BDS Overview Guide

Version 23.8.1.2, 06 November 2023

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

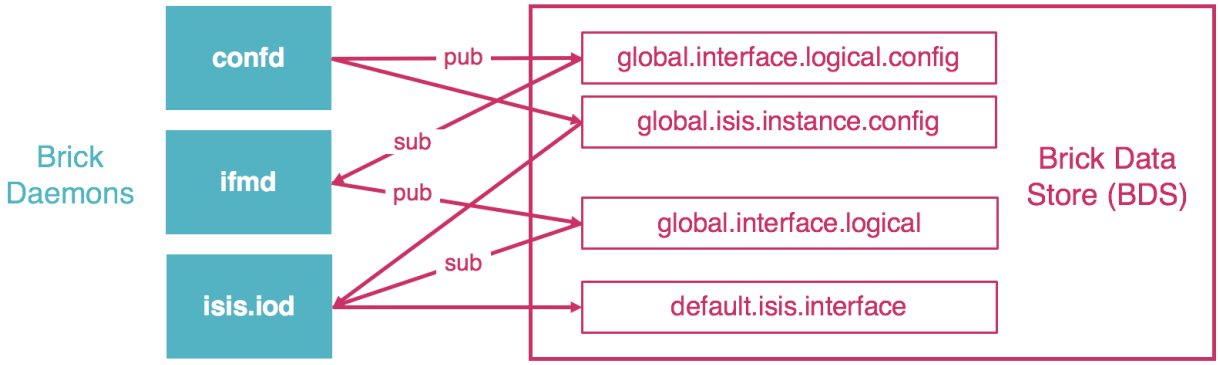
Table of Contents

1. Overview

The Brick Data Store (BDS) is a purpose-built, in-memory state database optimized for cloud networking. In RBFS, all system state information is stored as objects in BDS tables. Objects are entries in BDS tables that represent a state.

1.1. Pub/Sub Model

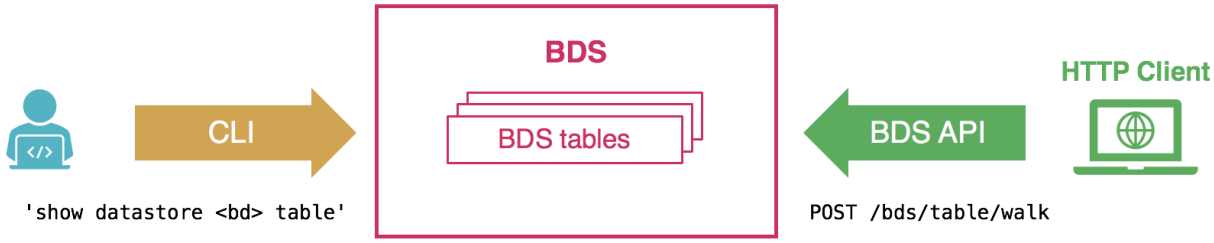
All Brick Daemons (BD) independently publish and subscribe to tables in a pub/sub model. This model provides resilience and scalability. The figure illustrates the concept:



In this example, the configuration daemon (confd) publishes tables that contain configuration data for logical interfaces or IS-IS instances. The interface management daemon (ifmd) is responsible for creating and maintaining interfaces. It therefore subscribes for example to the logical interface configuration table. After processing the data, it creates the logical interfaces and publishes them in the logical interface table. The IS-IS input/output daemon (isis.iod) subscribes to the logical interface table as well as the IS-IS configuration tables. It in turn creates and runs interfaces on which IS-IS protocol packets are exchanged, and publishes them in the IS-IS interface table.

1.2. BDS User Interface

All BDS tables and objects are fully accessible to the RBS user both via CLI and an API. This provides unprecedented visibility into the system state. This guide covers the BDS CLI. For the BDS API, refer to the BDS API Reference.



Please note the RBFS CLI supports show commands to verify the configuration and

operation of the system for all features. Therefore you usually do not need to inspect BDS tables directly. For example, for verifying the status of the logical interfaces, you can simply use the 'show interface summary' or the 'show interface logical' commands, instead of displaying the logical interface table. The BDS CLI and API to inspect BDS tables are rather available in addition to advanced analysis or troubleshooting.

