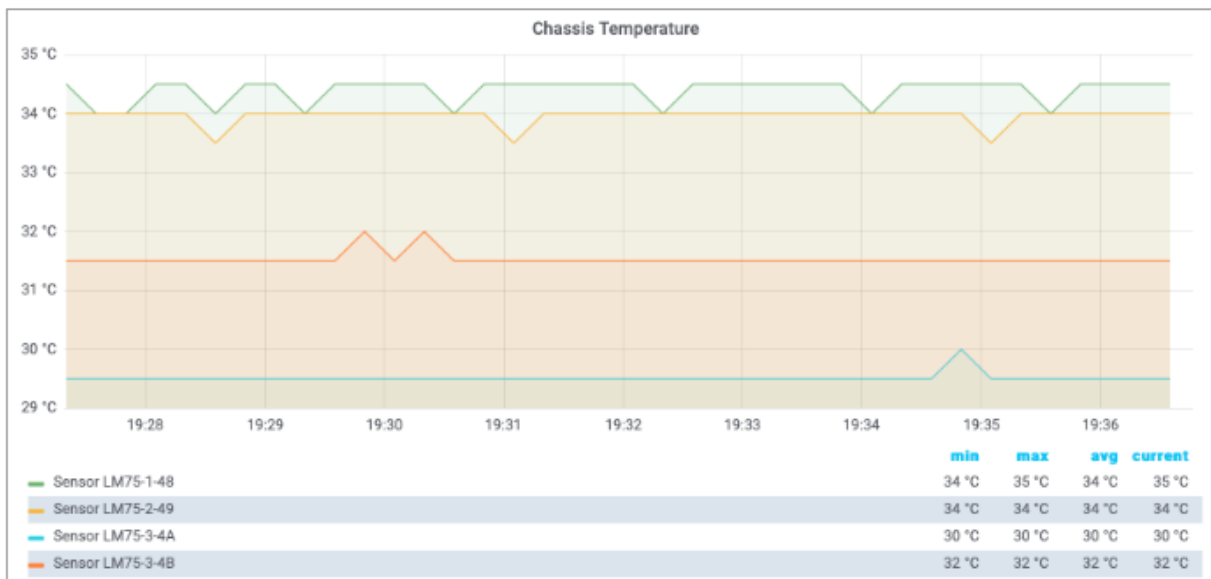


Figure 2. Chassis temperature Grafana panel.



The Grafana dashboard settings for the metrics used in this tutorial can be requested from RtBrick professional services.

Monitoring Temperature Values

A high temperature can damage the device or shorten its lifetime. Therefore it makes sense to monitor the temperature to get notified about critical temperature values. The alert condition is defined by the acceptable duration of exceeding a specified temperature value, for example, the average chassis temperature is not allowed to exceed 40°C over the last five minutes.



The temperature threshold and evaluation period are example values. The actual values must be taken from the hardware platform documentation or requested from the vendor.

The listing below shows the complete chassis temperature alert rule.

Listing 4 - Chassis temperature alert rule.

```

{
  "rtbrick-config:alert": [
    {
      "name": "ChassisTemperatureAlert",
      "group": "health",
      "interval": "1m",
      "expr": "avg_over_time(chassis_temperature_millidegrees[1m]) / 1000 >
40",
      "for": "5m",
      "level": "1",
      "summary": "The chassis temperature exceeded 40°C.",
      "description": "The {{$labels.element_name}} average chassis
temperature over the past 5 minutes exceeded 40°C."
    }
  ]
}

```

The alert rule evaluates every single minute (*interval*) whether the average temperature in the past minute exceeded 40 degrees (*expr*) and raises an alert if the expression is satisfied for 5 minutes (*for*), that is, 5 times in a row. The *summary* field contains a short description of the problem whereas the optional *description* field contains a more detailed message. The summary is mapped to the *short_message* GELF field and the description is mapped to the *full_message* GELF field. The severity is set to Alert (*level*). The *level* attribute values are taken from the GELF format which in turn took it from the Syslog protocol. The table below lists all supported levels:

GELF message severity levels

Level	Description as in RFC 5424	
	Name	Comment
0	Emergency	System is unusable
1	Alert	Action must be taken immediately
2	Critical	Critical conditions
3	Error	Error conditions
4	Warning	Warning conditions
5	Notice	Normal but significant condition
6	Informational	Informational messages
7	Debug	Debug-level messages

Every alert rule has a unique name (*alert_rule_name*). The PUT operation replaces

an existing alert rule with the same name. Every alert rule is assigned to exactly one alert group (`alert_group`). All alert rules in the same alert group with the same interval setting are evaluated at the same time.

The `for` attribute is optional. A similar alert rule can be implemented by omitting the `for` attribute and computing the average temperature over the past five minutes:

Listing 5 - Alternative chassis temperature alert rule.

```
{
  "rtbrick-config:alert": [
    {
      "name": "ChassisTemperatureAlert",
      "group": "health",
      "interval": "1m",
      "expr": "avg_over_time(chassis_temperature_millidegrees[5m]) / 1000 > 40",
      "level": "1",
      "summary": "The chassis temperature exceeded 40°C.",
      "description": "The {{$labels.element_name}} average chassis temperature over the past 5 minutes exceeded 40°C."
    }
  ]
}
```

There is a subtle difference between both rules. Consider the following temperature values:

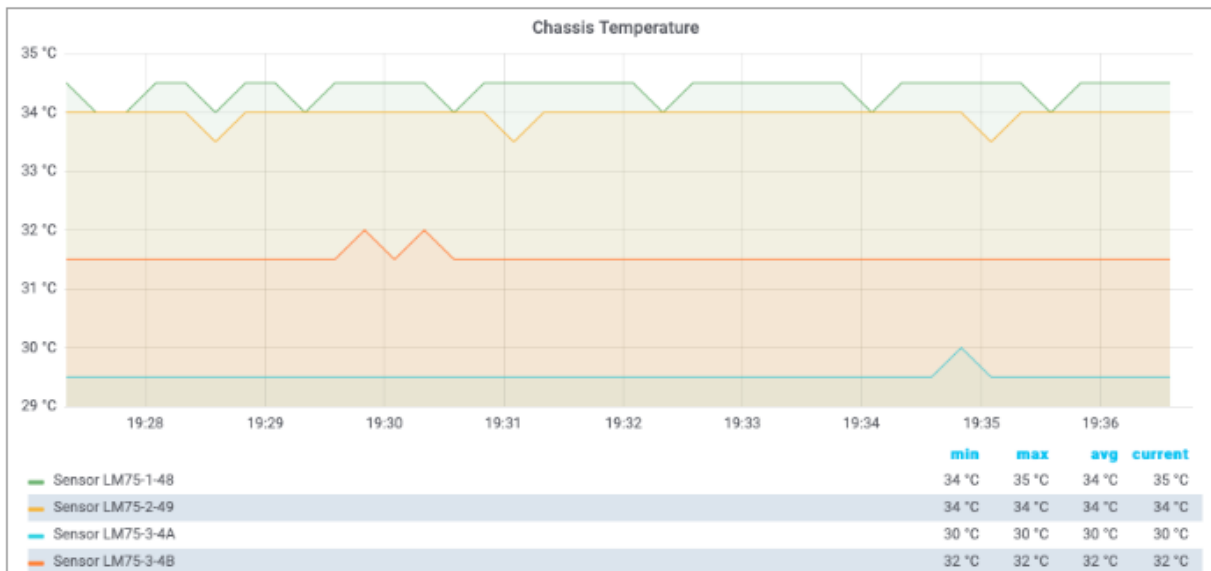


Figure 3. Chassis temperature values.

The first rule does not fire because the threshold is only exceeded for three times, whereas the second rule fires because the average over the past five minutes

exceeds 40°C. In fact, the second rule fires an alert albeit the temperature exceeds the alert threshold for 4 minutes only. How about using the min rather than the avg function, i.e. the temperature must exceed the threshold for five minutes? In this case, the rule wouldn't fire an alert if the chassis temperature is wobbling around the threshold.

The first rule aims to mitigate both effects:

- The first rule fires an alert if the chassis temperature is wobbling around the threshold but on average exceeds the threshold five times in a row.
- The first rule does not fire an alert in case of a chassis temperature spike as depicted in Figure 3, because a spike does not satisfy the rule five times in a row.

6.3. CPU Utilization

The second example measures the CPU utilization to outline how to work with counter metrics. Open the RBFS CLI and run `show cpu usage` to display the current CPU utilization.

Listing 6 - CPU core utilization CLI command.

```
supervisor@rtbrick>BNG: op> show cpu usage
Name          Total    User    System    Nice    I/O Wait    Idle    IRQ    Soft IRQ
cpu           4%      2%      2%       0%      0%          95%    0%     0%
cpu0          1%      0%      1%       0%      0%          99%    0%     0%
cpu1         16%     12%     4%       0%      0%          83%    0%     0%
cpu2          3%      2%      0%       0%      0%          96%    0%     0%
cpu3         10%     1%      9%       0%      0%          89%    0%     0%
cpu4          2%      1%      1%       0%      0%          97%    0%     0%
cpu5          4%      0%      4%       0%      0%          96%    0%     0%
cpu6          3%      3%      0%       0%      0%          97%    0%     0%
cpu7          0%      0%      0%       0%      0%          100%   0%     0%
```

The CPU provides a set of counters to measure the CPU utilization in jiffies /3/. A jiffy is the duration of a software clock tick, which is platform-dependent. By that, a jiffy is neither a constant period of time nor very meaningful to a human, which is why the counter values need to be put into perspective.

First, it is important to measure the total CPU utilization to see how busy the switch is. Secondly, if the CPU utilization is considerably high, it is interesting to find out which processes cause the high CPU utilization. Both aspects are addressed in this tutorial.

The time spend in user and kernel space needs to be divided by the total amount

of available processing time to compute the total CPU utilization:

$$\text{total_cpu_utilization} = \frac{(\text{total_cpu_user_jiffy} + \text{total_cpu_sys_jiffy})}{(\text{total_cpu_total_jiffy})}$$

where

- **total_cpu_user_jiffy** is the total amount of time spent in user mode in a sampling interval,
- **total_cpu_sys_jiffy** is the total amount of time spent in kernel mode in a sampling interval and
- **total_cpu_total_jiffy** is the total amount of computing time available in a sampling interval.

The **cpu_total_utilization** value is dimensionless. The value range is between 0 and 1. It can be converted into percentage by being multiplied by 100%.

$$\text{total_cpu_utilization_percentage} = \text{total_cpu_utilization} * 100\%$$

The process total load ratio expresses the ratio a process load to the total load:

$$\text{proc_total_load_ratio} = \frac{(\text{proc_cpu_user_jiffy} + \text{proc_cpu_sys_jiffy})}{(\text{total_cpu_user_jiffy} + \text{total_cpu_sys_jiffy})}$$

where

- **proc_cpu_user_jiffy** is the process time spent in user mode in a sampling interval and
- **cpu_sys_proc_jiffy** is the process time spent in user mode in a sampling interval

The **process_total_load_ratio** value is dimensionless. The value range is between 0 and 1. It can be converted into percentage by being multiplied by 100%.

$$\text{proc_total_load_ratio_percentage} = \text{proc_total_load_ratio} * 100\%$$

Sampling CPU Counters

The CPU counters are located in two different tables. The total CPU utilization can be sampled from the **user_cpu_tick**, **sys_cpu_tick** and **total_cpu_tick** attributes in the **global.chassis_0.resource.cpu_usage** table. This table contains the total

counters but also counters per supported hardware thread (virtual core).

The JSON objects below enables CPU counter sampling for the three mentioned counters:

Listing 7 - JSON object to enable total CPU utilization counter sampling.

```
{
  "rtbrick-config:metric": {
    "name": "total_cpu_total_jiffy",
    "table-name": "global.chassis_0.resource.cpu_usage",
    "bds-type": "object-metric",
    "prometheus-type": "counter",
    "description": "Total CPU utilization in jiffies",
    "attribute": [
      {
        "attribute-name": "total_cpu_tick",
        "label": [
          {
            "label-key": "cpu",
            "label-value": "cpu_id",
            "label-type": "dynamic"
          }
        ]
      }
    ]
  }
}
```

Listing 8 - JSON object to enable total user mode CPU utilization counter sampling.

```
{
  "rtbrick-config:metric": {
    "name": "total_cpu_user_jiffy",
    "table-name": "global.chassis_0.resource.cpu_usage",
    "bds-type": "object-metric",
    "prometheus-type": "counter",
    "description": "Total user CPU utilization.",
    "attribute": [
      {
        "attribute-name": "user_cpu_tick",
        "label": [
          {
            "label-key": "cpu",
            "label-value": "cpu_id",
            "label-type": "dynamic"
          }
        ]
      }
    ]
  }
}
```

Listing 9 - JSON object to enable total kernel CPU utilization counter sampling.

```
{
```

```

"rtbrick-config:metric": {
  "name": "total_cpu_sys_jiffy",
  "table-name": "global.chassis_0.resource.cpu_usage",
  "bds-type": "object-metric",
  "prometheus-type": "counter",
  "description": "Total kernel CPU utilization in jiffies",
  "attribute": [
    {
      "attribute-name": "sys_cpu_tick",
      "label": [
        {
          "label-key": "cpu",
          "label-value": "cpu_id",
          "label-type": "dynamic"
        }
      ]
    }
  ]
}

```

Computing Total CPU Utilization From Counter Samples

The Prometheus Query Language /2/ provides functions to work with counters and also allows to put time series into perspective.



Some PromQL functions should be used for gauge metrics only others only for counter metrics.

The PromQL queries below computes the total user, kernel and user + kernel CPU utilization:

```

rate(total_cpu_user_jiffy{cpu="cpu"}[60s])
/ rate(total_cpu_total_jiffy{cpu="cpu"}[60s])

rate(total_cpu_sys_jiffy{cpu="cpu"}[60s])
/ rate(total_cpu_total_jiffy{cpu="cpu"}[60s])

( rate(total_cpu_user_jiffy{cpu="cpu"}[60s])
+ rate(total_cpu_sys_jiffy{cpu="cpu"}[60s]) )
/ rate(total_cpu_total_jiffy{cpu="cpu"}[60s])

```

The rate function computes the delta between two sampled count values. The rate function is optimized for counters and can detect counter resets by being aware that a counter value always increases unless a reset has taken place. The rate function handles counter resets properly. The cpu label filters for the total count values for all virtual cores.

The PromQL query below computes the virtual core utilization:

```
( rate(total_cpu_user_jiffy{cpu!="cpu"}[60s])
+ rate(total_cpu_sys_jiffy{cpu!="cpu"}[60s]))
/ rate(total_cpu_total_jiffy{cpu!="cpu"}[60s])
```

The `cpu` label identifies the virtual core. The BDS contains count values for each virtual core but also the total count over all virtual cores. The first dashboard queried the total count by filtering for `cpu="cpu"`, whereas the second dashboards fetched the per virtual core counters by filtering for `cpu!="cpu"`, i.e. by excluding the total count over all virtual cores from the result set.

The screenshots below show Grafana dashboard panels to display the computed total CPU utilizations and the utilization of the virtual cores.

