Red Hat Data Grid 8.1

Data Grid Command Line Interface

Access data and manage Data Grid with the CLI
Access data and manage Data Grid with the CLI
Abstract

Connect to Data Grid servers via the command line interface (CLI) to access data and perform management operations.
<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 1. RED HAT DATA GRID .............................................................. 7</td>
</tr>
<tr>
<td>1.1. DATA GRID DOCUMENTATION ......................................................... 7</td>
</tr>
<tr>
<td>1.2. DATA GRID DOWNLOADS ................................................................. 7</td>
</tr>
<tr>
<td>1.3. MAKING OPEN SOURCE MORE INCLUSIVE ........................................ 7</td>
</tr>
<tr>
<td>CHAPTER 2. GETTING STARTED WITH THE DATA GRID CLI ................................. 8</td>
</tr>
<tr>
<td>2.1. CREATING AND MODIFYING USERS ............................................... 8</td>
</tr>
<tr>
<td>2.2. CONNECTING TO DATA GRID SERVERS ............................................ 8</td>
</tr>
<tr>
<td>2.3. NAVIGATING CLI RESOURCES ....................................................... 9</td>
</tr>
<tr>
<td>2.3.1. CLI Resources ........................................................................ 10</td>
</tr>
<tr>
<td>2.4. SHUTTING DOWN DATA GRID SERVER ........................................... 11</td>
</tr>
<tr>
<td>2.4.1. Restarting Data Grid Clusters .................................................... 12</td>
</tr>
<tr>
<td>CHAPTER 3. PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI .............. 13</td>
</tr>
<tr>
<td>3.1. CREATING CACHES WITH THE DATA GRID COMMAND LINE INTERFACE (CLI) ................................. 13</td>
</tr>
<tr>
<td>3.1.1. XML Configuration .................................................................. 13</td>
</tr>
<tr>
<td>3.1.2. JSON Configuration .................................................................. 14</td>
</tr>
<tr>
<td>3.2. ADDING CACHE ENTRIES ............................................................... 14</td>
</tr>
<tr>
<td>3.3. CLEARING CACHES AND DELETING ENTRIES .................................... 15</td>
</tr>
<tr>
<td>3.4. DELETING CACHES ........................................................................ 15</td>
</tr>
<tr>
<td>CHAPTER 4. PERFORMING BATCH OPERATIONS ........................................... 16</td>
</tr>
<tr>
<td>4.1. PERFORMING BATCH OPERATIONS WITH FILES .................................. 16</td>
</tr>
<tr>
<td>4.2. PERFORMING BATCH OPERATIONS INTERACTIVELY .............................. 16</td>
</tr>
<tr>
<td>CHAPTER 5. CONFIGURING THE DATA GRID CLI .......................................... 18</td>
</tr>
<tr>
<td>5.1. SETTING DATA GRID CLI PROPERTIES AND PERSISTENT STORAGE ........ 18</td>
</tr>
<tr>
<td>5.2. CREATING COMMAND ALIASES ..................................................... 18</td>
</tr>
<tr>
<td>5.3. TRUSTING DATA GRID SERVER CONNECTIONS ................................ 19</td>
</tr>
<tr>
<td>5.4. DATA GRID CLI STORAGE DIRECTORY ........................................... 19</td>
</tr>
<tr>
<td>CHAPTER 6. WORKING WITH COUNTERS .................................................... 21</td>
</tr>
<tr>
<td>6.1. CREATING COUNTERS ..................................................................... 21</td>
</tr>
<tr>
<td>6.2. ADDING DELTAS TO COUNTERS ...................................................... 22</td>
</tr>
<tr>
<td>CHAPTER 7. QUERYING CACHES WITH PROTOBUF METADATA ....................... 23</td>
</tr>
<tr>
<td>7.1. CONFIGURING MEDIA TYPES ......................................................... 23</td>
</tr>
<tr>
<td>7.2. REGISTERING PROTOBUF SCHEMAS ............................................. 24</td>
</tr>
<tr>
<td>7.3. QUERYING CACHES WITH PROTOBUF SCHEMAS ................................. 25</td>
</tr>
<tr>
<td>CHAPTER 8. PERFORMING CROSS-SITE REPLICATION OPERATIONS .................. 28</td>
</tr>
<tr>
<td>8.1. BRINGING BACKUP LOCATIONS OFFLINE AND ONLINE ....................... 28</td>
</tr>
<tr>
<td>8.2. PUSHING STATE TO BACKUP LOCATIONS ........................................ 28</td>
</tr>
<tr>
<td>CHAPTER 9. COMMAND REFERENCE .......................................................... 30</td>
</tr>
<tr>
<td>9.1. ADD(1) ....................................................................................... 30</td>
</tr>
<tr>
<td>9.1.1. NAME .................................................................................... 30</td>
</tr>
<tr>
<td>9.1.2. SYNOPSIS ............................................................................... 30</td>
</tr>
<tr>
<td>9.1.3. OPTIONS ............................................................................... 30</td>
</tr>
<tr>
<td>9.1.4. EXAMPLES ............................................................................. 30</td>
</tr>
<tr>
<td>9.1.5. SEE ALSO ............................................................................... 30</td>
</tr>
<tr>
<td>9.2. ALIAS(1) .................................................................................... 30</td>
</tr>
<tr>
<td>9.2.1. NAME .................................................................................... 30</td>
</tr>
</tbody>
</table>
9.2.2. SYNOPSIS
9.2.3. EXAMPLES
9.2.4. SEE ALSO

