API References

Version 23.8.1, 12 September 2023

Table of Contents

1. RBFS API User Guide

1.1. RBFS REST APIs

This document contains all the information about RBFS REST API services, their purposes and how to use these APIs. This documentation goes hand-in-hand with the RBFS OpenAPI Specification document which provides information about all RBFS REST API endpoints. It is recommended to refer to this document in conjunction with the RBFS OpenAPI Specification document.

Introduction to RBFS REST APIs

RBFS REST APIs allow customers and partners to programmatically access information from the RBFS software components. RBFS REST APIs enable users to manage and automate many of their tasks by accessing and consuming the RBFS data simply and securely.

RBFS REST API architecture supports containerized deployments with a centralized configuration and management. It also enables the configuration and management of distinct daemons and services.

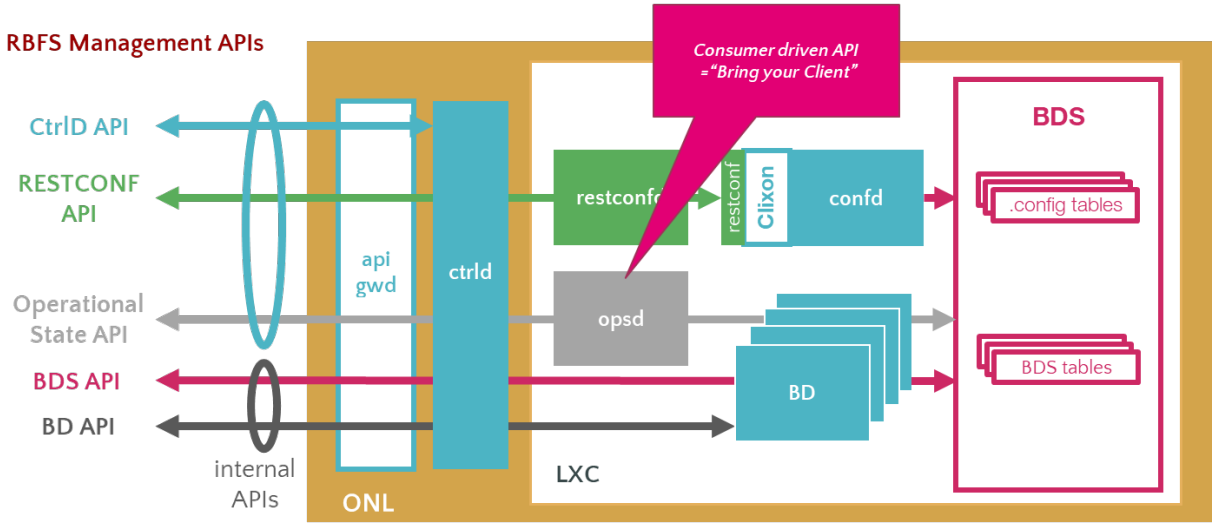
RBFS APIs adhere to the Representational State Transfer (REST) principles. RBFS APIs enable network administrators to securely connect to the RBFS device and execute remote procedure calls (rpc commands). RBFS REST APIs use JavaScript Object Notation (JSON) for the exchange of information. The OpenAPI Specification format, which is a broadly accepted industry standard for describing REST APIs, is used to describe, consume, and visualize RBFS REST APIs.

Understanding RBFS APIs

RBFS, a disaggregated Broadband Network Gateway, uses many underlying APIs to make all communications possible with various RBFS software components, the host operating system (ONL), and the hardware platform. RBFS consists of several independent microservices including the API Gateway daemon (ApiGwD) and Control daemon (CtrlD). Both of these microservices, known as daemons, play crucial roles in managing RBFS instances.

The architecture of the RBFS requires all API requests from external clients to be routed to the API Gateway. After successful authentication by the API Gateway,

these requests are forwarded to the Control daemon. CtrlD, which is aware of the state and port information of all daemons that reside of the RBFS container, can forward requests to the respective daemons.



The illustration presents how the RBFS REST APIs communicate with various underlying software components. RBFS microservices, which perform various functions, are containerized in an open Linux container. RBFS APIs are generally categorized into Public Management APIs and Internal APIs. Public Management APIs include CtrlD API, RESTCONF API, and Operational state API. These APIs are used by network administrators for managing and automating many of their network administration tasks.