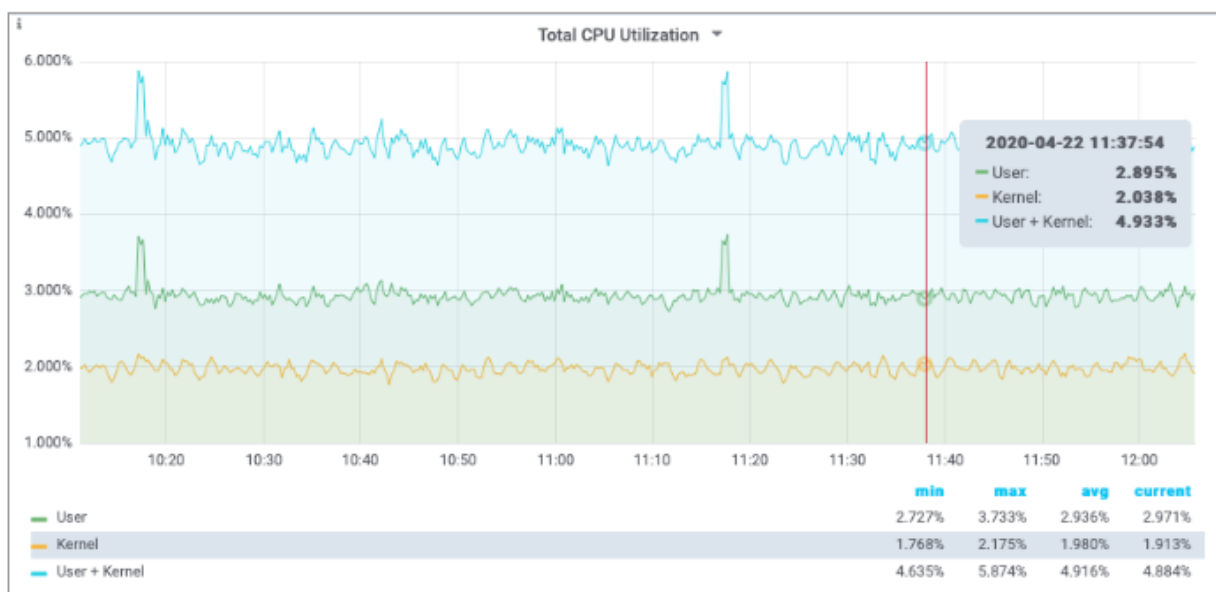


Figure 4. Total CPU utilization Grafana panel.

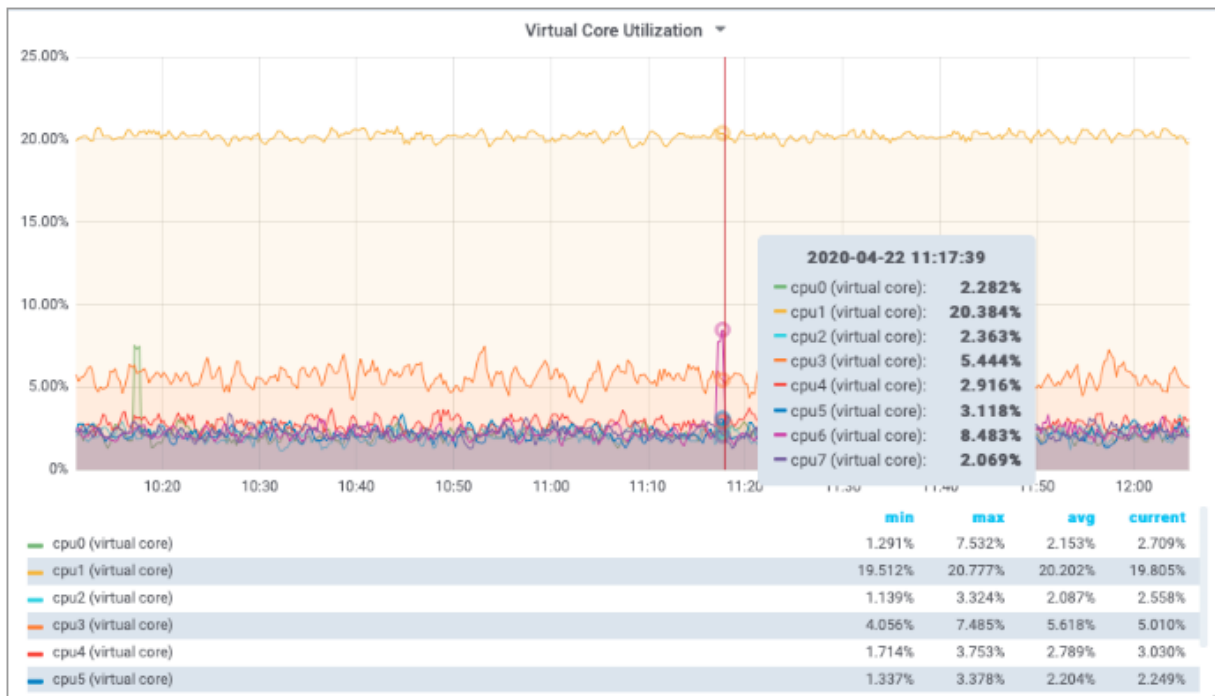


Figure 5. Virtual core utilization Grafana panel.

Sampling Process CPU Counters

The next step is to compute the per process CPU utilization. This requires to sample the process utilization counters of each process and put them into perspective of the total CPU counters.

The process CPU usage can be read from the `cpu_user` and `cpu_sys` attributes in the `global.chassis_0.resource.proc_usage` table. The process name can be read from the `process_name` attribute. The listings below configure user mode and kernel mode CPU utilization sampling per process:

Listing 10 - JSON object to enable process kernel mode CPU utilization sampling.

```
{
  "rtbrick-config:metric": {
    "name": "proc_cpu_sys_jiffy",
    "table-name": "global.chassis_0.resource.proc_usage",
    "bds-type": "object-metric",
    "prometheus-type": "counter",
    "description": "Process kernel mode CPU utilization in jiffies",
    "attribute": [
      {
        "attribute-name": "cpu_sys",
        "label": [
          {
            "label-key": "process",
            "label-value": "process_name",
            "label-type": "dynamic"
          }
        ]
      }
    ]
  }
}
```

```

    ]
  }
]
}
}

```

Listing 11 - JSON object to enable process user mode CPU utilization sampling.

```

{
  "rtbrick-config:metric": {
    "name": "proc_cpu_user_jiffy",
    "table-name": "global.chassis_0.resource.proc_usage",
    "bds-type": "object-metric",
    "prometheus-type": "counter",
    "description": "Process user mode CPU utilization in jiffies",
    "attribute": [
      {
        "attribute-name": "cpu_user",
        "label": [
          {
            "label-key": "process",
            "label-value": "process_name",
            "label-type": "dynamic"
          }
        ]
      }
    ]
  }
}

```

Computing Process CPU Utilization From Counter Samples

The PromQL query puts the CPU counters of each process into perspective of the total CPU utilization.

```

( rate(proc_cpu_sys_jiffy[60s]) + rate(proc_cpu_user_jiffy[60s]) )
/ scalar(rate(total_cpu_total_jiffy{cpu="cpu"}[60s]))

```

The scalar function converts the one-dimensional `total_cpu_total` vector to a scalar to put the CPU process utilization into perspective.



Prometheus differentiates between vectors and scalars. Algebraic operations between two vectors, like the addition of the `proc_cpu_sys` and the `proc_cpu_user` vectors above, require that both vectors have the same labels. Otherwise no data points are returned by Prometheus, because a built-in filter excludes all items with different labels from the computation.

The screenshot below shows a Grafana panel to display the total CPU utilization of

each brick daemon.

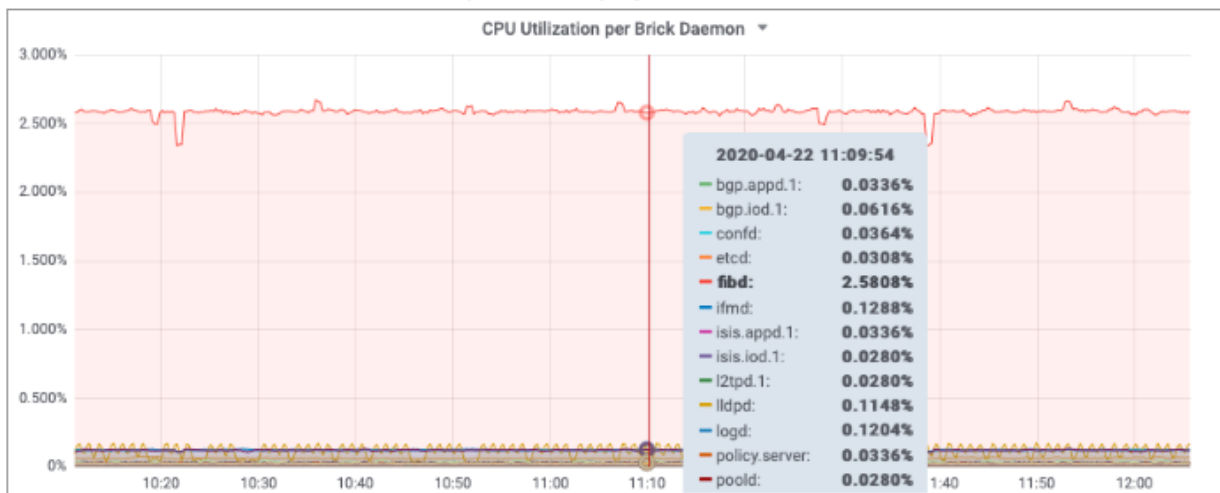


Figure 6. Brick daemon CPU utilization.

6.4. PPPoE Session Count

Sampling the number of PPPoE sessions is an example for a metric from a dedicated index table. The subscriber daemon maintains statistics of subscriber sessions grouped by access interface, access type and session lifecycle state. The `show subscriber count` command lists the session summary statistics

Listing 12 - Subscriber session count statistics

```

supervisor@BNG>rbms-tst00.vm.nbg.rtbrick.net: cfg> show subscriber count

```

	Total	Setup	Established	Terminating	Standby
Summary	1000	22	978	0	0
PPPoE	1000	22	978	0	0
L2TP	0	0	0	0	0
IPoE	0	0	0	0	0
L2BSA	0	0	0	0	0
Test	0	0	0	0	0
hostif-0/0/1	1000	22	978	0	0
PPPoE	1000	22	978	0	0
L2TP	0	0	0	0	0
IPoE	0	0	0	0	0
L2BSA	0	0	0	0	0
Test	0	0	0	0	0

The Listing below configures a metric to sample the PPPoE counters from the subscriber session statistic.

Listing 13 - JSON object to enable PPPoE session count sampling

```

{
  "rtbrick-config:metric": {
    "name": "pppoe_session_count",

```



```

"table-name": "local.access.subscriber.count",
"bds-type": "object-metric",
"prometheus-type": "gauge",
"description": "PPPoE sessions",
"attribute": [
  {
    "attribute-name": "pppoe_setup",
    "label": [
      {
        "label-key": "ifp_name",
        "label-value": "ifp_name",
        "label-type": "dynamic"
      },
      {
        "label-key": "access_type",
        "label-value": "pppoe",
        "label-type": "static"
      },
      {
        "label-key": "state",
        "label-value": "setup",
        "label-type": "static"
      }
    ]
  },
  {
    "attribute-name": "pppoe_established",
    "label": [
      {
        "label-key": "ifp_name",
        "label-value": "ifp_name",
        "label-type": "dynamic"
      },
      {
        "label-key": "access_type",
        "label-value": "pppoe",
        "label-type": "static"
      },
      {
        "label-key": "state",
        "label-value": "established",
        "label-type": "static"
      }
    ]
  },
  {
    "attribute-name": "pppoe_terminating",
    "label": [
      {
        "label-key": "ifp_name",
        "label-value": "ifp_name",
        "label-type": "dynamic"
      },
      {
        "label-key": "access_type",
        "label-value": "pppoe",
        "label-type": "static"
      },
      {
        "label-key": "state",
        "label-value": "terminating",

```

```

        "label-type": "static"
      }
    ]
  }
}

```



The PPPoE session count can increase and decrease. Therefore the metric type must be gauge rather than counter.

Each count is labelled with the access type, the session state and the access interface name. This allows aggregation over all time-series to compute the total counts per interface and also for the entire switch in Prometheus.

Run `show datastore subscriberd table local.access.subscriber.count` to display the statistics raw data. The `local.access.subscriber.count` table contains one object for each interface. Each object contains an attribute for each combination of access type and lifecycle state.

Listing 14 - Subscriber session raw data.

```

show datastore subscriberd.1 table local.access.subscriber.count
Object: 0, Sequence 1, Last update: Fri Jan 12 09:48:35 GMT +0000 2024
Attribute                                     Type                                     Length
Value
ifp_name (1)                                 string (9)                               13
hostif-0/0/1
interval (2)                                 interval (26)                             4
3000
pppoe_setup (3)                              uint32 (4)                                4
22
pppoe_established (4)                       uint32 (4)                                4
978
pppoe_terminating (5)                      uint32 (4)                                4
0
pppoe_standby (6)                          uint32 (4)                                4
0
l2tp_setup (7)                              uint32 (4)                                4
0
l2tp_established (8)                       uint32 (4)                                4
0
l2tp_terminating (9)                      uint32 (4)                                4
0
l2tp_standby (10)                         uint32 (4)                                4
0
ipoe_setup (11)                            uint32 (4)                                4
0
ipoe_established (12)                     uint32 (4)                                4
0
ipoe_terminating (13)                    uint32 (4)                                4
0
ipoe_standby (14)                         uint32 (4)                                4
0

```

```

l2bsa_setup (15)                uint32 (4)                4
0
l2bsa_established (16)          uint32 (4)                4
0
l2bsa_terminating (17)         uint32 (4)                4
0
l2bsa_standby (18)             uint32 (4)                4
0
test_setup (19)                uint32 (4)                4
0
test_established (20)          uint32 (4)                4
0
test_terminating (21)          uint32 (4)                4
0
test_standby (22)              uint32 (4)                4
0

```

6.5. Count of Received IPv4 Unicast Prefixes

Sampling the number of received unicast prefixes is an example for an index based metric. The IPv4 unicast prefixes are stored in the `instance.default.ribd.1.fib-local.ipv4.labeled-unicast` table, which exists per routing instance with `instance` being the instance name. The prefix count can be read by sampling the `active-entry-count` attribute of the primary index. The listing below shows the JSON object to sample the active PPPoE sessions:

Listing 15 - JSON object to enable PPPoE session count sampling

```

{
  "rtbrick-config:metric": {
    "name": "default_ipv4_unicast_prefix_count",
    "table-name": "default.ribd.1.fib-local.ipv4.unicast",
    "bds-type": "index-metric",
    "index-name": "primary",
    "prometheus-type": "gauge",
    "description": "IPv4 unicast prefixes default instance",
    "attribute": [
      {
        "attribute-name": "active-entry-count"
      }
    ]
  }
}

```



The prefix count can increase and decrease. Therefore the metric type must be gauge rather than counter.

Run `show datastore ribd schema table table-name default.ribd.1.fib-local.ipv4.unicast` to inspect the definition of the `default.ribd.1.fib-local.ipv4.unicast` table.

Listing 16 - Excerpt of the table definition CLI output

```
$ show datastore fibd schema table-name default.ribd.1.fib-local.ipv4.labeled-unicast
{
  "table": {
    "type": "rib_fiblocal_v4_table",
    "object": "rib_entry",
    "table_objects": {
      "shared_object": "generic_table_attributes",
      "app_objects": [
        "generic_table_attributes"
      ]
    },
    "index": [
      {
        "name": "primary",
        "type": "radix",
        "immutable": true,
        "key": [
          "prefix4"
        ]
      },
      ...
    ]
  }
}
```

The primary index uses the IPv4 prefix (prefix4) as key and contains an object per prefix. Counting the objects therefore represents the stored prefixes.

6.6. Metric Management

A HTTP GET request to the `/api/v1/rbfs/elements/{{element}}/services/restconfd/proxy/restconf/data/rtrbrick-config:time-series/metric=/name` CTRLD/RESTCONF API endpoint lists all metrics configured on the switch. A HTTP GET request to `/api/v1/rbfs/elements/{{element}}/services/restconfd/proxy/restconf/data/rtrbrick-config:time-series/metric={metric_name}` returns the complete metric settings. `{element}` is the assigned element name and `{metric_name}` contains the name of the requested metric.