9.3. CACHE(1)
  9.3.1. NAME
  9.3.2. SYNOPSIS
  9.3.3. EXAMPLE
  9.3.4. SEE ALSO

9.4. CAS(1)
  9.4.1. NAME
  9.4.2. SYNOPSIS
  9.4.3. OPTIONS
  9.4.4. EXAMPLE
  9.4.5. SEE ALSO

9.5. CD(1)
  9.5.1. NAME
  9.5.2. DESCRIPTION
  9.5.3. SYNOPSIS
  9.5.4. EXAMPLE
  9.5.5. SEE ALSO

9.6. CLEARCACHE(1)
  9.6.1. NAME
  9.6.2. SYNOPSIS
  9.6.3. EXAMPLES
  9.6.4. SEE ALSO

9.7. CONFIG(1)
  9.7.1. NAME
  9.7.2. SYNOPSIS
  9.7.3. DESCRIPTION
  9.7.4. COMMAND SYNOPSIS
  9.7.5. COMMON OPTIONS
  9.7.6. PROPERTIES
  9.7.7. EXAMPLES
  9.7.8. SEE ALSO

9.8. CONNECT(1)
  9.8.1. NAME
  9.8.2. DESCRIPTION
  9.8.3. SYNOPSIS
  9.8.4. OPTIONS
  9.8.5. EXAMPLE
  9.8.6. SEE ALSO

9.9. CONTAINER(1)
  9.9.1. NAME
  9.9.2. SYNOPSIS
  9.9.3. EXAMPLE
  9.9.4. SEE ALSO

9.10. COUNTER(1)
  9.10.1. NAME
  9.10.2. SYNOPSIS
  9.10.3. EXAMPLE
  9.10.4. SEE ALSO

9.11. CREATE(1)
  9.11.1. NAME
9.29.1. NAME
9.29.2. SYNOPSIS
9.29.3. OPTIONS
9.29.4. EXAMPLES

9.30. STATS(1)
9.30.1. NAME
9.30.2. SYNOPSIS
9.30.3. EXAMPLES
9.30.4. SEE ALSO

9.31. TASK(1)
9.31.1. NAME
9.31.2. SYNOPSIS
9.31.3. EXAMPLES
9.31.4. OPTIONS
9.31.5. SEE ALSO

9.32. UNALIAS(1)
9.32.1. NAME
9.32.2. SYNOPSIS
9.32.3. EXAMPLES
9.32.4. SEE ALSO

9.33. USER(1)
9.33.1. NAME
9.33.2. SYNOPSIS
9.33.3. DESCRIPTION
9.33.4. COMMAND SYNOPSIS
9.33.5. COMMON OPTIONS
9.33.6. USER CREATE/MODIFY OPTIONS
9.33.7. USER LS OPTIONS
9.33.8. USER ENCRYPT-ALL OPTIONS

9.34. VERSION(1)
9.34.1. NAME
9.34.2. SYNOPSIS
9.34.3. EXAMPLE
9.34.4. SEE ALSO
CHAPTER 1. RED HAT DATA GRID

Data Grid is a high-performance, distributed in-memory data store.

**Schemaless data structure**
Flexibility to store different objects as key-value pairs.

**Grid-based data storage**
Designed to distribute and replicate data across clusters.

**Elastic scaling**
Dynamically adjust the number of nodes to meet demand without service disruption.

**Data interoperability**
Store, retrieve, and query data in the grid from different endpoints.

### 1.1. DATA GRID DOCUMENTATION

Documentation for Data Grid is available on the Red Hat customer portal.