RBFS does not expose internal APIs such brick daemon APIs and BDS APIs.

The illustration also presents the API Gateway daemon (ApiGwD) and Control daemon (CtrlD), which are deployed on the host operating system (open network Linux) along with the RBFS container. The API Gateway acts as an entry point that provides a secure channel for all REST APIs by authenticating all requests. After successful authentication by the API Gateway, CtrlD (Control Daemon) passes all the requests to the respective daemons (that reside in the RBFS container) which are responsible for performing certain tasks.

RBFS APIs

CtrlD API: CtrlD runs on the host OS (ONL) and acts as a proxy to the other APIs including high-level APIs for the RBFS configurations. The CtrlD API, implemented by the CtrlD, performs various tasks such as starting the container and rebooting

the device. In case of a software upgrade, this API is used to trigger the upgrade.

API Gateway: You can deploy the API Gateway Daemon (ApiGwD) on the host OS (that is ONL) to secure the RBFS management plane. The API Gateway authenticates all API requests using JSON web tokens. The API Gateway Daemon acts as the TLS endpoint for the hardware platform and it converts external access token into an internal RtBrick token /SEC/. Finally, it forwards the requests to the CtrlD.

The API Gateway also enforces an API throttler that provides a mechanism called API throttle quotas to protect the RBFS system resources from being exhausted with too many requests by a single client system that uses the RBFS APIs very extensively.

Operational State APIs: The Operational State API, provided by the Operational State Daemon (opsd), allows accessing system states such as routing protocol states, interface states, subscriber states and resource utilization. The Operational State Daemon takes care of examining the operational state of a switch and runs actions to diagnose and troubleshoot the problems. The operational state is ephemeral and state data is lost when the switch reboots.

RESTCONF APIs: With RESTCONF, you can manage all the configurations in RBFS.

Prometheus APIs: Prometheus is an open-source monitoring and alerting software that is containerized or packaged as a microservice along with other RBFS microservices in a Linux container. Prometheus APIs help to retrieve metrics from various RBFS components for visibility.

1.2. Functions and Use cases of RBFS REST APIs

CtrlD API

The CtrlD performs various functions in the lifecycle of an RBFS container. Zero-Touch Provisioning installs the RBFS software on the hardware devices with minimal human intervention. It is the responsibility of the CtrlD to initiate, discover and download the startup configuration files as part of Zero-Touch provisioning (ZTP) process. The CtrlD retrieves the base URL and executes the startup configuration on the device, that is pre-installed with ONL, the host operating system.

You can locate the CtrlD OpenAPI Specification at CTRLD API Reference.

RESTCONF API

RESTCONF is an HTTP-based REST API protocol for network management and automation. It provides a programmatic interface for accessing data defined in YANG. The YANG model describes the configuration syntax. In RBFS, the RESTCONF API provides the configuration data.

You can use RESTCONF API to execute various configurations in RBFS.

You can locate the RESTCONF OpenAPI Specification at RESTCONF API Reference

RFC and draft compliance are partial except as specified.

Operational State API

The Operational State API (opsd) provides the system state information. The ospd is backward compatible and supports running older and newer versions of applications together in the network. The Backward compatibility feature is useful whenever newer RBFS releases are rolled out.

The Operational State API was implemented using the Python language that provides a collaborative system to all stakeholders including integration partners, customers, professional services and engineering to collaborate on the API endpoints.

You can locate the Operational State OpenAPI Specification at Operational State API Reference.

Guidelines and Limitations

When you execute configurations through management APIs, and then with the Command Line Interface at the same time, it results in conflicts when you commit the configuration through the CLI. The reason is that CtrlD directly interacts with the backend applications and these changes are not synced with the CLI.

1.3. Configurations

This section describes the configuration files of the APIGWD and CTRLD.

APIGWD

In a production environment, the APIGWD binary starts with default parameters. This service of APIGWD is called rtbrick-apigwd.

To see the default parameters and the files where configurations are stored, the easiest way to figure out is to do a apigwd -help.

To see the actual installed APIGWD version we can use apigwd -version.