Metric sampling is stopped by sending a HTTP DELETE request to the `/api/v1/rbfs/elements/{{element}}/services/restconfd/proxy/restconf/data/rtrbrick-config:time-series/metric={metric_name}` RESTCONF API endpoint to remove the metric settings from the switch configuration. More information can be found in the CTRLD/RESTCONF API /1/.

6.7. Grafana Dashboards

Grafana can visualize time series data from Prometheus /5/. Grafana can query the Prometheus instance on RBFS by using CTRLD as proxy:

```
http://<SWITCH_MGMT_IP>:19091/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS/proxy/
```

The downside of this approach is that a Prometheus datasource needs to be created in Grafana for every switch. In addition, all dashboards must be created per switch too, because a dashboard panel can operate on a single datasource only. Fortunately, Prometheus can federate data from other Prometheus instances /6/. By that, all sampled metrics get accessible through a single Prometheus instance. In combination with Grafana dashboard variables, a dashboard can be configured to access all existing switches.

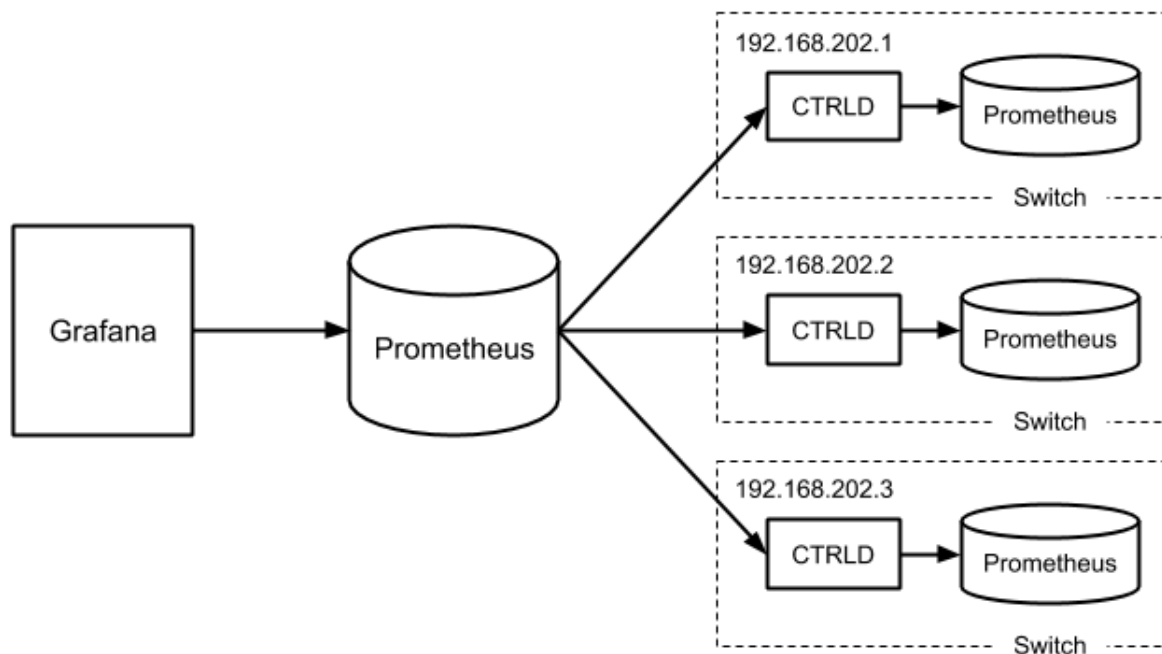


Figure 7. Prometheus federation.

The federating Prometheus instance can assign new label names. This allows to assign a unique `element_name` label value, if the element name is not specified on the switches and defaults to `rtbrick`. The listing below shows an excerpt of the Prometheus configuration to federate data from other Prometheus instances.

Listing 17 - Excerpt of a Prometheus federation configuration

```
scrape_configs:
```

```

- job_name: federate
  params:
    match[']:
      - '{job="bds"}'
  static_configs:
    - targets: ["192.168.202.1:19091"]
      labels:
        element_name: "l1.pod2"
        __metrics_path__:
"/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS/proxy/federate"
    - targets: ["192.168.202.2:19091"]
      labels:
        element_name: "s1.pod2"
        __metrics_path__:
"/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS/proxy/federate"
    - targets: ["192.168.202.3:19091"]
      labels:
        element_name: "b11.pod2"
        __metrics_path__:
"/api/v1/rbfs/elements/rtbrick/services/PROMETHEUS/proxy/federate"

```

The remaining step is to create a single datasource in Grafana to query the federated time series data.

6.8. Summary

This tutorial outlines how to configure metric sampling and monitoring in RBFS. Providing a full introduction to Grafana, Prometheus and the Prometheus Query Language would go beyond the scope of this tutorial. However, we mentioned some pitfalls and key aspects for working with PromQL and Grafana and recommend looking up more information in the Grafana and Prometheus documentations.

A postman collection to work with RBFS metrics and Grafana dashboards, including the dashboards this tutorial refers to, can be requested from RtBrick.

6.9. References

/1/	/resources/techdocs/current/ctrlld/01_switch_mgmt_api.html[Switch Management API Overview]
/2/	Querying Prometheus https://prometheus.io/docs/prometheus/latest/querying/basics/
/3/	Overview of time and timers http://man7.org/linux/man-pages/man7/time.7.html

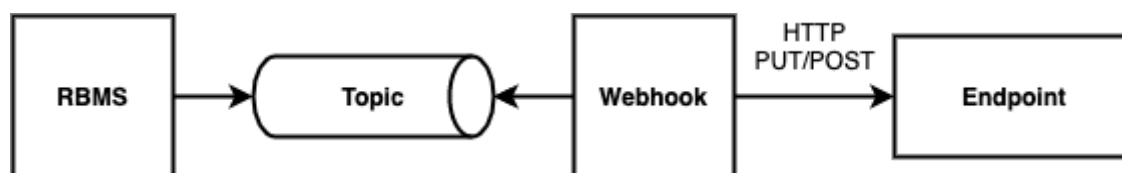
/4/	GELF - Graylog Extended Logging Format https://docs.graylog.org/en/3.2/pages/gelf.html
/5/	Grafana Documentation https://grafana.com/docs/grafana/latest/
/6/	Prometheus Federation https://prometheus.io/docs/prometheus/latest/federation/

7. Administering RBMS

7.1. Administration

Managing Webhooks

A webhooks is a registered HTTP endpoint that forwards notifications from RBMS to an external endpoint.



RBMS stores a *domain event* if a status in RBMS has changed. An event is only created when the transaction was committed. An event is not fired when the transaction rolls back.

The events are grouped in different topics:

- **element**, the element topic contains all element-related messages
- **image**, the image topic contains all image-related messages

An event has a descriptive name that describes what state change is being reported. All events have a unique ID to identify different instances of the same event unambiguously.

For example, the *ElementRenamedEvent* informs about an element being renamed. The event is stored in the element topic.

A webhook subscribes a topic and calls the configured endpoint for all events that match the specified name filter. By default, the message send to the endpoint contains the JSON representation of the domain event. An optional template allows rewriting the event message.

The authentication can be done via HTTP Basic Authorization or bearer token. RBMS stores the provided credentials AES-protected in the database.



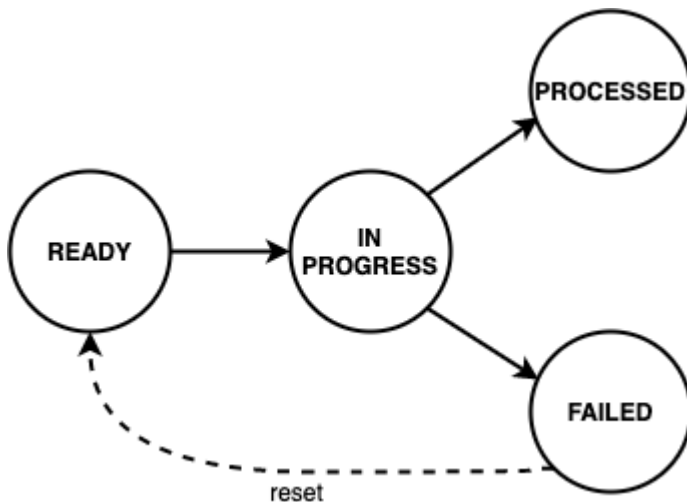
The AES secret and initialization vector (IV) can be specified in the `master.secret` and `master.iv` environment variables.

Unauthenticated endpoint calls are also supported.

A webhook invocation is considered *successful* if a HTTP success family status code is returned. For all other status codes the invocation is considered as failed.

A webhook can retry all failed messages. In addition, a webhook can be reset to a certain message to process this message and all subsequent messages again.

The complete message processing lifecycle is shown below:



New domain event messages are *READY* for being processed. The state changes to *IN PROGRESS* when the processing has begun and eventually to *PROCESSED* if the endpoint processed the message successfully and to *FAILED* otherwise respectively.

A webhook can be disabled to temporarily suspend the event processing. All events that occurred while the webhook was disabled are processed when the webhook gets enabled again unless the event got dropped because the topic buffering capacity was exceeded.

Viewing Webhooks

To view the list of webhooks

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Click the name of the webhook that you want to view.

Adding Webhooks

To add a webhook

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Click **Add webhook**
4. Specify the general, subscription and authentication information about the webhook.
5. Click **Save webhook**.

Disabling a Webhook

To disable a webhook

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Select the webhook to be disabled.
4. Click **Disable webhook**.

Enabling a Webhook

To enable a webhook

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Select the webhook to be enabled.
4. Click **Enable webhook**.

Reset a Webhook

To reset a webhook

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Select the webhook to be reset.

4. Click **Message Queue** in the left navigation pane. The list of the last 100 processed messages appear.
5. Enter the event ID in the **Filter** field and click **Filter**.
6. Open the displayed message.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

[Webhooks](#)

[Access Keys](#)

[Access Key Validator](#)

[Users](#)

[Roles](#)

Element_Updates -

webhook undefined

[Settings](#)

[Template](#)

Message Queue

[Statistics](#)

Webhooks > Element_Updates > Message Queue > 206b1481-8f85-40e8-9955-59140df96744

Message History

Review the message history of all processed messages and optionally reset the webhook to a previous message.

Event ID	206b1481-8f85-40e8-9955-59140df96744
Topic	element
Event Name	ElementRemovedEvent
Event Payload	<pre>{ "group_id": "434e53da-ba28-4c32-9556-d47a490d4d8b", "group_name": "blr", "group_type": "pod", "element_id": "d441e2ee-1038-45e2-ad7f-4254ffb2c826", "element_name": "leaf-a", "element_role": "accessleaf", "administrative_state": "NEW" }</pre>
Rewritten Message	<pre>{ "event_id": "206b1481-8f85-40e8-9955-59140df96744", "event_name": "ElementRemovedEvent", "message": { "group_id": "434e53da-ba28-4c32-9556-d47a490d4d8b", "group_name": "blr", "group_type": "pod", "element_id": "d441e2ee-1038-45e2-ad7f-4254ffb2c826", "element_name": "leaf-a", "element_role": "accessleaf", "administrative_state": "NEW" }, "topic_name": "element", "date_created": "2020-07-08T21:24:11.025+02:00" }</pre>
Date Created	08-JUL-2020 21:24:11.025

[Reset message](#)
[Reset webhook](#)

7. Click **Reset webhook** to process the message and all subsequent messages again. The message queue view appears.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

- Webhooks
- Access Keys
- Access Key Validator
- Users
- Roles

- Element_Updates webhook -
- Settings
- Template
- Message Queue**
- Statistics

Webhooks > Element_Updates > Message History

Message Queue

Review the Element_Updates webhook message queue

Filter Filter

Filter messages by correlation ID

Messages					
Date Modified	Event Name	Correlation ID	State	Execution Time	
13-JUL-2020 01:29:05.504	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:28:57.169	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:28:48.152	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:28:37.973	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:28:30.475	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:28:21.149	ElementSettingsUpdatedEvent	-	READY	-	
13-JUL-2020 01:18:28.139	ElementConfigStoredEvent	-	READY	-	
13-JUL-2020 01:18:11.853	ElementConfigStoredEvent	-	READY	-	
08-JUL-2020 21:24:11.025	ElementRemovedEvent	-	READY	-	
08-JUL-2020 21:24:08.440	ElementSettingsUpdatedEvent	-	PROCESSED	6 ms	
08-JUL-2020 21:24:04.911	ElementSettingsUpdatedEvent	-	PROCESSED	6 ms	
08-JUL-2020 21:24:00.623	ElementSettingsUpdatedEvent	-	PROCESSED	12 ms	
08-JUL-2020 21:18:00.157	ElementSettingsUpdatedEvent	-	PROCESSED	29 ms	
08-JUL-2020 21:17:56.021	ElementAddedEvent	-	PROCESSED	9 ms	
08-JUL-2020 21:11:27.296	ElementRemovedEvent	-	PROCESSED	16 ms	

Viewing Webhook Statistics

The webhook statistics provides information about processing times and the message count grouped by the processing state.

To view the webhook statistics

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Select the webhook for which to retry the failed invocations.
4. Click **Statistics** in the left navigation pane. The webhook statistics appear.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

Webhooks

Access Keys

Access Key Validator

Users

Roles

Element_Updates
webhook

Settings

Template

Message Queue

Statistics

Webhooks > Element_Updates > Webhook Statistics

Webhook Statistics

View the message processing statistics and try to process failed messages again.

Message State	Message Count	Min Exec Time	Avg Exec Time	Max Exec Time	Exec Time Stddev
PROCESSED	174	3 ms	16.413794 ms	131 ms	22.141975 ms
FAILED	0	-	-	-	-
IN_PROGRESS	0	-	-	-	-
READY	0	-	-	-	-

[Reset failed messages](#)

Retrying Failed Webhook Invocations

To retry failed webhook invocations

1. Click the **Administration** tab.
2. Click **Webhooks** in the left navigation pane. The list of all webhooks appear.
3. Select the webhook for which to retry the failed invocations.
4. Click **Statistics** in the left navigation pane. The message statistics appear.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

Webhooks

Access Keys

Access Key Validator

Users

Roles

Element_Updates
webhook

Settings

Template

Message Queue

Statistics

Webhooks > Element_Updates > Webhook Statistics

Webhook Statistics

View the message processing statistics and try to process failed messages again.

Message State	Message Count	Min Exec Time	Avg Exec Time	Max Exec Time	Exec Time Stddev
PROCESSED	174	3 ms	16.413794 ms	131 ms	22.141975 ms
FAILED	0	-	-	-	-
IN_PROGRESS	0	-	-	-	-
READY	0	-	-	-	-

[Reset failed messages](#)

1. Click **Reset failed messages** to reset all failed messages to ready state.

Managing Users



This section outlines how to manage users in the RBMS built-in user repository. If RBMS is connected to an authorization service the users are configured in the authorization service.

Viewing all existing users

To view all existing users

1. Click the **Administration** tab.
2. Click **Users** in the left navigation pane. The list of all existing users appear.

Administration

Manage Leitstand



Images Inventory Metrics Jobs Logs Administration

Logout

- Webhooks
- Users**
- Roles
- Access Keys
- Access Key Validator

Users

Overview of all existing users

Filter

Filter users by name or user ID

User Name	Family Name	Given Name	Email Address
chris	Chris	-	chris@rtbrick.com
joy	Joy	-	joy@rtbrick.com
martin	Martin	-	martin@rtbrick.com
ctrld	ctrld	-	-

Add user

3. Click the name of the user whose details you want to view.

Adding users

You can add a new users to the user repository.

To add a user

1. Click the **Administration** tab.
2. Click **Users** in the left navigation pane. The list of all existing users appear.
3. On the **Users** page, click **Add user**.