- Data Grid 8.1 Documentation
- Data Grid 8.1 Component Details
- Supported Configurations for Data Grid 8.1
- Data Grid 8 Feature Support
- Data Grid Deprecated Features and Functionality

### 1.2. DATA GRID DOWNLOADS

Access the Data Grid Software Downloads on the Red Hat customer portal.

**NOTE**
You must have a Red Hat account to access and download Data Grid software.

### 1.3. MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 2. GETTING STARTED WITH THE DATA GRID CLI

The command line interface (CLI) lets you remotely connect to Data Grid servers to access data and perform administrative functions.

2.1. CREATING AND MODIFYING USERS

Data Grid Server requires users to authenticate against a default property realm. Before you can access Data Grid Server, you must add credentials by creating at least one user and a password. You can also add and modify the security authorization groups to which users belong.

Procedure

1. Open a terminal in $RHDG_HOME.
2. Create and modify Data Grid users with the user command.

TIP

Run help user for more details about using the command.

Creating users and passwords

- Linux

  $ bin/cli.sh user create myuser -p "qwer1234!"

- Microsoft Windows

  $ bin\cli.bat user create myuser -p "qwer1234!"

Creating users with group membership

- Linux

  $ bin/cli.sh user create myuser -p "qwer1234!" -g supervisor,reader,writer

- Microsoft Windows

  $ bin\cli.bat user create myuser -p "qwer1234!" -g supervisor,reader,writer

2.2. CONNECTING TO DATA GRID SERVERS

Establish CLI connections to Data Grid.

Prerequisites

Add user credentials and have at least one running Data Grid server instance.

Procedure

1. Open a terminal in $RHDG_HOME.
2. Start the CLI.
   - Linux:
     ```bash
     $ bin/cli.sh
     ```
   - Microsoft Windows:
     ```cmd
     $ bin\cli.bat
     ```

3. Run the `connect` command and enter your username and password when prompted.
   - Data Grid Server on the default port of **11222**:
     ```
     [disconnected]> connect
     ```
   - Data Grid Server with a port offset of **100**:
     ```
     [disconnected]> connect 127.0.0.1:11322
     ```

### 2.3. Navigating CLI Resources

The Data Grid CLI exposes a navigable tree that allows you to list, describe, and manipulate Data Grid cluster resources.

**TIP**

Press the tab key to display available commands and options. Use the `-h` option to display help text.

When you connect to a Data Grid cluster, it opens in the context of the default cache container.

```
[//containers/default]>
```
- Use `ls` to list resources.
  ```
  [//containers/default]> ls
caches
counters
configurations
schemas
tasks
  ```
- Use `cd` to navigate the resource tree.
  ```
  [//containers/default]> cd caches
  ```
- Use `describe` to view information about resources.
  ```
  [//containers/default]> describe
  {
  "name" : "default",
  "version" : "xx.x.x-FINAL",
  ```
"cluster_name" : "cluster",
"coordinator" : true,
"cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "_protobuf_metadata",
"org.infinispan.DIST_SYNC", "org.infinispan.LOCAL", "org.infinispan.INVALIDATION_SYNC",
"org.infinispan.REPL_SYNC", "org.infinispan.SCATTERED_SYNC",
"org.infinispan.INVALIDATION_ASYNC", "org.infinispan.DIST_ASYNC" ],
"physical_addresses" : ["192.0.2.0:7800"],
"coordinator_address" : "<hostname>",
"cache_manager_status" : "RUNNING",
"created_cache_count" : "1",
"running_cache_count" : "1",
"node_address" : "<hostname>",
"cluster_members" : [ "<hostname1>", "<hostname2>" ],
"cluster_members_physical_addresses" : [ "192.0.2.0:7800", "192.0.2.0:7801" ],
"cluster_size" : 2,
"defined_caches" : [ { 
"name" : "mycache",
"started" : true
}, { 
"name" : "_protobuf_metadata",
"started" : true
} ]
}

2.3.1. CLI Resources

The Data Grid CLI exposes different resources to:

- create, modify, and manage local or clustered caches.
- perform administrative operations for Data Grid clusters.

