BDS Overview Guide

Version 21.6.1, 25 June 2021

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

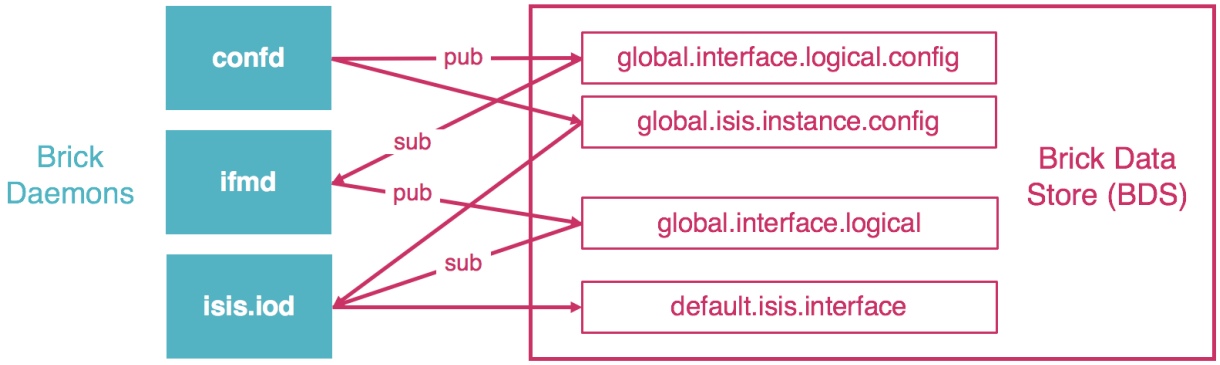
Table of Contents

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

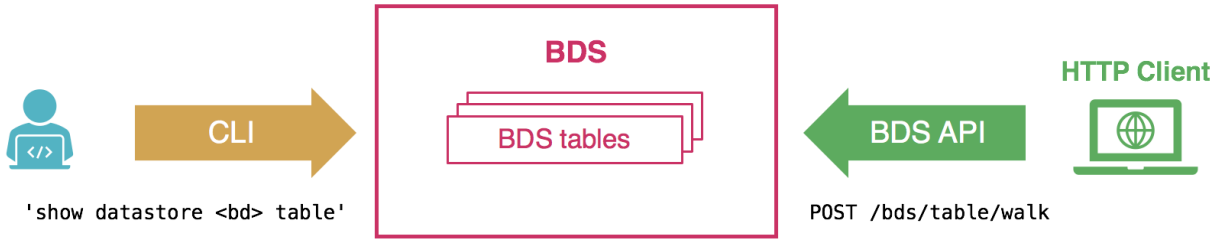
All Brick Daemons (BD) independently publish and subscribe to tables in a pub/sub model. This model provides resilience and scalability. The figure illustrates the concept:



In this example, the configuration daemon (confd) publishes tables which 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 an unprecedented visibility into the system state. This guide covers the BDS CLI. For the BDS API, please 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 for advanced analysis or troubleshooting.