4. Specify user details such as username, password, and access token.
5. Click **Add user**.

Removing users

To remove a user

1. Click the **Administration** tab.
2. Click **Users** in the left navigation pane. The list of all existing users appear.
3. Click the name of the user whom you want to remove.
4. On the **User Settings** page, click **Remove user**.

Resetting Password

To reset a user password

1. Click the **Administration** tab.
2. Click **Users** in the left navigation pane. The list of all existing users appear.
3. Click the name of the user whom you want to remove.
4. On the **User Settings** page, click **Reset password**. The **Reset Password** page appears.
5. Enter the new password.
6. Re-type the new password in order to detect accidental typos.
7. Click **Reset Password**.

Managing Roles



This section outlines how to manage roles in the RBMS built-in user repository. If RBMS is connected to an authorization service the roles are be configured in the authorization service. See [Scopes](#) for more information about the existing access scopes.

Viewing list of roles

To view the list of roles

1. Click the **Administration** tab.
2. Click **Roles** in the left navigation pane. The list of all existing users appear.
image::admin_roles.png[]
2. Click the role that you want to view or modify.

Creating Roles

To create a role

1. Click the **Administration** tab.
2. Click **Roles** in the left navigation pane. The list of all existing users appear.
3. On the **Roles** page, click **Add role**.
4. Specify the details of the new role such as role name, Accessible Resource Scopes, and description.
5. Click **Add role**.

Removing roles

To remove a role

1. Click the **Administration** tab.
2. Click **Roles** in the left navigation pane. The list of all existing roles appear.
3. Click the role that you want to remove.
4. On the **Role** <rolename> page, click **Remove role**.

Managing Access Keys

Viewing list of access keys

To view the list of all existing access keys

1. Click the **Administration** tab.
2. Click **Access Keys** in the left navigation pane. The list of all existing access keys appear.

Administration



Manage Leitstand

- Images
- Inventory
- Metrics
- Jobs
- Logs
- Administration

Logout

- Webhooks
- Users
- Roles
- Access Keys**
- Access Key Validator

Access Keys

Listing of all existing access keys.

Filter

Filter access keys by name

Access Keys

List of all issued and valid access keys. An access key can be revoked in order to be invalidated. Access keys are immutable. The modification of an access key requires to issue a new access key.

Name	Date Created	Description
CTRLD	27-MAY-2020 13:59:28.673	Allows CTRLD to update inventory records and to declare tasks as completed.
rtb-image	10-JUN-2020 18:12:38.399	

3. Click the name of the access key that you want to view or modify.

Creating Access Keys

To create an access key

1. Click the **Administration** tab.
2. Click **Access Keys** in the left navigation pane. The list of all existing access keys appear.
3. On the **Access Keys** page, click **Add access key**.

Administration



Manage Leitstand

- Images
- Inventory
- Metrics
- Jobs
- Logs
- Administration

Logout

- Webhooks
- Users
- Roles
- Access Keys**
- Access Key Validator

Access keys

New Access Key

Issue a new access key

Accesskey

Key Name

A unique key name that also forms the user login ID for all requestes authenticated by this key

Scopes

adm
adm.accesskey
adm.accesskey.read
adm.log
adm.read
adm.user
adm.user.read
adm.webhook
adm.webhook.read
ctrlid

4. Specify the details of the new access key such as key name, scopes, and

description.

5. Click **Create access key**.

Revoking an access key

To revoke an access key

1. Click the **Administration** tab.
2. Click **Access Key** in the left navigation pane. The list of all existing access keys appear.
3. Click the access key that you want to revoke.
4. On the <access key name> **accesskey** page, click **Revoke access key**.

Validating an access key

The validating access key feature enables you to validate an encoded access key.

To validate an access key

1. Click the **Administration** tab.
2. Click **Access Key Validator** in the left navigation pane.

Administration

Manage Leitstand

Images Inventory Metrics Jobs Logs Administration



Logout

Access Key Validator
Validate an encoded access key

Encoded Access Key

Access Key

Enter the access key to be validated.

Validate

3. In the **Access Key** text box, enter the access key to be validated.

4. Click **Validate**.

Restoring an revoked access key

To restore an accidentally revoked access key

1. Click the **Administration** tab.
2. Click the **Access Key Validator** in the left navigation pane.
3. Paste the access key to be restored in the text area.
4. Click **Validate**.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

Webhooks

Access Keys

Access Key Validator

Users

Roles

Access Key Validator

Validate an encoded access key

Revoked Access Key

The access key has been revoked. Click the restore button to re-enable the access key.

Done
Restore

5. Click **Restore** to restore the revoked access key.

Administration

Manage webhooks, access keys and user accounts



Images Inventory Metrics Jobs Logs Administration Logout

Webhooks

Access Keys

Access Key Validator

Users

Roles

Access Key Validator

Validate an encoded access key

Access Key ID	51cc8619-5310-4f63-a411-2620a8c8c4ec
Access Key Name	OSS_IT
Date Created	08-SEP-2020 14:17:18.494
Scopes	<ul style="list-style-type: none"> ▪ ivt.read

Done

Scopes

Access to RBMS is granted through an access token. The access token is either issued by an OAuth2 compliant authorization service or by RBMS itself, depending on whether RBMS delegates to an authorization service or the RBMS built-in user repository is used.

The access token conveys the list of *scopes* the user is allowed to access. The table below lists all existing scopes:

Scope	Description
adm	Full access to the RBMS administration API and UI.
adm.read	Readonly access to the RBMS administration API and UI.
adm.accesskey	Full access to the RBMS access key administration API and UI.
adm.accesskey.read	Readonly access to the RBMS access key administration API and UI.
adm.user	Full access to the RBMS user management API and UI.
adm.user.read	Readonly access to the RBMS user management API and UI.
adm.webhook	Full access to the RBMS webhook management API and UI.
adm.webhook.read	Readonly access to the RBMS webhook management API and UI.
ctrlld	Full access to all CTRLD actions that can be triggered from RBMS.
ctrlld.reinstall	Permission to trigger CTRLD to run ZTP sequence for an software image upgrade again.
ctrlld.settings	Permissions to update the CTRLD settings on the switch via RBMS.
ivt	Full access to the resource inventory.

ivt.read	Readonly access to the resource inventory.
ivt.element	Manage elements in the resource inventory.
ivt.element.settings	Manage element settings in the resource inventory.
ivt.element.config	Manage element configuration in the resource inventory.
ivt.element.dns	Manage element DNS records in the resource inventory.
ivt.element.module	Manage element hardware module information in the resource inventory.
ivt.group	Manage element groups in the resource inventory.
ivt.group.settings	Manage element group settings in the resource inventory.
ivt.image	Manage software images in the resource inventory.
ivt.rack	Manage racks in the resource inventory.
job	Full access to the RBMS job API and UI.
job.read	Readonly access to the RBMS job API and UI
job.task	Manage job tasks via RBMS Job API or UI.
tmy	Full access to the RBMS metric API and UI.
tmy.read	Readonly access to the RBMS metric API and UI.
tmy.metrics	Full access to manage RBMS metrics.
tmy.metrics.read	Readonly access to metrics.



Scopes are *cumulative* by convention. For example, the `ivt.element` scope includes the `ivt.element.settings` scope.



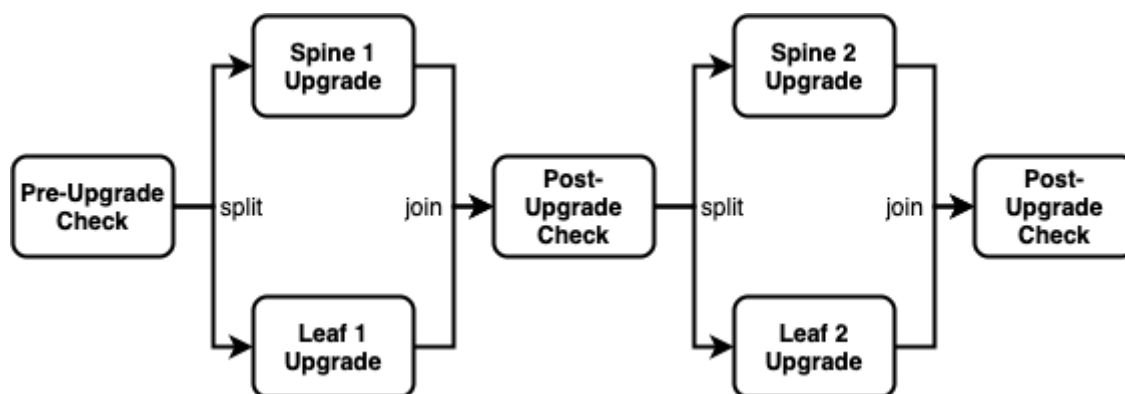
For UI access always grant the read scope in combination with a specific write scope to avoid trouble. For example, grant `ivt.element.settings` in combination with `ivt.read`. Otherwise a user might not be able to navigate to the view to apply the changes.

7.2. Managing Jobs

RBMS includes a job scheduler used by the network management applications to run management jobs. A job is a set of tasks that are executed in a specified order.

The task execution flow is defined by the application creating the job. Tasks can be executed sequentially or in parallel. Parallel execution flows can be joined to continue with a single flow. Technically speaking a job is described as *state engine*. Each task represents a node in the the state engine. The transition between tasks form the execution flow.

The figure below shows a simplified execution flow for a fabric upgrade.



The *Pre-Upgrade Check* task runs all checks to test whether the fabric can be upgraded. If the fabric passes all checks the execution flow is splitted to run the *Spine 1 Upgrade* and *Spine 2 Upgrade* in parallel. The *Post-Upgrade Check* waits for both upgrades to be completed before it runs the checks to test whether the upgrade was successful. If both switche upgrades were successful the job upgrades the two remaining switches in parallel. Finally another *Post-Uograde Check* is executed to check whether the upgrade was successful.

An application can program a job task to wait for an explicit confirmation. For example, an operator might want to inspect the state of a switch when a new image has been installed the very first time in the network. The upgrade application can program the *first* post-upgrade check to wait for confirmation before proceeding with the next upgrade.

The job module is a generic job viewer to inspect the state and progress of scheduled jobs. It also allows to confirm that a job can continue.

Viewing job list

To view the list of jobs

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Optinally filter the job list by job name. The job name can be specied as prefix, full name of regular expression.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration Logout

Job List

Jobs

Review the currently active or scheduled jobs

Filter

Filter

Search for jobs by pod name, job name, element role or state [Show advanced filtering options](#)

Name	Application	Type	State	Scheduled at	Last modified
l1.pod1.blr	ZTP	generate-config	COMPLETED	26-JUN-2020 00:33:15.824	26-JUN-2020 00:33:38.372
l1.pod1.blr	ztp	generate-config	COMPLETED	18-JUN-2020 01:27:24.606	18-JUN-2020 01:27:27.749
l1.pod1.blr	ztp	generate-config	COMPLETED	18-JUN-2020 01:25:49.554	18-JUN-2020 01:26:25.813
l1.pod1.blr	ztp	generate-config	COMPLETED	18-JUN-2020 00:18:03.567	18-JUN-2020 00:18:07.965
l1.pod1.blr	ztp	generate-config	COMPLETED	18-JUN-2020 00:17:03.380	18-JUN-2020 00:17:12.832
l1.pod1.blr	ztp	generate-config	COMPLETED	18-JUN-2020 00:15:41.806	18-JUN-2020 00:16:41.660
l1.pod1.blr	ztp	generate-config	COMPLETED	17-JUN-2020 22:20:10.072	17-JUN-2020 22:20:54.978
l1.pod1.blr	ztp	generate-config	COMPLETED	17-JUN-2020 22:11:49.700	17-JUN-2020 22:12:25.330
l1.pod1.blr	ztp	generate-config	COMPLETED	17-JUN-2020 22:09:16.960	17-JUN-2020 22:09:22.067

Viewing job task list

To view the list of job tasks

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to view. The **Job Tasks** page appears.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration

Logout

Job List

Jobs > Job Tasks

Job Tasks

l1.pod1.blr

Job Info

Settings

Tasks

Flow

Job Summary

General job settings and job process in terms of percentage of completed tasks

Job Application	ztp
Job Type	generate-config
Job Name	l1.pod1.blr
Job Owner	Postman
Job State	COMPLETED
Started at	18-JUN-2020 00:15:41.806

Job Tasks

Review the job tasks and their respective state.

Task List				
Task Type	Task Name	Element	State	Last modified
generate-config	ctrlid	accessleaf l1.pod1.blr	COMPLETED	18-JUN-2020 00:16:41.629
generate-config	running-configuration	accessleaf l1.pod1.blr	COMPLETED	18-JUN-2020 00:16:41.656

Remove

Viewing task flow

The task flow enables you to inspect taskflow and progress of the selected task.

To view the list of task flow

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to view.
3. Click **Flow** in the left navigation pane. The **Taskflow** page appears.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration Logout

Job List

[Jobs > Job Tasks > Taskflow](#)

Taskflow

Inspect taskflow and progress of job l1.pod1.blr

l1.pod1.blr

Job Info -

[Settings](#)

[Tasks](#)

Flow

```

graph TD
    A["generate-config  
accessleaf  
l1.pod1.blr  
null  
COMPLETED"] --> B["generate-config  
accessleaf  
l1.pod1.blr  
null  
COMPLETED"]
  
```

Viewing task details

To view the task details

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to view. The **Job Tasks** page appears.
3. Click the name of the task that you want to view. The **Job Task** page appears.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration Logout

[Job List](#)

Jobs > Job Tasks > Task Details

Task Details

l1.pod1.blr

Job Info +

ZTP

Task Info -

Details

Inspect task ZTP details.

Pod	blr
Element Role	accessleaf
Job Application	ZTP
Job Type	generate-config
Job Name	l1.pod1.blr
Task Type	generate-config
Task Name	ZTP
Task State	COMPLETED
Date Modified	26-JUN-2020 00:33:38.368

```
{
  "element": {
    "group_id": "434e53da-ba28-4c32-9556-d47a490d4d8b",
    "group_name": "blr",
    "group_type": "pod",
    "element_id": "f166b68c-ebf0-49b6-b1a9-6e7757eee4ce",
    "element_name": "l1.pod1.blr",
    "element_role": "accessleaf",
    "administrative_state": "ACTIVE",
    "mgmt_interfaces": {
```

Canceling a Job

To cancel a job

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to view.
3. Click **Tasks** in the left navigation pane. The **Job Tasks** page appears.
4. Click **Cancel job**. The job state changes to cancelled.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration Logout

Job List

I1.pod1.blr

Job Info -

Settings

Tasks

Flow

[Jobs > Job Tasks](#)

Job Tasks

Job Summary

General job settings and job process in terms of percentage of completed tasks

Job Application	ZTP
Job Type	generate-config
Job Name	I1.pod1.blr
Job Owner	martin
Job State	CANCELLED
Started at	14-JUL-2020 11:03:39.563

Job Tasks

Review the job tasks and their respective state.

Task List				
Task Type	Task Name	Element	State	Last modified
generate-config	running-configuration	accessleaf I1.pod1.blr	CANCELLED	14-JUL-2020 11:03:43.663
generate-config	ZTP	accessleaf I1.pod1.blr	CANCELLED	14-JUL-2020 11:03:43.667

Remove
Resume

Removing a Job

To remove a completed, cancelled or failed job

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to view.
3. Click **Tasks** in the left navigation pane. The **Job Tasks** page appears.
4. Click **Remove**. A confirmation dialog is displayed.
5. Click **Confirm** to remove the job.

Configuring Job Settings

To configure the job settings

1. Click the **Jobs** tab. The list of currently active or scheduled jobs appear.
2. Click the name of the job that you want to configure.