Cache Resources

```
[//containers/default]> ls
 caches
counters
configurations
schemas
```

caches
Data Grid cache instances. The default cache container is empty. Use the CLI to create caches from templates or `infinispan.xml` files.

counters

- **Strong** or **Weak** counters that record the count of objects.

configurations
Data Grid configurations.

schemas
Protocol Buffers (Protobuf) schemas that structure data in the cache.

tasks
Remote tasks creating and managing Data Grid cache definitions.
Cluster Resources

[hostname@cluster/] > ls
containers
cluster
server

containers
Cache containers on the Data Grid cluster.

cluster
Lists Data Grid servers joined to the cluster.

server
Resources for managing and monitoring Data Grid servers.

2.4. SHUTTING DOWN DATA GRID SERVER

Stop individually running servers or bring down clusters gracefully.

Procedure

1. Create a CLI connection to Data Grid.

2. Shut down Data Grid Server in one of the following ways:
   - Stop all nodes in a cluster with the `shutdown cluster` command, for example:
     
     ```
     [/containers/default]> shutdown cluster
     ```
   This command saves cluster state to the data folder for each node in the cluster. If you use a cache store, the `shutdown cluster` command also persists all data in the cache.

   - Stop individual server instances with the `shutdown server` command and the server hostname, for example:
     
     ```
     [/containers/default]> shutdown server <my_server01>
     ```

   **IMPORTANT**
   
   The `shutdown server` command does not wait for rebalancing operations to complete, which can lead to data loss if you specify multiple hostnames at the same time.

**TIP**

Run `help shutdown` for more details about using the command.

**Verification**

Data Grid logs the following messages when you shut down servers:

```
ISPN080002: Data Grid Server stopping
ISPN000080: Disconnecting JGroups channel cluster
```
2.4.1. Restarting Data Grid Clusters

When you bring Data Grid clusters back online after shutting them down, you should wait for the cluster to be available before adding or removing nodes or modifying cluster state.

If you shutdown clustered nodes with the `shutdown server` command, you must restart each server in reverse order. For example, if you shutdown `server1` and then shutdown `server2`, you should first start `server2` and then start `server1`.

If you shutdown a cluster with the `shutdown cluster` command, clusters become fully operational only after all nodes rejoin. You can restart nodes in any order but the cluster remains in DEGRADED state until all nodes that were joined before shutdown are running.
CHAPTER 3. PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

The command line interface (CLI) lets you remotely connect to Data Grid servers to access data and perform administrative functions.

3.1. CREATING CACHES WITH THE DATA GRID COMMAND LINE INTERFACE (CLI)

Use the Data Grid CLI to add caches from templates or configuration files in XML or JSON format.

Prerequisites
Create a user and start at least one Data Grid server instance.

Procedure

1. Create a CLI connection to Data Grid.

2. Add cache definitions with the `create cache` command.
   - Add a cache definition from an XML or JSON file with the `--file` option.
     
     ```bash
     [/containers/default]> create cache --file=configuration.xml mycache
     ```
   - Add a cache definition from a template with the `--template` option.
     
     ```bash
     [/containers/default]> create cache --template=org.infinispan.DIST_SYNC mycache
     ```
   
   TIP
   Press the tab key after the `--template=` argument to list available cache templates.

3. Verify the cache exists with the `ls` command.

   ```bash
   [/containers/default]> ls caches mycache
   ```

4. Retrieve the cache configuration with the `describe` command.

   ```bash
   [/containers/default]> describe caches/mycache
   ```

Reference

- Creating Data Grid CLI Connections
- Performing Cache Operations with the Data Grid CLI

3.1.1. XML Configuration

Data Grid configuration in XML format must conform to the schema and include:
3.1.2. JSON Configuration

Data Grid configuration in JSON format:

- Requires the cache definition only.
- Must follow the structure of an XML configuration.
  - XML elements become JSON objects.
  - XML attributes become JSON fields.

Example JSON Configuration

```json
{
  "distributed-cache": {
    "name": "myCache",
    "mode": "SYNC",
    "encoding": {
      "media-type": "application/x-protostream"
    },
    "memory": {
      "max-count": 1000000,
      "when-full": "REMOVE"
    }
  }
}
```

3.2. ADDING CACHE ENTRIES

Create key:value pair entries in the data container.

Prerequisites

Create a Data Grid cache that can store your data.