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: op> show datastore ribd summary
Brick Datastore Summary:
Table Name: local.bds.table.registry.ribd
  Index                                  Type                 Active   Obj
Memory   Index Memory
  sequence-index                         bplus                   147
13.68 KB     7.2 KB
  gc-index                               libtrees-bplus            0   0
bytes      0 bytes
  table-name-index                       bplus                   147
13.68 KB     7.2 KB
Table Name: local.trim.qrunner.table
  Index                                  Type                 Active   Obj
Memory   Index Memory
  sequence-index                         bplus                     1   108
bytes    112 bytes
  gc-index                               libtrees-bplus            0   0
bytes      0 bytes
  immutable_index                        bplus                     1   108
bytes    112 bytes
  qrunner-index                          qrunner                   1   108
bytes    112 bytes
Table Name: local.bds.memory.stats
  Index                                  Type                 Active   Obj
Memory   Index Memory
  sequence-index                         bplus                   146
10.09 KB     9.93 KB
  gc-index                               libtrees-bplus            0   0
bytes      0 bytes
  mem_stats_index                        bplus                   146
10.09 KB     9.93 KB
  mem_bytes_index                        libtrees-bplus          146
10.09 KB     9.93 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 commands 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 of 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 Tablee

```
supervisor@rtbrick: op> show datastore ifmd table global.interface.logical
Object: 0, Sequence: 1, Last update: Fri Jan 29 08:12:20 GMT +0000 2021
  Attribute                                Type
Length     Value
  logical_unit_id (1)                      uint16 (3)
2     1
  ifl_name (2)                             string (9)
11     lo-0/0/0/1
  ifp_name (3)                             string (9)
9     lo-0/0/0
  instance (5)                             string (9)
8     default
  mac_address (8)                          macaddr (22)
6     7a:25:1d:a0:00:00
  ipv4_mtu (9)                             uint16 (3)
2     1500
  ipv4_status (10)                         uint8 (2)
1     up
  ipv6_mtu (11)                            uint16 (3)
```

```
2     1500
  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     Loopback interface
  operational_status (24)                   uint8 (2)
1     up
  ifindex (25)                              uint32 (4)
4     5
  instance_id (27)                          uint32 (4)
4     0
Object: 1, Sequence: 2, Last update: Fri Jan 29 08:12:22 GMT +0000 2021
  Attribute                                 Type
Length     Value
  logical_unit_id (1)                       uint16 (3)
2     1
  ifl_name (2)                              string (9)
14     memif-0/1/1/1
  ifp_name (3)                              string (9)
12     memif-0/1/1
  instance (5)                              string (9)
8     default
  mac_address (8)                           macaddr (22)
6     7a:25:1d:60:01:01
  ipv4_mtu (9)                              uint16 (3)
2     1500
  ipv4_status (10)                          uint8 (2)
1     up
  ipv6_mtu (11)                             uint16 (3)
2     1500
  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     2307
  instance_id (27)                        uint32 (4)
4     0
<...>
```

Example 2: Filter IPv6 Route Table by Prefix

```
supervisor@rtbrick: op> show datastore ribd table default.ribd.1.fib-
local.ipv6.unicast attribute prefix6 ::2/128
Object: 0, Sequence: 15, Last update: Fri Feb 05 10:21:00 GMT +0000 2021
  Attribute                               Type
Length     Value
  prefix6 (4)                             ipv6prefix (16)
17     fd3d:3d:0:99::2/128
  nexthop_key (5)                         payload (8)
24     a432c38c549ad26b15554f964d822134becc07db933dba54
  source (7)                              uint8 (2)
1     isis
  igp_metric (19)                         uint32 (4)
4     60
  preference (22)                         uint32 (4)
4     15
  isis_lsp_id (23)                        iso-lspid (25)
8     1000.9900.0002.00-00
  external (24)                           boolean (6)
1     False
  readvertised (25)                       boolean (6)
1     False
  up_down (26)                            boolean (6)
1     False
  sid_index (35)                          uint32 (4)
4     126
  sub_type (39)                           uint8 (2)
1     level-1
  sub_src (40)                            uint8 (2)
1     Local-Peer
  rt_type (43)                            uint8 (2)
1     Ip Reachability
  sid_flag (77)                           uint8 (2)
1     64
```

Example 3: Filter IPv6 Route Table with Exact Match

```
supervisor@rtbrick: op> show datastore bgp.appd.1 table ip2vrf.bgp.rib-
in.ipv4.unicast.20.1.1.2.20.1.1.1 attribute prefix4 2.0.0.0/16 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     2.0.0.0/16
  rd (5)                                   route-distinguisher (40)
8     192.1.4.1:65001
  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     20.1.1.2
  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 knows the respective table.
table-name <table-name>	Display the schema of the given table.

Option	Description
object object-name <object-name>	Display the schema of the given object.
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 Memory Statistics

The BDS memory statistics command provides detailed memory usage information.

Syntax:

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

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