APIGWD has in essence 3 important configuration files:

- /etc/rtbrick/apigwd/config.json: Configuration for the apigwd
- /etc/rtbrick/apigwd/access_secret_jwks.json: JWKS file for external communication
- /etc/rtbrick/apigwd/tls.pem: X509 public/private key file in pem format

SSL certificate

Every call to the APIGWD is secured by TLS. If there is no TLS certificate provided, it is one generated and signed by a self-signed root CA.

To specify a TLS certificate, there are the following possibilities to achieve this:

- Provide the TLS file via ZTP
- Provide the TLS remote file URLs via the config.js:
  Therefore you can also specify a reload time. Every x seconds APIGWD tries to download a new TLS file. But for downloading, an RFC 7234-compliant cache is used.

JWKS File

The APIGWD validates the access token against a JSON Web Key Set (JWKS) (https://tools.ietf.org/html/rfc7517).

APIGWD allows to specify 2 sources for keyset, and for validation, the sets are consulted in the following order:

- A local file on the filesystem:
  This file can be provided via ZTP. It is recommended to provide one file, even if

it is an empty key set file. Otherwise, there is a preconfigured file on the system that will be used.

- Provide the JWKS remote file URLs via the config.js:
  Every time a token has to be verified the JWKS file will be downloaded, but for the download an RFC 7234-compliant cache is used.
- Provide the OpenIDConnect configuration via the config.js:
  Every time a token has to be verified the Issuer is consulted to get the link to the JWKS file, and the file will be downloaded, but for the download, an RFC 7234 compliant cache is used.

Caching

As stated above, for configuration file downloads an RFC 7234 compliant cache is used. The cache directives should be used wisely; otherwise, a lot of traffic could be generated.

config.json

This section describes the main configuration file of APIGWD. This file can be changed on the file system, APIGWD has a file watcher on the file and does a reload when the file changes.

/etc/rtbrick/apigwd/config.json example

```
{
  "access_token_jwks_urls": [
    "http://192.168.202.56:8080/primaryJWKS",
    "http://192.168.202.56:8080/secondaryJWKS"
  ],

  "request_rate": 5,
  "request_burst": 10,
  "report_rejects_every": 10
}
```

/etc/rtbrick/apigwd/config.json format

Name	Type	Description
access_token_jwks_urls	[]string	Allows to specify multipe jwks remote urls.
Remote PEM file		

pem_urls	[]string	Allows to specify multiple PEM remote URLs. Empty list disables the download.
pem_reload_time	int	Allows to specify the time after a new reload is triggered. 0 disables the download.
	Request rate limit	
request_rate	float	The allowed requests per second per client.
request_burst	int	Is the maximum number of tokens that can be consumed at once, without respect to the rate.
report_rejects_every	int	Report rejects only every x seconds to avoid massive logging to a GELF endpoint.

access_secret_jwks.json

This section describes the access_secret_jwks.json file. This file can be changed on the file system; APIGWD has a file watcher on the file and does a reload when the file changes.

JSON Web Key Set (JKWS) is described in the RFC 7517.

/etc/rtbrick/apigwd/access_secret_jwks.json example

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "access",
      "alg": "RS256",
      "n": "NOT A REAL KEY"
    }
  ]
}
```

These keys are for the authentication of external calls towards the APIGWD.

The right key is selected by the kid (key id). With this key, the access tokens are verified and converted to a RtBrick token.

tls.pem

This section describes the tls.pem file. This file contains the TLS certificate (public and private key) used to serve the TLS endpoint.

If the file is not specified, a new self-signed certificate is created.

This file can be changed on the file system, APIGWD has a file watcher on the file and does a reload when the file changes.

This file is an X509 public/private key file in PEM format specified in the RFC7468.

/etc/rtbrick/apigwd/tls.pem example

```
-----BEGIN CERTIFICATE-----
NOT A REAL KEY
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
NOT A REAL KEY
-----END RSA PRIVATE KEY-----
```

CTRLD

In a production environment, the CTRLD binary starts with default parameters. The CTRLD service is called rtbrick-ctrld.

To see the default parameters and the files where configurations are stored, the easiest way to figure out is to do a ctrld -help.

To see the actual installed CTRLD version, use ctrld -version.

CRTLD has three important configuration files:

- /etc/rtbrick/ctrld/config.json: Configuration for CTRLD
- /etc/rtbrick/ctrld/policy.json: Role-Based Access Control policy file.
- /var/lib/lxc/<container-name>/element.config: Element configuration file per container.

config.json

This section describes the main configuration file of CTRLD. This file can be changed via API; if it is changed on the file system, CTRLD has to be restarted.

/etc/rtbrick/ctrld/config.json example