Procedure

1. Create a CLI connection to Data Grid.
2. Add entries into your cache as follows:
   - Use the `put` command from the context of a cache:
     ```
     $ [//containers/default/caches/mymcache]> put hello world
     ```
   - Use the `--cache=` with the `put` command:
     ```
     $ [//containers/default]> put --cache=mymcache hello world
     ```
3. Use the `get` command to verify entries.
   ```
   $ [//containers/default/caches/mymcache]> get hello world
   ```

### 3.3. CLEARING CACHES AND DELETING ENTRIES

Remove data from caches with the Data Grid CLI.

**Procedure**

1. Create a CLI connection to Data Grid.
2. Do one of the following:
   - Delete all entries with the `clearcache` command.
     ```
     $ [//containers/default]> clearcache mycache
     ```
   - Remove specific entries with the `remove` command.
     ```
     $ [//containers/default]> remove --cache=mymcache hello
     ```

### 3.4. DELETING CACHES

Drop caches to remove them and delete all data they contain.

**Procedure**

1. Create a CLI connection to Data Grid.
2. Remove caches with the `drop` command.
   ```
   $ [//containers/default]> drop cache mycache
   ```
CHAPTER 4. PERFORMING BATCH OPERATIONS

Process operations in groups, either interactively or using batch files.

Prerequisites

- A running Data Grid cluster.

4.1. PERFORMING BATCH OPERATIONS WITH FILES

Create files that contain a set of operations and then pass them to the Data Grid CLI.

Procedure

1. Create a file that contains a set of operations.
   For example, create a file named `batch` that creates a cache named `mybatch`, adds two entries to the cache, and disconnects from the CLI.
   ```
   $ cat > batch<<EOF
   create cache --template=org.infinispan.DIST_SYNC mybatch
   put --cache=mybatch hello world
   put --cache=mybatch hola mundo
   disconnect
   EOF
   ```

2. Run the CLI and specify the file as input.
   ```
   $ bin/cli.sh -c localhost:11222 -f batch
   ```

3. Create a new Data Grid CLI connection and verify `mybatch`.
   ```
   [//containers/default]> ls caches
   ___protobuf_metadata
   mybatch
   [//containers/default]> ls caches/mybatch
   hola
   hello
   [//containers/default]> disconnect
   [disconnected]>
   ```

NOTE

CLI batch files support system property expansion. Strings that use the `${property}` format are replaced with the value of the `property` system property.

4.2. PERFORMING BATCH OPERATIONS INTERACTIVELY

Use the standard input stream, `stdin`, to perform batch operations interactively.

Procedure

1. Start the Data Grid CLI in interactive mode.
$ bin/cli.sh -c localhost:11222 -f -

NOTE
If you do not use the -c flag, you must run the connect command.

$ bin/cli.sh -f -
connect

2. Run batch operations, for example:

```
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
disconnect
quit
```

TIP
Use `echo` to add commands in interactive mode.

The following example shows how to use `echo describe` to get cluster information:

```
$ echo describe|bin/cli.sh -c localhost:11222 -f -
{
  "name" : "default",
  "version" : "10.0.0-SNAPSHOT",
  "coordinator" : false,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "___protobuf_metadata",
    "org.infinispan.DIST_SYNC", "qcach", "org.infinispan.LOCAL", "dist_cache_01",
    "org.infinispan.INVALIDATION_SYNC", "org.infinispan.REPL_SYNC",
    "org.infinispan.SCATTERED_SYNC", "mycache", "org.infinispan.INVALIDATION_ASYNC",
    "mybatch", "org.infinispan.DIST_ASYNC" ],
  "cluster_name" : "cluster",
  "physical_addresses" : ["192.168.1.7:7800"],
  "coordinator_address" : "thundercat-34689",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "4",
  "running_cache_count" : "4",
  "node_address" : "thundercat-47082",
  "cluster_members" : [ "thundercat-34689", "thundercat-47082" ],
  "cluster_members_physical_addresses" : [ "10.36.118.25:7801", "192.168.1.7:7800" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "___protobuf_metadata",
    "started" : true
  }, {
    "name" : "mybatch",
    "started" : true
  } ]
}
```
CHAPTER 5. CONFIGURING THE DATA GRID CLI

Define configuration properties for the Data Grid CLI.

5.1. SETTING DATA GRID CLI PROPERTIES AND PERSISTENT STORAGE

Configure Data Grid CLI startup operations and customize the location for persistent storage.

Prerequisites

Create at least one Data Grid user.