Jobs

Manage jobs and tasks



Images Inventory Metrics **Jobs** Logs Administration Logout

Job List

I1.pod1.blr

Job Info -

Settings

Tasks

Flow

Jobs > I1.pod1.blr **READY**

Job Settings

Settings of job I1.pod1.blr

General Settings

Job Name

A human-friendly job name explaining the purpose of this job.

Job Owner

The user who has scheduled this job.

Start job immediately
Starts the job immediately. This option is intended for launching processes in a maintenance window.

Start job at 15-JUL-2020 11:16
Schedule a job at the selected date.

Suspend update when not completed until 15-JUL-2020 15:16
Suspend execution of all remaining tasks that have not been completed until the selected date.

Save settings

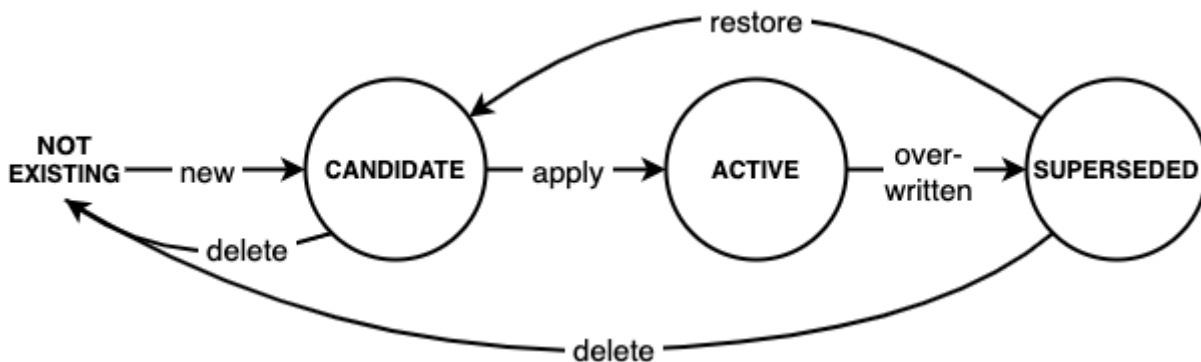
3. Make necessary configurations for the job.
4. Click **Save settings**.

8. RBMS Configuration Store

The RBMS resource inventory includes a configuration store to maintain switch configurations. The configuration store can store an arbitrary number of configurations per switch and provides a history of up to 50 revisions for each configuration.

RBFS stores the active configuration in RBMS after every configuration change.

The figure below shows the configuration lifecycle.



A new configuration is considered a *candidate* configuration. A candidate configuration turns into the *active* configuration when being applied to the switch. The previously activate configuration is marked as *superseded* at the same time. A superseded configuration can be restored by creating a new candidate configuration from it and applying the candidate configuration again. Candidate and superseded configurations can be removed from the configuration store. The active configuration cannot be deleted.

RBMS provides means to create or upload new configurations and apply them to the switch. In addition, RBMS UI allows inspecting configuration changes in the configuration history.

Creating a new candidate configuration

Uploading a candidate configuration

The simplest way to add a new candidate configuration is to upload a new configuration to the configuration store.

To upload a new candidate configuration

1. Click the **Inventory** tab.
2. Click **Elements** in the left navigation pane. The element list appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory -

- Pods
- Elements**
- Interfaces
- Facilities
- Racks
- DNS Zones

Administration +

Elements

Overview of all registered elements.

Filter

 Filter

3. Click the element name for which you want to upload a new configuration.
4. Click **Configuration** in the left navigation pane. The configurations list appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

- Pod
- Element -
- Element Settings
- CTRLD
- Configuration**
- Environments

Element Configurations

List of all existing element configurations

blr

Pod +

accessleaf I1.pod1.blr

Element -

Filter

Filter

No configurations found.

No configurations for the selected element found.

Add configuration

5. Click **Add configuration**. The New Configuration page appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration [Logout](#)

Inventory +

Element Configurations > New Configuration

New Configuration

Create a new configuration for element l1.pod1.blr .

blr

Pod +

accessleaf l1.pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration**
- Environments
- Location
- Dashboards
- Metrics

New Configuration
Creates a new candidate configuration for an existing configuration or add a new configuration.

Drop your configuration file onto the dashed region or select the configuration file in the file dialog.

[Select configuration file](#)

- Drop a configuration file onto the dashed area or click **Select configuration file** to open the file dialog to select a file.
- Review the configuration in the preview.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory + Element Configurations > New Configuration

New Configuration

Create a new configuration for element l1.pod1.blr .

New Configuration
Creates a new candidate configuration for an existing configuration or add a new configuration.

```
{
  "running-configuration": {
    "system:rtbrick": {
      "system-time-type": "GMT",
      "load-last-config": "True",
      "snapshot-logd": "False",
      "host-name:rtbrick": {
        "element-name": "l1.pod1.blr",
        "pod-name": "blr"
      }
    },
    "ctrlld": [
      {
        "ipv4-address": "10.0.3.1",
        "port": 19091
      }
    ]
  },
  "log": [
    {
      "bd_module_logmap:all bds all": {
        "level": "none"
      }
    },
    {
      "bd_module_logmap:all pubsub all": {
        "level": "none"
      }
    }
  ],
  "time-series": [
    {
      "metric:chassis-temperature-111111111111": {
```

Dismiss

Configuration Name

Descriptive configuration name

Content Type
 ⌵
The MIME-Type of the application

Comment

Comment on the applied configuration changes

Cancel Save configuration

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory + **Element Configurations**
List of all existing element configurations

blr
Pod +

accessleaf I1.pod1.blr

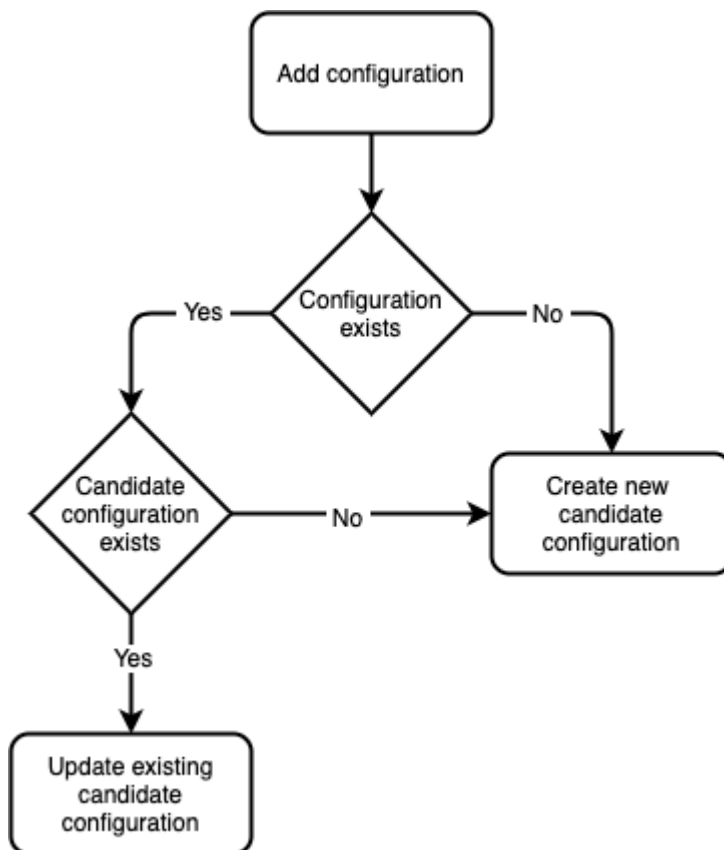
Filter
Filter configurations by their name. Filter

Config	State	Creator	Date Modified	Comment
running-config	CANDIDATE	martin	23-JUN-2020 22:27:30.262	Download

Add configuration

Element -
[Element Settings](#)
[CTRLD](#)
[Configuration](#)
[Environments](#)

RBMS either creates a new candidate configuration or updates an existing candidate configuration if a candidate configuration for the specified configuration name exists. The complete flow is illustrated below:



Generating a candidate configuration

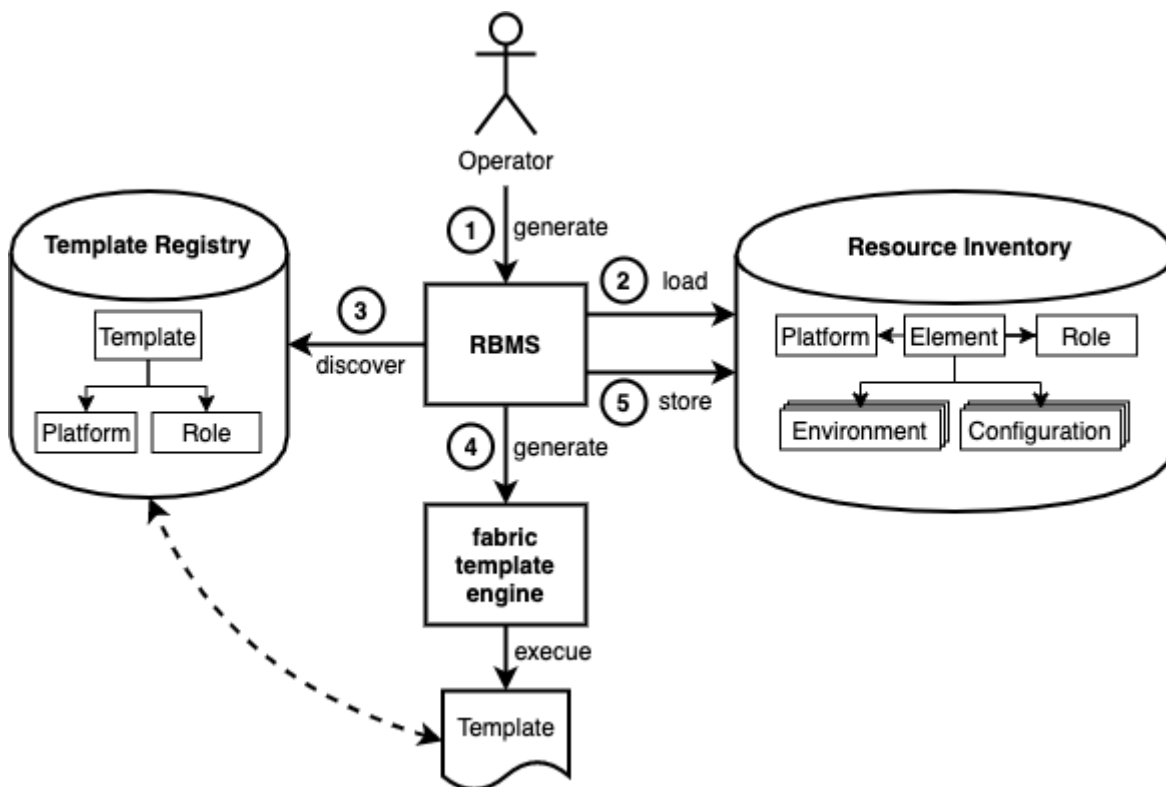
RBMS includes a template engine to generate switch configurations. The configuration templates are designed based on your conventions for building a

fabric and then added to the template engine. The templates need to be registered in RBMS to make them eligible for execution. RBMS passes the following information to the template engine:

- the general element settings
- the software image to be installed on the switch and
- all registered environments

RBMS can store an arbitrary set of *environments* per element. An environment is a JSON object containing parameters processed in the template. The structure and number of environments is defined by the template author.

The figure below shows the configuration generation flow:



1. The operator triggers the switch configuration generation in RBMS.
2. RBMS loads the element settings, environments and image information from the resource inventory.
3. RBMS discovers the templates eligible for the selected switch.
4. RBMS invokes the template engine for each discovered template.
5. RBMS stores the generated configuration as candidate configuration in the configuration store.

Template registration

The template registration maintains the list of templates available in the template engine.

Say you have three different templates:

- `ztp.gojson` generates the ZTP configuration snippet for a switch.
- `leaf-running-configuration.gojson` generates the running-configuration of a leaf switch.
- `spine-running-configuraton.gojson` generates the running-configuration of a spine switch.

All templates need to be registered in RBMS to get executed when the switch configuration is generated.

To register a new template

1. Click the **Inventory** tab.
2. Click **+** to expand the **Administration** menu.
3. Click **Templates** in the left navigatio pane. The template list appears. ZTP and spine templates are already registered.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels
- Templates**

Templates

Manage configuration templates.

Templates

 Filter

Filter templates by their name

Template	Description
spine-running-configuration	Spine running configuration template
ztp	ZTP service configuration template.

[Add template](#)

4. Click **Add template** to register the leaf template.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels
- Templates**

Templates > New template

New Template

Register a new template

General Settings

Template Name
leaf-running-configuration
The name of the template

Configuration Name
running-configuration
The name of the generated configuration

Description
Leaf running configuration template
An optional template description

Binding

Element Roles
accessleaf
borderleaf
spine
integrationtest
Restrict this template to the selected element roles.

Platforms
EC5916-54XK
EC5916-54XKS
Wedge-100BF-32X
Wedge-100BF-65X
Restrict this template to the selected platforms.

Cancel Add template

5. In **Template Name**, enter *leaf-running-configuration*.



The template name in RBMS matches the template folder name in the template engine. See the [template engine guide](#) for more details.

6. In **Configuration Name**, enter *running-configuration* since the template generates the *running-configuration* for a switch.

7. In **Element Roles**, select *accessleaf* to bind the template to accessleaf switches.

- In **Description**, optionally describe the template and the generated configuration.
- Click **Save template**. The template list appears. The new template is listed.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Administration -

- Platforms
- Roles
- Dashboards & Panels
- Templates**

Templates

Manage configuration templates.

Templates

 Filter

Filter templates by their name

Template	Description
leaf-running-configuration	Leaf running configuration template
spine-running-configuration	Spine running configuration template
ztp	ZTP service configuration template.

Add template

Environment management

Adding a new environment

To add a new environment

- Click **Environments** in the left navigation pane. The list of environment appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

blr

Pod +

accessleaf I1_pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration

Element Environments

List of environments declared for this element.

No environments found.

No environments have been registered for this element.

Add environment

- Click **Add environment**. The New Environment view appears.



Inventory

Manage network elements

Manage software images

Images | **Inventory** | Metrics | Jobs | Logs | Administration Logout

Inventory +

Element Environments > New environment

New Environment

Create new environment for element i1.pod1.bir

Environment

Environment Name
bgb-peering
The environment name is unique per element

Category
RBFS
Optional environment category.


Type

Optional environment type.

Description
BGP Instances
Optional environment description.

powered by ace

```
1 {
```

3. In **Environment Name** enter the name of this environment.
4. In **Category** enter *RBFS*.
 Only RBFS environments are processed by the template engine.
5. In **Type** enter an optional type or schema information of the environment.
6. In **Description** enter an optional environment description.
7. Enter the environment variables in the displayed JSON editor or click **Upload new environment** to upload a JSON file.
8. Click **Add environment** to add a new environment.

Editing an environment

To update an existing environment

1. Click **Environments** in the left navigation pane. The list of environment appears.
2. Click the name of the environment you want to edit.
3. Apply the modifications to the environment.
4. Click **Save environment** to save the environment.

Removing an environment

To remove an environment

1. Click **Environments** in the left navigation pane. The list of environment appears.
2. Click the name of the environment you want to remove.
3. Click **Remove environment**. A confirmation dialog is displayed.
4. Click **Confirm** to remove the environment.

Configuration generation

To generate a configuration

1. Click **Actions** in the left navigation pane.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Pods > blr > Elements > I1.pod1.blr > CTRLD

Actions for element I1.pod1.blr

Execute management actions on I1.pod1.blr

blr

Pod +

accessleaf I1.pod1.blr

Element -

- Element Settings
- CTRLD**
- Configuration
- Environments
- Location
- Dashboards
- Metrics
- Images
- Services
- Physical Interfaces
- Logical Interfaces

Ping

Test whether CTRLD switch management API is reachable. Ping

Configuration

Generate switch configuration and provision the ZTP service. **Generate configurations**

ACCEPTED Scheduled job to generate the element configuration.