```
{
  "element_name": "element_name",
  "pod_name": "pod_name",
  "rbms_enable": true,
  "rbms_host": "http://198.51.100.48",
```

```
  "rbms_authorization_header": "Bearer THIS IS NOT A REAL KEY",
  "rbms_heartbeat_interval": 10,
   "logging": {
    "heartbeat_interval": 60,
    "aliases": {
      "default": {
        "endpoints": [
          {
            "type": "gelf",
            "max_log_level": 5,
            "network": "http",
            "address": "http://10.200.32.49:12201/gelf"
          },
          {
            "type": "syslog",
            "max_log_level": 5,
            "network": "udp",
            "address": "10.200.32.49:516"
          }
        ]
      },
      "ztp": {
        "endpoints": [
          {
            "type": "gelf",
            "max_log_level": 4,
            "network": "http",
            "address": "http://10.200.32.49:12201/gelf"
          }
        ]
      }
    }
  },
  "auth_enabled": false
}
```

/etc/rtbrick/ctrld/config.json format

Name	Type	Description
element_name	string	The element name of the host personality of the switch.
pod_name	string	The pod name of the host personality of the switch.
rbms_enable	bool	To enable all RBMS outgoing messages rbms_host
rbms_host	string	RBMS base URL e.g.: http://198.51.100.144:9009
rbms_authorization_header	string	RBMS Authorization Header is set to all calls which are outgoing to RBMS
rbms_heartbeat_interval	int	RBMS heartbeat Interval in seconds (0 means deactivated)

<table>
<tr><td>auth_enabled</td><td>bool</td><td>To Enable the authorization and authentication</td></tr>
<tr><td>logging</td><td colspan="2">Log configuration for the host personality of the switch. The routing instances (elements) can configure the logging in the RBFS configuration, and that is forwarded per routing instance to CTRLD. The alias default, acts as the default alias if the specific one is not set.

<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>alias</td><td>string</td><td>Logical name of the endpoints e.g.: ztp for ztp messages.</td></tr>
</table>

Each alias can have multiple endpoints. Consider the following constellations.

If an alias does not define any endpoint, the alias is disabled, that means the message is not sent and will also not be sent to the default alias. This allows disabling the default alias, which means no message is sent as long as no specific alias gets defined.

<table>
<tr><td>Name</td><td>Type</td><td>Description</td></tr>
<tr><td>type</td><td>string</td><td>Type could be syslog or gelf.</td></tr>
<tr><td>max_log_level</td><td>string</td><td>MaxLogLevel that will be forwarded (default "Notice: 5)"</td></tr>
<tr><td>network</td><td>string</td><td>Network get network either tcp, udp or http. Consider the support matrix: * gelf: http * syslog: udp, tcp</td></tr>
<tr><td>address</td><td>string</td><td>Address where to send the message</td></tr>
<tr><td>formatter</td><td>string</td><td>The formatter that should be used. Consider the support matrix:

• gelf: none
• syslog: rfc5424</td></tr>
</table>
</td></tr>
</table>

policy.json

This section shows the **r**ole **b**ased **a**ccess **c**ontrol (RBAC) configuration for CTRLD. This file can be changed via API; if it is changed on the file system CTRLD has to be restarted.

/etc/rtbrick/ctrld/policy.json example