Procedure

1. Optionally set a custom path to the Data Grid CLI storage directory in one of the following ways:
   
   - Using the `cli.dir` system property:
     
     ```
     $ bin/cli.sh -Dcli.dir=/path/to/cli/storage ...
     ```
   
   - Using the `ISPN_CLI_DIR` environment variable:
     
     ```
     export ISPN_CLI_DIR=/path/to/cli/storage
     $ bin/cli.sh ...
     ```

2. Set values for configuration properties with the `config set` command.

3. Verify configuration properties with the `config get` command.

   **TIP**

   Run `help config` to review available configuration properties and get example usage.

5.2. CREATING COMMAND ALIASES

Create aliases for Data Grid CLI commands to define custom shortcuts.

Procedure

1. Create aliases with the `alias <alias>=<command>` command. For example, set `q` as an alias for the `quit` command:

   ```
   [/containers/default]> alias q=quit
   ```

2. Run the `alias` command to check the defined aliases.

   ```
   [/containers/default]> alias
   alias q='quit'
   ```

3. Delete aliases with the `unalias` command, for example:
5.3. TRUSTING DATA GRID SERVER CONNECTIONS

Secure Data Grid CLI connections to Data Grid Server with SSL/TLS certificates. If you create a key store as an SSL identity for Data Grid Server, the CLI can validate server certificates to verify the identity.

Prerequisites

- Set up an SSL identity for Data Grid Server.
- Create at least one Data Grid user.

Procedure

1. Specify the location of the server key store, as in the following example:

   ```
   $ bin/cli.sh config set truststore /home/user/my-trust-store.jks
   ```

2. Define the key store password, if necessary, as follows:

   ```
   $ bin/cli.sh config set truststore-password secret
   ```

3. Verify your CLI configuration.

   ```
   $ bin/cli.sh config get truststore
   truststore=/home/user/my-trust-store.jks
   
   $ bin/cli.sh config get truststore-password
   truststore-password=secret
   ```

Reference

Setting Up SSL Identities for Data Grid Server

5.4. DATA GRID CLI STORAGE DIRECTORY

Data Grid CLI stores configuration in the following default directory:

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Default Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux/Unix</td>
<td>$HOME/.config/red_hat_data_grid</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>%APPDATA%/Sun/Java/red_hat_data_grid</td>
</tr>
<tr>
<td>Mac OS</td>
<td>$HOME/Library/Java/red_hat_data_grid</td>
</tr>
</tbody>
</table>

This directory contains the following files:
cli.properties
   Stores values for CLI configuration properties.

aliases
   Stores command aliases.

history
   Stores CLI history.
CHAPTER 6. WORKING WITH COUNTERS

Counters provide atomic increment and decrement operations that record the count of objects.

Prerequisites

- Start the Data Grid CLI.
- Connect to a running Data Grid cluster.

6.1. CREATING COUNTERS

Create strong and weak counters with the Data Grid CLI.

Procedure

1. Create a CLI connection to Data Grid.

2. Run the `create counter` command with the appropriate arguments.
   a. Create `my-weak-counter`.
      
      ```
      [//containers/default]> create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --type=weak my-weak-counter
      ```
   b. Create `my-strong-counter`.
      
      ```
      [//containers/default]> create counter --initial-value=3 --storage=PERSISTENT --type=strong my-strong-counter
      ```

3. List available counters.

   ```
   [//containers/default]> ls counters
   my-strong-counter
   my-weak-counter
   ```

4. Verify counter configurations.
   a. Describe `my-weak-counter`.
      
      ```
      [//containers/default]> describe counters/my-weak-counter
      {
          "weak-counter":{
              "initial-value":5,
              "storage":"PERSISTENT",
              "concurrency-level":1
          }
      }
      ```
   b. Describe `my-strong-counter`.
      
      ```
      [//containers/default]> describe counters/my-strong-counter
      ```
6.2. ADDING DELTAS TO COUNTERS

Increment or decrement counters with arbitrary values.

Procedure

1. Select a counter.

   ```
   [/containers/default]> counter my-weak-counter
   ```

2. List the current count.

   ```
   [/containers/default/counters/my-weak-counter]> ls
   5
   ```

3. Increment the counter value by 2.

   ```
   [/containers/default/counters/my-weak-counter]> add --delta=2
   ```

4. Decrement the counter value by -4.

   ```
   [/containers/default/counters/my-weak-counter]> add --delta=-4
   ```

NOTE

Strong counters return values after the operation is applied. Use `--quiet=true` to hide the return value.

For example, `my-strong-counter]> add --delta=3 --quiet=true`.