ZTP

Copy configurations to ZTP server. Provision ZTP

Running Configuration Update

Merge candidate running configuration into the current running configuration. Daemons that receive a configuration change might be restarted. **Apply running-configuration**

Software Upgrade

Run Zero-Touch Provisioning (ZTP) to upgrade the switch. **Upgrade**

2. Click **Generate configurations**. RBMS schedules a configuration generation job. The job contains a task for each configuration and invokes the template engine to create the configuration.

Jobs

Manage the metrics sampled from network devices



Images Inventory Metrics **Jobs** Logs Administration Logout

[Job List](#)

[Jobs > Job Tasks](#)

Job Tasks

Job Summary
General job settings and job process in terms of percentage of completed tasks

Job Application	ZTP
Job Type	generate-config
Job Name	I1.pod1.blr
Job Owner	martin
Job State	COMPLETED
Started at	23-JUN-2020 10:27:30.105

Job Tasks
Review the job tasks and their respective state.

Task List

Task Type	Task Name	Element	State	Last modified
generate-config	running-configuration	accessleaf I1.pod1.blr	COMPLETED	23-JUN-2020 10:27:30.275
generate-config	ZTP	accessleaf I1.pod1.blr	COMPLETED	23-JUN-2020 10:27:30.351

[Remove](#)



The generated configurations are stored as new candidate configurations.

To review the generated configuration

1. Click **Configuration** in the left navigation pane. The configuration list shows new candidate configurations.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

blr

Pod +

accessleaf l1.pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration**
- Environments
- Location
- Dashboards

Element Configurations

List of all existing element configurations

Filter

 Filter

Filter configurations by their name.

Config	State	Creator	Date Modified	Comment
running-config	CANDIDATE	martin	23-JUN-2020 10:27:30.262	Download
ZTP	CANDIDATE	martin	23-JUN-2020 10:27:30.338	Download

[Add configuration](#)

2. Click the timestamp of the generated configuration to show the configuration content. Click the configuration name to display the configuration history.

Review configuration history

To review configuration changes in the configuration history

1. Click the **Inventory** tab. The list of pods appears.
2. Click **Element** in the left navigation pane. The element list appears.
3. In **Filter** enter the name of the switch and click **Filter**. The list of matching elements appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory -

- Pods
- Elements**
- Interfaces
- Facilities
- Racks
- DNS Zones

Administration +

Elements

Overview of all registered elements.

Filter

 Filter

Select the property to filter elements for. [Show advanced filtering options](#)

Pod	Element	Alias	Adm.-State	Tags
blr	l1.pod1.blr		ACTIVE	BLR RBMS lab

1 element(s) found.

- Click the element name for which you want to inspect configuration changes. The element settings appears.
- Click **Configurations**. The list of configurations appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Pod +

accessleaf I1.pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration**
- Environments
- Location

Element Configurations

List of all existing element configurations

Filter Filter

Filter configurations by their name.

Elements	Config	State	Creator	Date Modified	Comment	
	CTRLD	ACTIVE	karthik	24-JUN-2020 06:51:28.609	RBMS upgrade	Download
	running-configuration	ACTIVE	karthik	24-JUN-2020 07:37:17.079	Commit Immediate	Download

Add config

- Click the configuration name. The configuration history appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory +

Pod +

accessleaf I1.pod1.blr

Element -

- Element Settings
- CTRLD
- Configuration**
- Environments
- Location

Element Configurations > running-configuration

Element Configuration History

running-configuration configuration history of I1.pod1.blr

Revisions

Compare	State	Author	Last Modified	Comment	
<input checked="" type="radio"/>	ACTIVE	karthik	24-JUN-2020 07:37:17.079	Commit Immediate	Download
<input type="radio"/>	SUPERSEDED	karthik	24-JUN-2020 07:35:23.307	Commit Immediate	Download
<input type="radio"/>	SUPERSEDED	karthik	24-JUN-2020 07:34:45.516	Commit Immediate	Download
<input type="radio"/>	SUPERSEDED	karthik	24-JUN-2020 07:33:03.198	Commit via Rest	Download
<input type="radio"/>	SUPERSEDED	karthik	24-JUN-2020 07:21:31.216	Commit via Rest	Download

- Select the two configurations you want to compare and click the **Compare** button. The diff viewer appears.

Inventory

Manage network elements



Images **Inventory** Metrics Jobs Logs Administration Logout

Inventory + [Element Configurations > running-configuration > Element Configuration Revisions Comparator](#)

I1.pod1.blr running-configuration diff

Compare running-configuration changes between 24-JUN-2020 07:37:17.079 and 24-JUN-2020 07:34:45.516

	24-JUN-2020 07:37:17.079	24-JUN-2020 07:34:45.516
blr		
Pod		
accessleaf I1.pod1.blr		
Element		
Element Settings		
CTRLD		
Configuration	<pre>{ "running-configuration": { "system:rtbrick": { "system-time-type": "GMT", "load-last-config": "True", "snapshot-logd": "False", "host-name:rtbrick": { "element-name": "I1.pod1.blr", "pod-name": "blr" } }, "ctrlld": [{ "ipv4-address": "10.0.3.1", "port": 19091 }] }, "log": [{ "bd_module_loggroup:all bds all": { "level": "None" }, "bd_module_loggroup:all pubsub all": { "level": "None" } }], "time-series": [{ "metric:chassis_fan_speed_rpm": { "metric-bds-type": "object-metric", "prometheus-type": "naive" } }] }</pre>	<pre>{ "running-configuration": { "system:rtbrick": { "system-time-type": "GMT", "load-last-config": "True", "snapshot-logd": "False", "host-name:rtbrick": { "element-name": "I1.pod1.blr", "pod-name": "blr" } }, "ctrlld": [{ "ipv4-address": "10.0.3.1", "port": 19091 }] }, "log": [{ "bd_module_loggroup:all bds all": { "level": "None" }, "bd_module_loggroup:all pubsub all": { "level": "None" } }], "time-series": [{ "metric:chassis_fan_speed_rpm": { "metric-bds-type": "object-metric", "prometheus-type": "naive" } }] }</pre>
Environments		
Location		
Dashboards		
Images		
Services		
Physical Interfaces		
Logical Interfaces		

The diff viewer shows both configurations and the diff if you scroll down.

```
{
  "physical:ifp-0/0/52": {
    "speed": "40G"
    "auto_negotiation": true
  }
}
```

9. RBMS Template Engine

The RBMS Template Engine is an execution engine for templates. A folder in the filesystem serves as template storage for the engine. The content of the folder follows a convention.

Template Folder structure

Template folder structure

```
templates
|-- includes
|   |-- <include-template>.gojson
|-- <template name>
|   |-- config.yaml
|   |-- <include-template>.gojson
|   |-- <main-template>.gojson
```

The template engine uses one templates folder where all the templates are stored. Each template resides in his own folder, the folder name is the template name. The `config.yaml` file inside a template folder indicates that this folder is a template. In this file also other configurations for the template engine can be made.

The template folder contains one `main-template` and can contain multiple `include-templates`. The include-templates can be included into the main template.

Folders that don't contain a `config.yaml` are not treated as templates. This folders can be used as containers for other include-template files.

Template config

This section describes the `config.yaml` file. Image this folder structure for the next examples.

Simple example folder structure

```
templates
|-- includes
|   |-- global_include.gojson
|-- sample
|   |-- config.yaml
|   |-- local_include.gojson
|   |-- main.gojson
```

Here there is a main-template which includes the local_include-template and the global_include-template. The `config.yaml` is used by the template engine to parse the right files, so that the include-directives work.

templates/sample/config.yaml

```
engine: golang
main_template: "main.gojson"
main_pattern: "*.gojson"
include_pattern: "includes/*.gojson"
post_processors:
  - removeTrailingCommas
  - prettyJSON
```

config.yaml attributes

Attribute	Default	Description
engine	golang	selects the template engine, at the moment only golang is supported
main_template	none	points to the entrypoint of the rendering process, this template is used as the top most, it has to be included in the main pattern.
main_pattern	none	describes which files the engine should parse from the template folder.
include_pattern	none	describes which files the engine should additionally parse relative to the templates folder.
post_processors	none	allows to specify post processors that are used in that order on top of the generated output.

Post processors

Attribute	Description
removeTrailingCommas	removes in json files the commas which are not valid, this makes the template much easier
removeEmptyLines	removes empty lines
prettyJSON	Pretty converts the input json into a more human readable format where each element is on its own line with clear indentation

Attribute	Description
uglyJSON	Ugly removes insignificant space characters from the input json byte slice and returns the compacted result.

GO Lang Template Engine

The default engine is the golang template engine. This gives some links to more detailed information.

The GO Lange template engine is based on:

- [GoLang test template](#)
The golang text template engine. This allows evaluating arguments, execute actions and include other templates.
- [sprig functions](#)
Beside of the default functions golang already provides, the sprig function library is added to the engine.

Example

This section shows a simple example, that covers a lot of functionality of the templates.

The example uses the following folder structure. Each file will be described in more detail.

Full example folder structure

```
templates
|-- includes
|   |-- global_include.gojson
|-- sample
|   |-- config.yaml
|   |-- example_variables.json
|   |-- local_include.gojson
|   |-- main.gojson
```

The template is called sample, because there is a config.yaml in the folder sample.

templates/sample/config.yaml

```
engine: golang
main_template: "main.gojson"
main_pattern: "*.gojson"
```

```
include_pattern: "includes/*.gojson"
post_processors:
  - removeTrailingCommas
  - prettyJSON
```

The `config.yaml` file states that the `main_template` is called `main.gojson`, so that's the entrypoint for the generation.

The `main_pattern` defines this files `templates/sample/*.gojson` should be parsed into the template engine, so also the `main_pattern` is included.

The `include_patterns` defines this files `templates/includes/*.gojson` should be parsed into the template engine.

The `post_processors` are used to remove the trailing commas and make the JSON output more readable.

Let's expect the following example variables structure.

templates/sample/example_variables.json

```
{
  "description": "sample",
  "interfaces": [
    {
      "name": "ifp_0/0/1",
      "ipv4": "127.0.0.1",
      "x": 5,
      "y": 3
    },
    {
      "name": "ifp_0/0/2",
      "ipv4": "127.0.0.2",
      "x": 4,
      "y": 4
    }
  ]
}
```

The these variables can be used to fill a template.

templates/sample/main.gojson

```
{{define "t1"}}
  "hostname": "static",
{{end}}
{
  {{template "t1"}}
  "description": "{{.description}}",
  "interfaces": {
    {{template "local_include.gojson" .interfaces}}
  },
  "list": {{template "global_include.gojson" .}}
```



```
}

```

This templates starts with a definition of a new template **t1** that will be used in this template.

This template **t1** is included immediately after **{**.

Then the description is added, the selection from the variable is done via the **.description**.

For the interfaces we use the local template **local_include.gojson**, the variables that are forwarded to the template are the **.interfaces** so only the array of the original variable set.

To render the list we include the **global_include.gojson** template and forward the original variable set.

templates/sample/local_include.gojson

```
{{range .}}
  "{{.name}}": {
    "ip": "{{.ipv4}}",
    "1000/x*y": {{div 10000 (mul .x .y) }},
  },
{{end}}
```

The **local_include.gojson** iterates over the interfaces list and prints the name and ip-address of the interface.

Also a simple computation is done by using the **sprig** functions **div** and **mul**.

templates/includes/global_include.gojson

```
[
  {{range .interfaces}}"{{.name}}",{{end}}
]
```

The **global_include.gojson** iterates over the interfaces list and prints in an array.



This json template does not create a valid json. The commas are not set correct. The document is not well formatted. Therefore it is easier to create the templates. To create a syntactically correct and well formatted document we use post processors. The syntax is corrected by `removeTrailingCommas`` post processor. The format is corrected by the `prettyJSON` post processor.

The next source block shows the expected outcome when applying the variables from above to the template.

templates/sample/example_result.json

```
{
  "description": "sample",
  "hostname": "static",
  "interfaces": {
    "ifp_0/0/1": {
      "1000/x*y": 666,
      "ip": "127.0.0.1"
    },
    "ifp_0/0/2": {
      "1000/x*y": 625,
      "ip": "127.0.0.2"
    }
  },
  "list": [
    "ifp_0/0/1",
    "ifp_0/0/2"
  ]
}
```

TestKit

In order to do a fast template prototyping we developed a test kit. The test kit allows to execute a template with a given variable set and validate the outcome against an expected result.

To execute we have to specify:

- `templatePath`: Template main folder (default ".")
- `template`: Template name
- `format`: File format [txt, json, json5] (default "txt")

So for example if we execute `template-engine-test -template sample -test example -format json` inside the templates folder, this command will execute the `sample` template with the content of the `example_variables.json` file as input variables. After execution the outcome is stored in the `example_got.json` file, and validated against the `example_result.json` file. The format not only specifies the file endings, it also specifies how the validation is done. So for example the json format does not care about ordering of whitespace differences.

10. Auto-DNS Provisioning

10.1. DNS Introduction

The Domain Name Service (DNS) assigns names to IP addresses of network resources such as hosts, routers, or services. Names are easier to memorize than IP addresses, especially when intuitive naming conventions have been established. Moreover, hostnames form the common name (CN) of a certificate and are needed to issue valid certificates in order to enable Transport Layer Security (TLS).

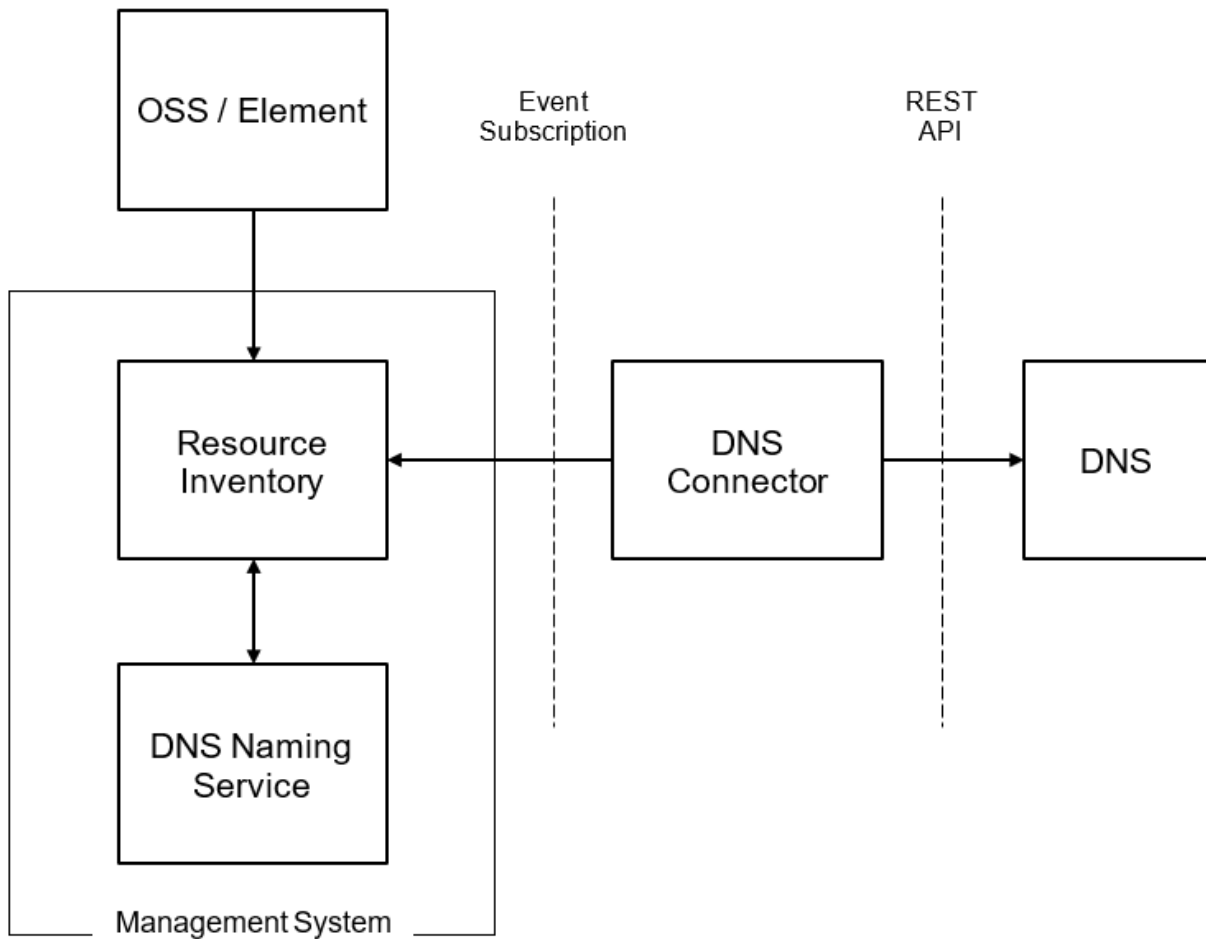
The management system keeps track of what elements exist in the network. Consequently, the management system shall be connected to the DNS such that DNS records are updated automatically whenever needed. This document discusses how to connect the management system to a DNS infrastructure.