```
{
  "permissions": [
    {"sub": "system", "obj": "/*", "act": ".*" },
    {"sub": "supervisor", "obj": "/*", "act": ".*" },
    {"sub": "operator", "obj": "/*", "act": ".*"},
    {"sub": "reader", "obj": "/*", "act": "GET"},
    {"sub": "reader", "obj":
"/api/v1/rbfs/elements/{element_name}/services/{service_name}/proxy/bds/table/walk
", "act": ".*"},
    {"sub": "reader", "obj":
"/api/v1/rbfs/elements/{element_name}/services/{service_name}/proxy/bds/object/get
", "act": ".*"}
  ]
}
```

/etc/rtbrick/ctrld/policy.json format

Name	Type	Description
sub	string	Subjects means the role which has the permission. Here RegexMatch Function is used: a regular expression pattern matcher.
obj	string	Object is the REST endpoint. Here KeyMatch4 Function is used: KeyMatch4 determines whether key1 matches the pattern of key2 (similar to RESTful path), key2 can contain a * and other patterns: • "/foo/bar" matches "/foo/*" • "/resource1" matches "/{resource}" • "/parent/123/child/123" matches "/parent/{id}/child/{id}" • "/parent/123/child/456" does not match "/parent/{id}/child/{id}"

Name	Type	Description
act	string	And Action is the HTTP Method. Here RegexMatch Function is used: a regular expression pattern matcher.

So the example rules mean:

- The user with the role system is allowed to access all rest endpoints and act on them with all HTTP methods.
- The user with the role reader can access all rest endpoints but can only call the HTTP GET method.
- All authenticated users are allowed to access the proxy endpoint with all HTTP methods. The user with the role system can access all rest endpoints and act on them with all HTTP methods.

element.config

This section shows the element.config, which can be created per container. This file allows redefining the element name so that the name can defer from the container name.

/var/lib/lxc/<container-name>/element.config example

```
```

/var/lib/lxc/<container-name>/element.config format

Name	Type	Description
element_name	string	Name of the Element (Default container name)
pod_name	string	Name of the POD
ztp_enabled	bool	If enabled, the ZTP post process starts after the switch changes to an operational state "up". It's recommended to set this to false. In that case, only the initial installation or reinstallation will trigger that process.

1.4. Business Events

APIGWD and CTRLD send different GELF messages about status changes or

progress of processes to a GELF endpoint.

GELF message format

Name	Type	Mandatory	Description
Default Message Fields			
version	String	Yes	The GELF message format version. Default value: 1.1
host	String	Yes	The hostname is assigned via DHCP to the management interface. Defaults to the management IP address if no hostname is assigned.
level	int	Yes	Message Severity. See Table-1.
timestamp	float	Yes	Unix epoch time in seconds with an optional fraction of milliseconds.
short_message	String	Yes	Problem message.
full_message	String	No	Detailed problem description.
_daemon	String	Yes	Name of the daemon.
_log_module	String	Yes	The module name identifies the component that created the log record. It allows segregating log records into different streams. Each stream can apply different processing rules and also be processed by different organizational units of the network operator.

Name	Type	Mandatory	Description
_log_event	String	Yes	The log event identifies the log message template in the log configuration. The log event simplifies finding where in the system the log record was created. The log event should be succinct and typically conveys a unique reason code. In addition, the log event should be a reference that can be looked up in the product troubleshooting guide.
_serial_number	String	Yes	The serial number of the switch. This allows tracking hardware replacements, even if the element name remains the same. Empty if not available.
_rtb_image_version	String	No	ONL Image Version that is installed on the switch that reports this message.
_origin	String	No	host or container, defines the origin of a message. This is only set for events that are ambiguous.
ZTP Message Fields			
_config_name	String	No	Exposes the loaded configuration name. Only set when a configuration file was processed or an attempt to process the file failed (e.g., 404 Not Found response from the HTTP server while attempting to load the configuration)
_config_sha1	String	No	Exposes the SHA1 checksum of the loaded configuration. Only set when the HTTP server returns a configuration.

Name	Type	Mandatory	Description
_operational_state	String	No	Exposes the operational state of the element.
Request Message Fields			
_rid	String	No	Request ID, either X-Request-ID or new generated
_user_name	String	No	User name out of the access token
_user_subject	String	No	User subject out of the access token
_received_time	String	No	Time when the requested arrived
_method	String	No	HTTP method
_url	String	No	HTTP url
proto	String	No	HTTP protocol
_remote_ip	String	No	HTTP remote ip address
Service State Message Fields			
_service_name	String	No	Service name
_service_operational_state	String	No	Operational Service
_service_startup_time	Number	No	Service startup time in unix epoch time, the number of seconds elapsed since January 1, 1970 UTC.
_service_down_flap_time	Number	No	Last down flap time in unix epoch time, the number of seconds elapsed since January 1, 1970 UTC.
_service_down_flap_counter	Number	No	Last down flap time in unix epoch time, the number of seconds elapsed since January 1, 1970 UTC.
_service_restarted	String	No	Restart is set to true if service_startup_time was changed.

Level Descriptions as in RFC 5424

Level	Name	Comment
0	**Emergency**	System is unusable

Level	Name	Comment
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

GELF sample message