1.3. 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.

2. BDS Operational Commands

This section summarizes some useful BDS CLI commands. It assumes you have some basic knowledge of BDS, and are familiar with the respective tables you are looking for. Describing all tables involved in a particular feature or functionality is out of the scope of this guide.

2.1. BDS Summary

The BDS summary command provides some metadata of the BDS tables.

Syntax:

show datastore <bd-name> **summary** <option>

Option	Description
<bd-name>	Name of the brick daemon to request this information from. As all BDs independently publish and subscribe to BDS tables, they all hold a different set of tables. As a best practice, select the BD that owns the respective table you are looking for.
table <table-name>	Display metadata for the given table.

Example:

```
supervisor@rtbrick>LEAF01: op> show datastore ribd summary
Brick Datastore Summary:
Table Name: local.bds.table.registry.ribd
  Index                                          Type                 Active
Obj Memory   Index Memory
  sequence-index                                 bds_rtb_bplus           234
24.38 KB     9.59 KB
  gc-index                                       bds_rtb_bplus             0   0
bytes      0 bytes
  table-name-index                               bplus                   234
24.38 KB     9.59 KB
Table Name: local.trim.qrunner.table
  Index                                          Type                 Active
Obj Memory   Index Memory
  sequence-index                                 bds_rtb_bplus             7
840 bytes    728 bytes
  gc-index                                       bds_rtb_bplus             0   0
bytes      0 bytes
  immutable_index                                bplus                     7
840 bytes    728 bytes
  qrunner-index                                  qrunner                   7
840 bytes    728 bytes
Table Name: local.bds.statistics
  Index                                          Type                 Active
Obj Memory   Index Memory
  sequence-index                                 bds_rtb_bplus           319
34.84 KB     14.36 KB
  gc-index                                       bds_rtb_bplus             0   0
bytes      0 bytes
  immutable-index                                bplus                   319
34.84 KB     14.36 KB
Table Name: local.bds.module.registry
  Index                                          Type                 Active
Obj Memory   Index Memory
  sequence-index                                 bds_rtb_bplus            68
3.53 KB      2.79 KB
  gc-index                                       bds_rtb_bplus             0   0
bytes      0 bytes
  module-name-index                              bplus                    68
3.53 KB      2.79 KB
<...>
```

2.2. BDS Tables

You can use the BDS table commands to display the table objects that contain the actual state information.

Syntax:

show datastore <bd-name> **table** <option>

Option	Description
<bd-name>	Name of the brick daemon to request this information from. As all BDs independently publish and subscribe to BDS tables, they all hold a different set of tables. As a best practice, select the BD that owns the respective table you are looking for.
<table-name>	Name of the BDS table to display. Without further options, this command displays all objects in a table format.
<table-name> json	Display the complete table data in JSON format.
<table-name> attribute <attribute-name> <attribute-value> exact	Filter the table objects based on attribute name and value. You can filter on any attribute, except for attributes of type array. The filter performs a regex match. You can therefore specify the attribute value as a regular expression (regex). You can use the exact match along with the (default) regular expression match.
<table-name> summary	Display metadata for the given table.
properties	Display owner, published/subscribed, and locality information for all tables known by the given daemon.

Example 1: Logical Interface Table

```
supervisor@rtbrick>LEAF01: op> show datastore ifmd table global.interface.logical
Object: 0, Sequence 100125, Last update: Mon Apr 03 13:49:39 GMT +0000 2023
  Attribute                               Type                           Length
Value
  logical_unit_id (1)                     uint16 (3)                          2
0
  ifl_name (2)                            string (9)                         13
ifl-0/1/31/0
  ifp_name (3)                            string (9)                         11
ifp-0/1/31
  instance (5)                            string (9)                          8
default
  mac_address (8)                         macaddr (22)                        6
e8:c5:7a:8f:76:f2
  ipv4_status (10)                        uint8 (2)                           1
up
  ipv6_status (12)                        uint8 (2)                           1
up
  mpls_mtu (13)                           uint16 (3)                          2
1500
  mpls_status (14)                        uint8 (2)                           1
up
  iso_mtu (15)                            uint16 (3)                          2
1500
  iso_status (16)                         uint8 (2)                           1
down
  admin_status (17)                       uint8 (2)                           1
up
  link_status (18)                        uint8 (2)                           1
up
  ifl_type (19)                           uint8 (2)                           1
Logical Sub interface
  operational_status (24)                 uint8 (2)                           1
up
  ifindex (25)                            uint32 (4)                          4
63745
  instance_id (27)                        uint32 (4)                          4
0
<...>
```