Supported Platforms

Not all features are necessarily supported on each hardware platform. Refer to the *Platform Guide* for the features and the sub-features that are or are not supported by each platform.

10.2. DNS Overview

Figure 1 gives an overview of the network management system. The resource inventory forms a cornerstone of the network management system and maintains inventory records for all elements installed in the network. Operation support systems (OSS) or the element itself feed the resource inventory with data. The resource inventory issues a domain event whenever the state of an inventory record has changed.



The DNS Naming Service subscribes all domain events that have an effect on the DNS name of an element and updates the DNS name accordingly. The DNS connector subscribes domain events related to DNS name changes and invokes the REST-API of the DNS infrastructure to update the DNS records. The focus of this document is the contract between the DNS Connector and the management system.

Terms and Definitions

Element

An element represents either a physical or a virtual resource in the network. For example, the switches forming the fabric are physical resources whereas a virtual machine on a compute node is a virtual resource. DNS records are created for elements, but not necessarily every element requires a DNS record.

Element Role

The element role describes the function of an element in the network. Leaf switch,

Spine switch and OLT are examples of element roles.

PoD

The Point of Distribution aggregates all subscribers of a geographic region and connects them to the core network.

Resource Inventory

The resource inventory stores all elements, including their role, configuration, resources, capabilities, operational and administrative state. It also allows to group elements.

The resource inventory is fed by planning processes, which add planned elements to the inventory, and all active elements, which register themselves in the resource inventory and report configuration or operational state changes to the resource inventory.

Domain Event

A domain event describes a change that has taken place in the resource inventory. The domain event contains the information of what has changed as well as the identifiers needed to read additional information from the resource inventory.

Operation Support System (OSS)

The Operation Support System supports to control, monitor, manage and plan the network.

Management System

The management system allows the execution of management functions on the elements forming the network and inspects the configuration and operational state of an element.

DNS Naming Service

The DNS naming service assigns DNS names to all elements in the resource inventory that are supposed to have a DNS name.

Resource Record and Resource Record Set

A resource record is the basic DNS configuration entity. The resource record set contains all resource records related to the same DNS name.

DNS Connector

The DNS connector creates resource record sets from all assigned DNS names.

10.3. DNS Record Management

DNS Naming Convention

A sophisticated DNS naming convention leads to intuitive and concise names and supports the use of wildcard certificates, which in turn simplifies certificate management.

The prerequisite to creating a DNS naming convention is to decide which element roles and services require a DNS name.

Forward DNS Lookup

This section summarizes DNS aspects of common element management patterns briefly, without discussing the advantages and disadvantages of each management pattern in general. A sustainable DNS provisioning solution must support all common management patterns.

Out-of-Band Management

Out-of-Band management establishes a dedicated management network to manage the network elements. Each element is connected via the management interface to this network. The management system also needs to be connected to the management network to access the network elements.

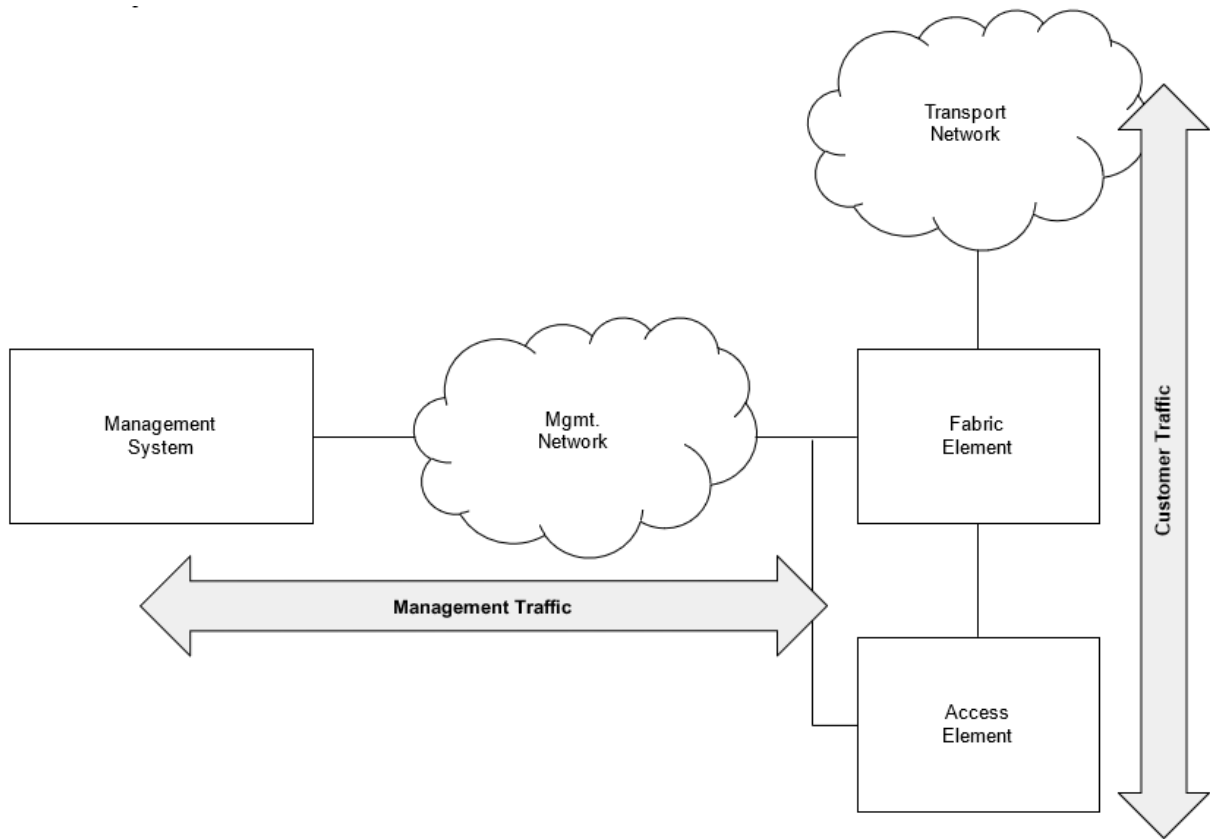


Figure 8. Schematic illustration of the out-of-band management pattern

The DNS lookup resolves the IP address of the management interface.

In-Band Management

In-band management leverages the transport network, which conveys the customer traffic, and also for network management.

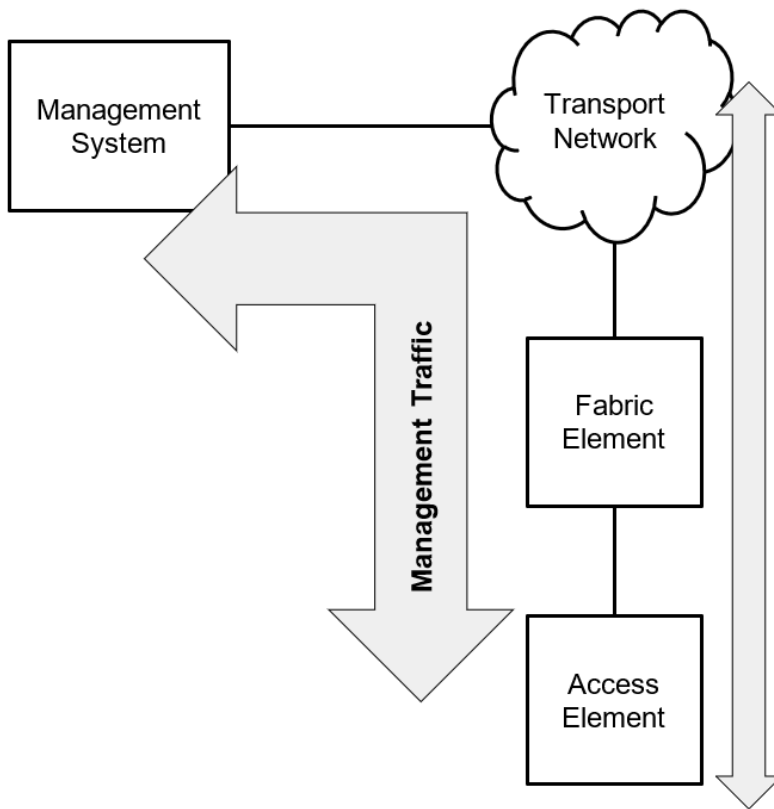


Figure 9. Schematic illustration of the in-band management pattern

The DNS lookup resolves the transport layer loopback IP address for all elements connected with more than one interface to the transport network. For elements with a single connection to the transport network, the IP address assigned to this interface is resolved.

API Gateway

An API gateway forms a single entry point to access multiple elements. The management system interfaces with the API gateway and the API gateway forwards the API call to the appropriate element. The API gateway is either accessed in-band or out-of-band.

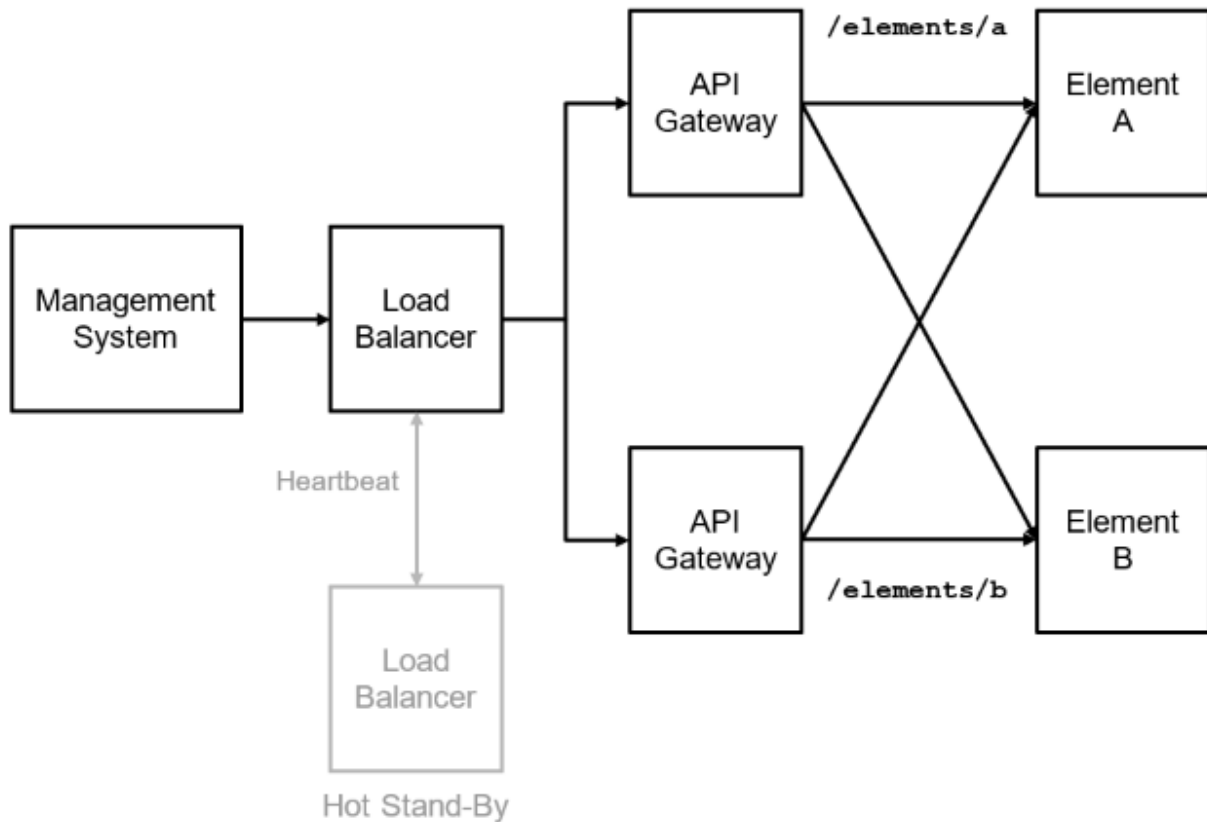


Figure 10. Schematic illustration of the API gateway pattern

The API gateway addresses non-functional requirements such as TLS termination, authentication and authorization.

Many instances of an API gateway are typically started at the same time to avoid creating a single point of failure. All instances are accessible through the same IP address, the cluster IP address. Consequently, the API Gateway DNS name resolves the cluster IP address. Additional DNS records per gateway instance are required to access a gateway instance for troubleshooting.

The management system needs to know the REST API endpoint URL for every element. The REST API invocation is the same for all management patterns outlined before. A management process reads the REST API endpoint from the element record in the resource inventory to invoke the API. Looking at the API gateway pattern, an element does not have to have a DNS name despite providing a REST API endpoint.

Reverse DNS Lookup

A reverse DNS lookup discovers the hostname for a given IP address. Reverse DNS lookups are handy for troubleshooting as they translate an IP address to a human-

friendly name. Consequently, a hostname needs to be assigned to the transport layer loopback IP address or the IP address of the interface connected to the transport network respectively.

DNS Naming Service

The DNS naming service implements the DNS naming convention and assigns a DNS name to all elements that are supposed to have a DNS name.



DNS Naming Service is beyond the scope of this document as it depends on the DNS Naming Convention.

The resource inventory will be enhanced to store zero, one or more DNS records per element as illustrated below. The DNS Naming Service maintains the DNS records per element.

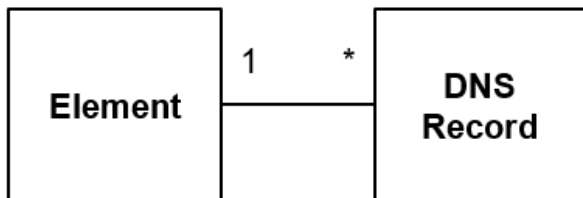


Figure 11. Relation between element and DNS record

A DNS record basically consists of:

- DNS name
- IPv4/v6 address
- status flag indicating whether the DNS record is enabled
- creation date and last modification date and
- an optional expiry date (defaults to never expire)

The resource inventory fires a domain event when a DNS record set has been added, modified or removed (**DnsRecordSetModifiedEvent**).