```
{
  "_config_name": "ctrld",
  "_config_sha1": "f1e06ef1e53becde6f8baf2b2fafe7dc9c36f6f0",
  "_daemon": "ctrld",
  "_element_name": "leaf01",
  "_log_event": "ZTP0011I",
  "_log_module": "ztp",
  "_serial_number": "591654XK1902037",
  "host": "leaf01",
  "level": 6,
  "short_message": "ztp ctrld config set",
  "timestamp": 1588382356.000511,
  "version": "1.1"
}
```

Event Types

Instance	severity	log_module	log_event	description
ztp	Notice	ztp	ZTP0011I	ztp ctrld config set
ztp	Warn	ztp	ZTP0012W	ztp ctrld config not provided
ztp	Alert	ztp	ZTP0013E	ztp ctrld config not set
ztp	Notice	ztp	ZTP0021I	ztp startup config set
ztp	Warn	ztp	ZTP0022W	ztp startup config not provided
ztp	Alert	ztp	ZTP0023E	ztp startup config not set
ztp	Notice	ztp	ZTP0041I	ztp ctrld rbac config set
ztp	Warn	ztp	ZTP0042W	ztp ctrld rbac config not provided
ztp	Alert	ztp	ZTP0043E	ztp ctrld rbac config not set

Instance	severity	log_module	log_event	description
ztp	Notice	ztp	ZTP0051I	ztp tls config set
ztp	Warn	ztp	ZTP0052W	ztp tls config not provided
ztp	Alert	ztp	ZTP0053E	ztp tls config not set
ztp	Notice	ztp	ZTP0061I	ztp accessjwks config set
ztp	Warn	ztp	ZTP0062W	ztp accessjwks config not provided
ztp	Alert	ztp	ZTP0063E	ztp accessjwks config not set
ztp	Notice	ztp	ZTP0071I	ztp apigwd config set
ztp	Warn	ztp	ZTP0072W	ztp apigwd config not provided
ztp	Alert	ztp	ZTP0073E	ztp apigwd config not set
ztp	Notice	ztp	ZTP1000I	ztp process finished
security	Warn	security	SEC0001W	access forbidden
security	Warn	security	SEC0002W	access invalid rtb token
security	Warn	security	SEC0003W	access invalid access token
security	Warn	security	SEC0004W	not able to download remote keys
security	Warn	security	SEC0005W	not able to download remote pem
security	Warn	security	SEC0006W	request rate limited (this message is also rate limited, and can be controlled in the apiwd config)
element	Notice	element	HTB0001	heartbeat with the operational_state
element	Notice	element	STA0001	element state change
element	Notice	element	STA0021	service up
element	Error	element	STA0022	service unexpected down
element	Notice	element	STA0023	service expected down
element	Notice	element	STA0003	ready for service
element	Notice	element	STA0031	module new (one of the modules is newly discovered e.g. fan, SFP ..., this event will be fired after every reboot of ctrld)

Instance	severity	log_module	log_event	description
element	Notice	element	STA0032	module changed (one of the modules got changed e.g. fan, SFP ...)

1.5. Related Documentation

/ONIE/	The ONIE documentation outlines the DHCP options supported for image discovery. https://opencomputeproject.github.io/onie/design-spec/discovery.html
/ZTP/	The Zero-Touch Provisioning Guide outlines the current configuration discovery process. https://documents.rtbrick.com/current/ztp/ztp_guide_online.html
/SEC/	The Secure the Management Plane guide gives a detailed insight on this topic. https://documents.rtbrick.com/current/secmgmt/secmgmt_guide_online.html
/RADIUS/	The RADIUS attribute matrix provides an overview of the supported RADIUS attributes including a reference to the RFC that defines the message attribute. To obtain this document, contact your customer support team.
/CTRLD/	The CTRLD API describes all CTRLD REST API endpoints in detail. https://documents.rtbrick.com/index_api.html
/RESTCONF/	The RESTCONF API reference describes all configuration API endpoints. https://documents.rtbrick.com/index_api.html
/GELF/	The Graylog Extended Log Format (GELF) is a log format, this document outlines the fundamentals. To obtain this document, contact your customer support team.

2. RBFS APIs