Example 2: Filter IPv6 Route Table by Prefix

```
supervisor@rtbrick>LEAF01: op> show datastore ribd table default.ribd.1.fib-
local.ipv6.unicast
 attribute prefix6 2001:db8::1/128
Object: 0, Sequence 1900002, Last update: Mon Apr 03 13:49:39 GMT +0000 2023
  Attribute                               Type                          Length
Value
  prefix6 (4)                             ipv6prefix (16)                   17
2001:db8::1/128
  source (11)                             uint8 (2)                          1
direct
  sub_src (12)                            uint8 (2)                          1
Host
  nexthop_key (25)                        payload (8)                       24
3c86eaebe6617cf61ed96c819cfd63839bd90cc85a533067
  preference (40)                         uint32 (4)                         4
0
  bcm_status (52)                         uint8 (2)                          1
None
  return_code (53)                        uint32 (4)                         4
0
  vpp_status (54)                         uint8 (2)                          1
None
  route_status (55)                       uint32 (4)                         4
|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-
```

Example 3: Filter IPv6 Route Table with Exact Match

```
supervisor@rtbrick>LEAF01: op> show datastore bgp.appd.1 table ip2vrf.bgp.rib-
in.ipv4.unicast.198.51.100.30.198.51.100.25 attribute prefix4 198.51.100.11/24
exact
Object: 0, Sequence: 367772, Last update: Wed May 19 08:05:08 GMT +0000 2021
  Attribute                               Type                          Length
Value
  status (1)                              uint8 (2)                          1
Valid
  recv_path_id (2)                        uint32 (4)                         4
0
  prefix4 (3)                             ipv4prefix (13)                    5
198.51.100.40/24
  rd (5)                                  route-distinguisher (40)           8
198.51.100.100:65001
  source (6)                              uint8 (2)                          1
bgp
  sub_src (7)                             uint8 (2)                          1
Local-Peer
  as_path (9)                             array (7), uint32 (4)             20
[57381, 42708, 1299, 5511, 3215]
  origin (10)                             uint8 (2)                          1
IGP
  peer_type (12)                          uint8 (2)                          1
2
  igp_metric (13)                         uint32 (4)                         4
4294967295
  send_path_id (18)                       uint32 (4)                         4
3238151775
  bgp_nh4 (19)                            ipv4addr (12)                      4
198.51.100.30
  community (24)                          array (7), community (27)          8
['1299:20000', '42708:200']
```

2.3. BDS Schema

The Brick Data Store is schema-driven. Table and object schema definitions are located in RBFS in /usr/share/rtbrick/libbds/. Instead of inspecting schema files, you can use the BDS schema commands to view the schemata directly in the CLI.

Syntax:

show datastore <bd-name> **schema** <option>

Option	Description
<bd-name>	Name of the brick daemon to request this information from. As all BDs independently publish and subscribe to BDS tables, they all hold a different set of tables. To view a table or object schema, you can select any BDs that know the respective table.
table-name <table-name>	Display the schema of the given table.
object object-name <object-name>	Display the schema of the given object.

Option	Description
object table-name <table-name>	Display the schema of the object for a given table. This option is useful if you do not know the name of the object but the name of the table in which it is used.

2.4. BDS Statistics Memory

The BDS statistics memory command provides detailed memory usage information.

Syntax:

show datastore <bd-name> **statistics memory**

Option	Description
<bd-name>	Name of the brick daemon of which to display the memory usage information.