Weak counters return empty responses.
CHAPTER 7. QUERYING CACHES WITH PROTOBUF METADATA

Data Grid supports using Protocol Buffers (Protobuf) to structure data in the cache so that you can query it.

Prerequisites

- Start the Data Grid CLI.
- Connect to a running Data Grid cluster.

7.1. CONFIGURING MEDIA TYPES

Encode cache entries with different media types to store data in a format that best suits your requirements.

For example, the following procedure shows you how to configure the `application/x-protostream` media type.

Procedure

1. Create a Data Grid configuration file that adds a distributed cache named `qcache` and configures the media type, for example:

   ```xml
   <infinispan>
   <cache-container>
     <distributed-cache name="qcache">
       <encoding>
         <key media-type="application/x-protostream"/>
         <value media-type="application/x-protostream"/>
       </encoding>
     </distributed-cache>
   </cache-container>
   </infinispan>
   ```

2. Create `qcache` from `pcache.xml` with the `--file=` option.

   ```bash
   [//containers/default]> create cache --file=pcache.xml pcache
   ```

3. Verify `pcache`.

   ```bash
   [//containers/default]> ls caches
   pcache
   ___protobuf_metadata
   [//containers/default]> describe caches/pcache
   {
   "distributed-cache" : {
   "mode" : "SYNC",
   "encoding" : {
   "key" : {
   "media-type" : "application/x-protostream"
   }
   },
   ```
4. Add an entry to `pcache` and check the encoding.

```
[[/containers/default]医疗卫生] put --cache=pcache good morning
[[/containers/default]医疗卫生] cd caches/pcache
[[/containers/default/caches/pcache]医疗卫生] get good
{
  "_type": "string",
  "_value": "morning"
}
```

7.2. REGISTERING PROTOBUF SCHEMAS

Protobuf schemas contain data structures known as messages in `.proto` definition files.

**Procedure**

1. Create a schema file named `person.proto` with the following messages:

```proto
package org.infinispan.rest.search.entity;

message Address {
  required string street = 1;
  required string postCode = 2;
}

message PhoneNumber {
  required string number = 1;
}

message Person {
  optional int32 id = 1;
  required string name = 2;
  required string surname = 3;
  optional Address address = 4;
  repeated PhoneNumber phoneNumbers = 5;
  optional uint32 age = 6;
  enum Gender {
    MALE = 0;
    FEMALE = 1;
  }
  optional Gender gender = 7;
}
```
2. Register `person.proto`.

```
[/containers/default]> schema --upload=person.proto person.proto
```

3. Verify `person.proto`.

```
[/containers/default]> cd caches/___protobuf_metadata
[/containers/default/caches/___protobuf_metadata]> ls
person.proto
[/containers/default/caches/___protobuf_metadata]> get person.proto
```

### 7.3. QUERYING CACHES WITH PROTOBUF SCHEMAS

Data Grid automatically converts JSON to Protobuf so that you can read and write cache entries in JSON format and use Protobuf schemas to query them.

For example, consider the following JSON documents:

**lukecage.json**

```json
{
   "_type":"org.infinispan.rest.search.entity.Person",
   "id":2,
   "name":"Luke",
   "surname":"Cage",
   "gender":"MALE",
   "address":{"street":"38th St","postCode":"NY 11221"},
   "phoneNumbers":[{"number":4444},{"number":5555}]
}
```

**jessicajones.json**

```json
{
   "_type":"org.infinispan.rest.search.entity.Person",
   "id":1,
   "name":"Jessica",
   "surname":"Jones",
   "gender":"FEMALE",
   "address":{"street":"46th St","postCode":"NY 10036"},
   "phoneNumbers":[{"number":1111},{"number":2222},{"number":3333}]
}
```

**matthewmurdock.json**

```json
{
   "_type":"org.infinispan.rest.search.entity.Person",
   "id":3,
   "name":"Matthew",
   "surname":"Murdock",
   "gender":"MALE",
   "address":{"street":"57th St","postCode":"NY 10019"},
   "phoneNumbers":[]
}
```
Each of the preceding JSON documents contains:

- A _type field that identifies the Protobuf message to which the JSON document corresponds.
- Several fields that correspond to datatypes in the person.proto schema.