```
    <link rel="stylesheet" type="text/css" href="./_attachments/rtbrick.css">
    <div id="swagger-ui"></div>
    <script src="./_attachments/swagger-ui-bundle.js"></script>
    <script src="./_attachments/swagger-ui-standalone-preset.js"></script>
    <script>
    window.onload = function () {
      const DisableTryItOutPlugin = function() {
        return {
          statePlugins: {
            spec: {
              wrapSelectors: {
                allowTryItOutFor: () => () => false
              }
            }
          }
        }
      }

      // Begin Swagger UI call region
      const ui = SwaggerUIBundle({
        urls: [
          { "url": "./_attachments/rbfs/swagger_ctrld.yaml", "name": "CTRLD API
Reference" },
          { "url": "./_attachments/rbfs/rtbrick-config_restconf_swagger.json",
"name": "RESTCONF API Reference" },
          { "url": "./_attachments/rbfs/swagger_opsd.yaml", "name": "Operational
State API Reference" },
          { "url": "./_attachments/rbfs/swagger_bds.yaml", "name": "BDS API
Reference" },
        ],
        dom_id: '#swagger-ui',
        deepLinking: true,
        docExpansion:"none",
        presets: [
          SwaggerUIBundle.presets.apis,
          SwaggerUIStandalonePreset
        ],
        plugins: [
          SwaggerUIBundle.plugins.DownloadUrl,
          DisableTryItOutPlugin
        ],
        layout: "StandaloneLayout"
      })
      // End Swagger UI call region

      window.ui = ui
    }
    </script>
```

3. RBMS APIs

```
  <link rel="stylesheet" type="text/css" href="./_attachments/rtbrick.css">

  <div id="swagger-ui"></div>

  <script src="./_attachments/swagger-ui-bundle.js"> </script>
  <script src="./_attachments/swagger-ui-standalone-preset.js"> </script>
  <script>
    window.onload = function () {
      const DisableTryItOutPlugin = function() {
        return {
          statePlugins: {
            spec: {
              wrapSelectors: {
                allowTryItOutFor: () => () => false
              }
            }
          }
        }
      }

      // Begin Swagger UI call region
      const ui = SwaggerUIBundle({
        urls: [
          { "url": "./_attachments/rbms/swagger_ztp_mgmt.yaml", "name": "ZTP
Management Server API Reference" },
          { "url": "./_attachments/rbms/swagger_leitstand_template_engine.yaml",
"name": "Template Engine API Reference" },
          { "url": "./_attachments/rbms/metric.yaml", "name": "Resource Inventory
Metric API" },
          { "url": "./_attachments/rbms/meta.yaml", "name": "Resource Matadata API"
},
          { "url": "./_attachments/rbms/jobs.yaml", "name": "Job Management API" },
          { "url": "./_attachments/rbms/image.yaml", "name": "Resource Inventory
Image API" },
          { "url": "./_attachments/rbms/group.yaml", "name": "Resource Inventory
Element Group API" },
          { "url": "./_attachments/rbms/facility.yaml", "name": "Resource Inventory
Rack and Facility API" },
          { "url": "./_attachments/rbms/element.yaml", "name": "Resource Inventory
Element API" },
          { "url": "./_attachments/rbms/dns.yaml", "name": "DNS Zone API" },
          { "url": "./_attachments/rbms/commons.yaml", "name": "commons" },
        ],
        dom_id: '#swagger-ui',
        deepLinking: true,
        docExpansion:"none",
        presets: [
          SwaggerUIBundle.presets.apis,
          SwaggerUIStandalonePreset
        ],
        plugins: [
          SwaggerUIBundle.plugins.DownloadUrl,
          DisableTryItOutPlugin
        ],
        layout: "StandaloneLayout"
      })
      // End Swagger UI call region
```

```
    window.ui = ui
  }
</script>
```

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