The following information is conveyed with a DNS record domain event:

- **event_id** - the event ID in UUIDv4 format.
- **event_name** - the event name (DnsRecordSetModifiedEvent).
- **group_id** - the element group ID in UUIDv4 format

- **group_type** - the element group type (set to pod)
- **group_name** - the element group name
- **element_id** - the ID, in UUIDv4 format, of the element whose DNS name has been modified
- **element_name** - the name of the element whose DNS name has been modified
- **element_alias** - an optional alias of the element whose DNS name has been modified. This property is omitted if no alias has been set.
- **element_role** - the role of the element whose DNS name has been modified
- **group_id** - the ID of the group the element is a member of
- **group_name** - the name of the group the element is a member of
- **group_type** - the type of the group the element is a member of, which is always set to pod
- **dns_recordset** - the DNS record set
 - dns_zone_id** - the DNS zone ID in UUIDv4 format of the DNS zone in the resource inventory.
 - dns_zone_name** - the canonical DNS zone name
 - dns_name** - the canonical DNS name that shall be stored in the DNS
 - dns_ttl** - the optional time-to-live for the DNS record (in seconds).
 - dns_withdrawn_name** - the DNS name that shall be removed from the DNS
 - dns_type** - the DNS record type (e.g. A, AAAA, CNAME)
 - dns_record** - the array of DNS records
 - dns_value** - the DNS record value (e.g. IPv4 address, IPv6 address, alias)
 - dns_setptr** - a flag indicating whether to create a PTR record
 - disabled** - a flag indicating whether this record is disabled

In addition, the resource inventory fires an event if a DNS zone was added (**DnsZoneCreatedEvent**) or removed (**DnsZoneRemovedEvent**). A DNS zone event merely contains the zone ID and canonical DNS zone name:

- **event_id** - the event ID in UUIDv4 format.

- **event_name** - the event name (DnsZoneCreatedEvent or DnsZoneRemovedEvent).
- **dns_zone** - the DNS zone that was subject to the reported change
 - dns_zone_id** - the DNS zone ID in UUIDv4 format of the DNS zone in the resource inventory.
 - dns_zone_name** - the canonical DNS zone name

DNS Connector

The DNS Connector subscribes to the DNS domain events outlined before using providing a REST API endpoint that accepts **HTTP POST** requests with the domain event as a requesting entity. The DNS connector creates and removes DNS zones according to the DNS zone events and translates the DNS record set domain event to resource record sets using the following semantics:

- Store a resource record set in the DNS if the **dns_name** property is present. The **dns_name** property value becomes the resource record set name. The resource record set contains up to two resource records: a type A record for the IPv4 address and another type AAAA record for the IPv6 address. The connector derives the time-to-live (TTL) of the resource record set from the expiry date. The TTL is omitted if no expiry date has been specified.
- Remove a resource record set from the DNS if the **dns_withdrawn_name** property is present. The **dns_withdrawn_name** property contains the name of the resource record set to be removed.

The DNS Connector uses the REST API provided by PowerDNS, an open-source DNS server, to maintain the DNS records. PowerDNS can either be configured as a DNS server or merely acts as a gateway and forwards all DNS changes to the actual DNS service.

Figure 6 illustrates the DNS records provisioning flow.

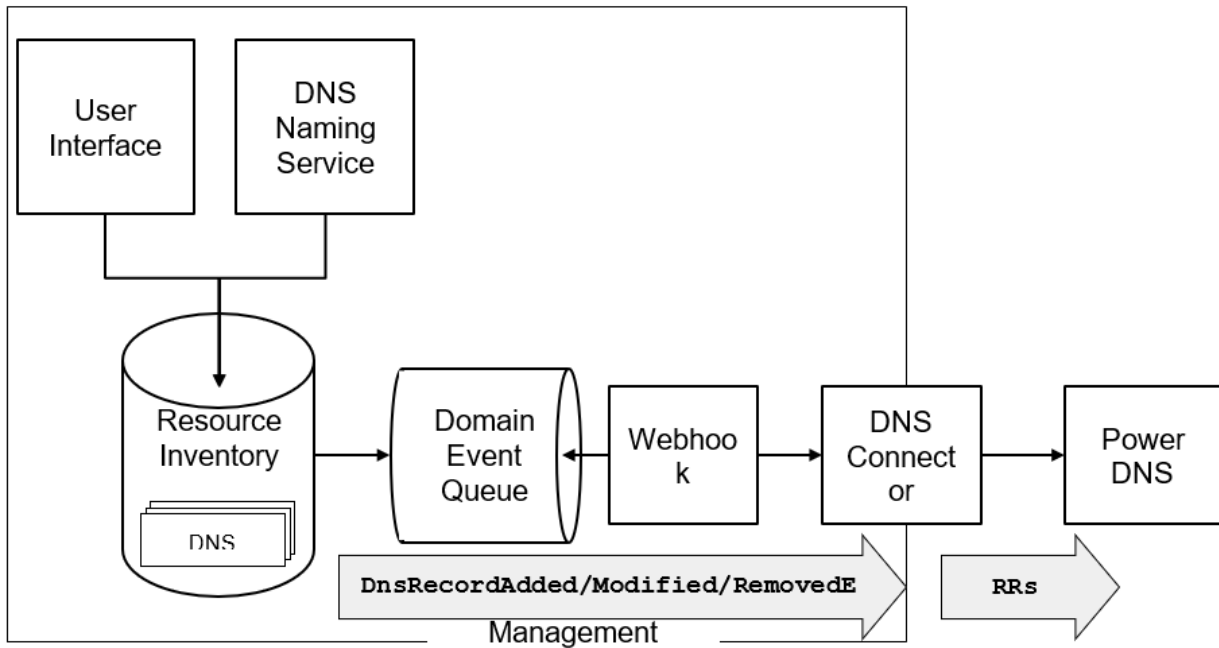


Figure 12. DNS Resource Record Set provisioning flow

The DNS records are stored in the resource inventory and maintained by the DNS Naming Service application or manually through the user interface. All DNS record changes fire an event, which is stored in the Domain Event Queue. The domain event queue is a persistent and transactional queue, which means that domain events are stored for successfully committed transactions only. A webhook, which is a configurable service, forwards all DNS events to the DNS connector, which in turn maintains the resource records sets (RRs) in PowerDNS.

Sample Fabric DNS Naming Convention

This section introduces a sample fabric DNS naming convention. The DNS name tree is depicted below.

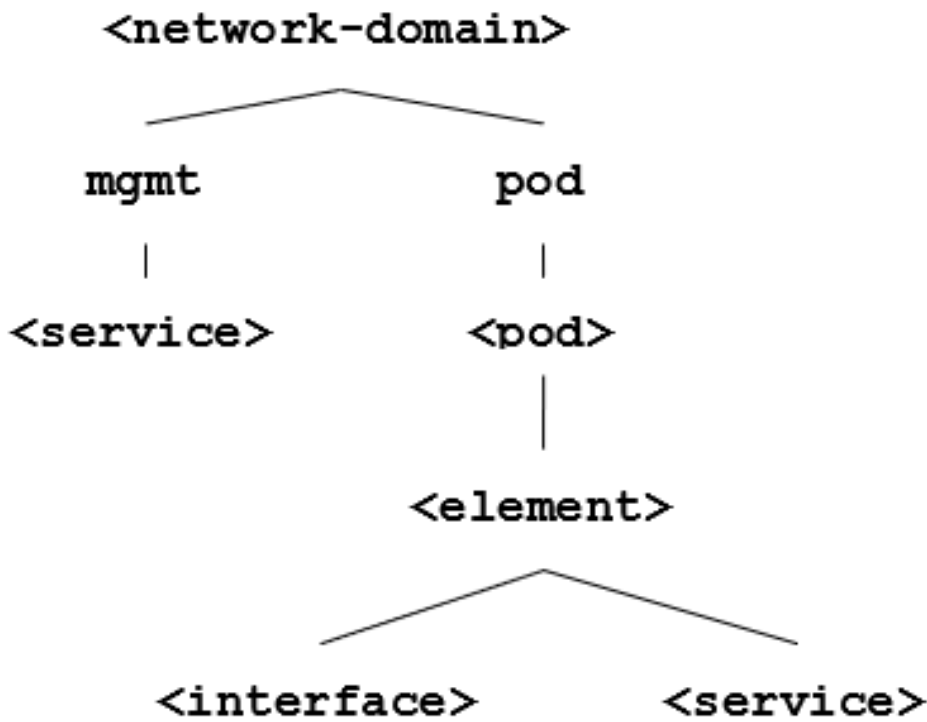


Figure 13. Sample DNS naming convention

The *<network-domain>* is the base domain name for the entire network (e.g. lab.rtbrick.net for the RtBrick lab environment). The next level of the DNS tree differentiates between central management services (mgmt) and pods (pod). Every management service has a designated DNS name. Every element in a pod has a designated DNS name too. Moreover, certain interfaces (for example, the management interface) or services deployed on an element might also get a DNS name assigned.

The network diagram below illustrates a lab topology that consists of two pod fabrics, Bangalore and Nuremberg, each formed by four switches and managed by a fabric daemon as well as a central management system. The central management system consists of three services: the control center to manage the network, the log management system to query log messages and process alerts, and the telemetry management system to process and visualize metrics. Each switch has a unique router ID, which is equal to the transport layer loopback IP address, a management interface and runs the control daemon (ctrlld) to manage the switch.

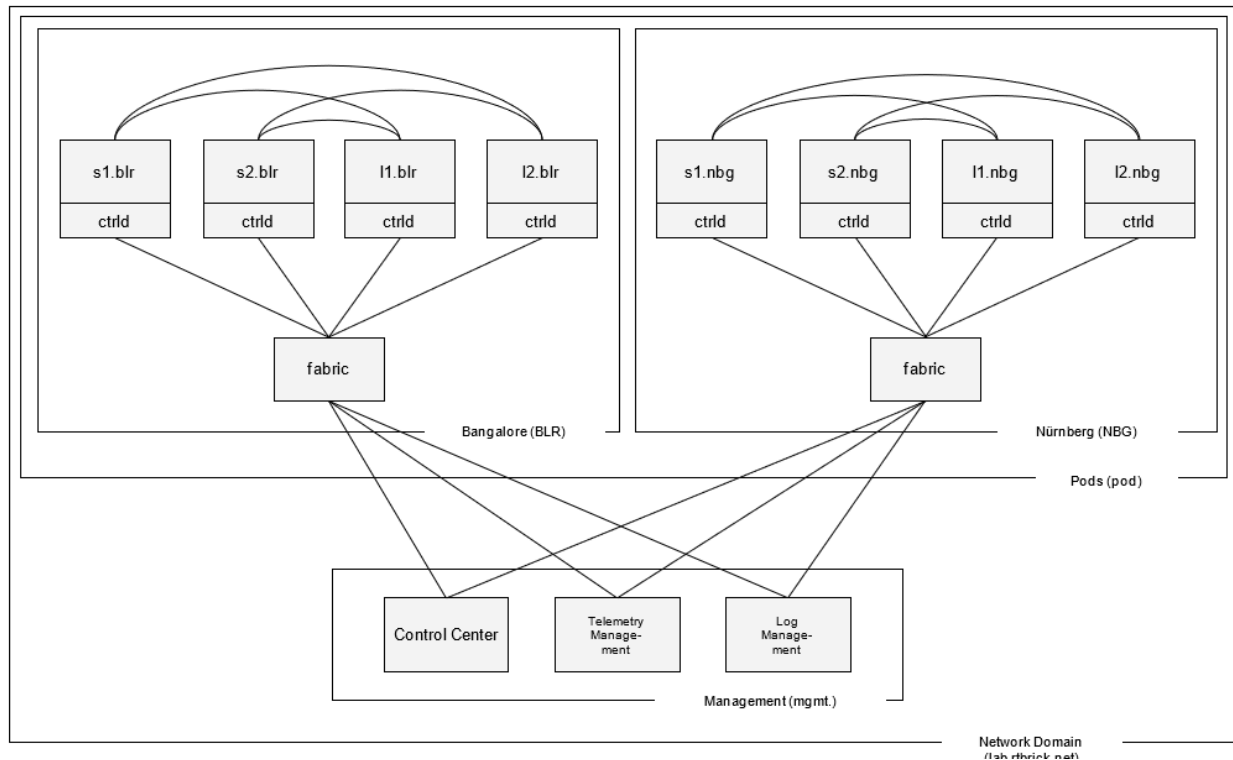


Figure 14. Sample network topology

The table below lists samples DNS names of elements and services in the sample network topology:

Sample DNS names

DNS Name	Description
lab.rtbrick.net	Network domain forming the top-level domain for all elements in the network
mgmt.lab.rtbrick.net	Subdomain for all management services in the network.
leitstand.lab.rtbrick.net	Control center service name.
log.lab.rtbrick.net	Log management service name.
telemetry.lab.rtbrick.net	Telemetry management service name.
pod.lab.rtbrick.net	Subdomain for all pods and their respective elements.
blr.pod.lab.rtbrick.net	Subdomain for all elements and services in pod Bangalore (BLR)
nbg.pod.lab.rtbrick.net	Subdomain for all elements and services in pod Nürnberg (NBG).

DNS Name	Description
fabric.blr.pod.lab.rtbrick.net	Name of fabric in pod Bangalore (BLR)
l1.blr.pod.lab.rtbrick.net	Name of leaf 1 in pod Bangalore (BLR). This name resolves to the transport layer loopback IP address.
me0.1.blr.pod.lab.rtbrick.net	Name of the management interface of leaf 1 in pod Bangalore (BLR). This name resolves to the IP address assigned to the management interface.
ctrlld.l1.blr.pod.lab.rtbrick.net	An alias to access the control daemon of leaf 1 in pod Bangalore. In the case of in-band management, the name is an alias of leaf1.blr.pod.lab.rtbrick.net whereas for out-of-band management the name is an alias for the management me0.leaf1.blr.pod.lab.rtbrick.net
All remaining leaf and spine switches have similar names. Below is another example for spine 2 located in pod Nürnberg (nbg)	
s2.nbg.pod.lab.rtbrick.net	Name of spine 2 in pod Nürnberg (NBG). This name resolves to the transport layer loopback IP address.
me0.s2.nbg.pod.lab.rtbrick.net	Name of the management interface of spine 2 in pod Nürnberg (NBG). This name resolves to the IP address assigned to the management interface.
ctrlld.s2.nbg.pod.lab.rtbrick.net	An alias to access the control daemon of spine 2 in pod Bangalore. In the case of in-band management, the name is an alias of spine2.nbg.pod.lab.rtbrick.net whereas for out-of-band management the name is an alias for the management me0.spine2.nbg.pod.lab.rtbrick.net

The Domain Naming Service subscribes the following domain events in order to maintain the DNS names according to this naming scheme:

Domain Events subscribed by DNS Naming Service in order to maintain DNS records

Domain Event	Action
ElementAddedEvent	Create DNS records for the added element.
ElementRenamedEvent	Update the DNS records for the renamed element.
ElementMovedEvent	Update the DNS records for the moved element, i.e. the element is now in a different pod.
ElementRetiredEvent / ElementRemovedEvent	Remove the DNS records of a retired or removed element. An element must be in the retired state before it can be removed from the inventory. A retired element is inactive and kept in the repository for documentation purposes only. It depends on whether the DNS record shall be part of the documentation, and whether the records are removed when an element is retired or gets removed from the inventory.
ElementIfIAddedEvent	Update the DNS record of an element if the management interface or transport layer loopback interface was added
ElementIfIModifiedEvent	Update the DNS record of an element if the management interface or transport layer loopback interface IP address has changed
ElementIfIRemovedEvent	Update the DNS record of an element if the management interface or transport layer loopback interface was removed
PodRenamedEvent	Update the DNS records for all elements in the pod.

Registered Address	Support	Sales
40268, Dolerita Avenue Fremont CA 94539		
+1-650-351-2251		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

©Copyright 2024 RtBrick, Inc. All rights reserved. The information contained herein is subject to change without notice. The trademarks, logos and service marks ("Marks") displayed in this documentation are the property of RtBrick in the United States and other countries. Use of the Marks are subject to RtBrick's Term of Use Policy, available at <https://www.rtbrick.com/privacy>. Use of marks belonging to other parties is for informational purposes only.