**Procedure**

1. Navigate to the **pcache** cache.

```bash
[//containers/default/caches]> cd pcache
```

2. Add each JSON document as an entry to the cache, for example:

```bash
[//containers/default/caches/pcache]> put --encoding=application/json --file=jessicajones.json jessicajones
[//containers/default/caches/pcache]> put --encoding=application/json --file=matthewmurdock.json matthewmurdock
[//containers/default/caches/pcache]> put --encoding=application/json --file=lukecage.json lukecage
```

3. Verify that the entries exist.

```bash
[//containers/default/caches/pcache]> ls
lukecage
matthewmurdock
jessicajones
```

4. Query the cache to return entries from the Protobuf **Person** entity where the gender datatype is **MALE**.

```bash
[//containers/default/caches/pcache]> query "from org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
{
  "total_results" : 2,
  "hits" : [ {
    "hit" : {
      "_type" : "org.infinispan.rest.search.entity.Person",
      "id" : 2,
      "name" : "Luke",
      "surname" : "Cage",
      "gender" : "MALE",
      "address" : {
        "street" : "38th St",
        "postCode" : "NY 11221"
      },
      "phoneNumbers" : [ {
        "number" : "4444"
      }, {
        "number" : "5555"
      } ]
    }
  }, {
    "hit" : {
      "_type" : "org.infinispan.rest.search.entity.Person",
      "id" : 2,
      "name" : "Luke",
      "surname" : "Cage",
      "gender" : "MALE",
      "address" : {
        "street" : "38th St",
        "postCode" : "NY 11221"
      },
      "phoneNumbers" : [ {
        "number" : "4444"
      }, {
        "number" : "5555"
      } ]
    }
  }
}
```
"id" : 3,
"name" : "Matthew",
"surname" : "Murdock",
"gender" : "MALE",
"address" : {
  "street" : "57th St",
  "postCode" : "NY 10019"
}
CHAPTER 8. PERFORMING CROSS-SITE REPLICATION OPERATIONS

Data Grid clusters running in different locations can discover and communicate with each other to backup data.

Prerequisites

- Start the Data Grid CLI.
- Connect to a running Data Grid cluster.

8.1. BRINGING BACKUP LOCATIONS OFFLINE AND ONLINE

Take backup locations offline manually and bring them back online.

Procedure

1. Create a CLI connection to Data Grid.

2. Check if backup locations are online or offline with the `site status` command:

   ```
   //containers/default]> site status --cache=cacheName --site=NYC
   ```

   **NOTE**
   --site is an optional argument. If not set, the CLI returns all backup locations.

3. Manage backup locations as follows:

   - Bring backup locations online with the `bring-online` command:
     ```
     //containers/default]> site bring-online --cache=customers --site=NYC
     ```
   
   - Take backup locations offline with the `take-offline` command:
     ```
     //containers/default]> site take-offline --cache=customers --site=NYC
     ```

   For more information and examples, run the `help site` command.

8.2. PUSHING STATE TO BACKUP LOCATIONS

Transfer cache state to remote backup locations.

Procedure

1. Create a CLI connection to Data Grid.

2. Use the `site` command to push state transfer, as in the following example:

   ```
   //containers/default]> site push-site-state --cache=cacheName --site=NYC
   ```
For more information and examples, run the **help site** command.
CHAPTER 9. COMMAND REFERENCE

Review manual pages for Data Grid CLI commands.

TIP

Use `help` command to access manual pages directly from your CLI session.

For example, to view the manual page for the `get` command do the following:

```
$ help get
```

9.1. ADD(1)

9.1.1. NAME

add - increments and decrements counters with arbitrary values.

9.1.2. SYNOPSIS

add ['OPTIONS'] ['COUNTER_NAME']

9.1.3. OPTIONS

`--delta='nnn'`

Sets a delta to increment or decrement the counter value. Defaults to 1.

`-q, --quiet='[true|false]'`

Hides return values for strong counters. The default is `false`.

9.1.4. EXAMPLES

```
add --delta=10 cnt_a
```

Increments the value of `cnt_a` by 10.

```
add --delta=-5 cnt_a
```

Decrements the value of `cnt_a` by 5.

9.1.5. SEE ALSO

`cas(1), reset(1)`

9.2. ALIAS(1)

9.2.1. NAME

alias - creates or displays aliases.

9.2.2. SYNOPSIS

```
alias ['ALIAS-NAME'='COMMAND']
```

Red Hat Data Grid 8.1 Data Grid Command Line Interface
9.2.3. EXAMPLES

alias q=quit
Creates q as an alias for the quit command.

alias
Lists all defined aliases.

9.2.4. SEE ALSO

config(1), unalias(1)

9.3. CACHE(1)

9.3.1. NAME

cache - selects the default cache for subsequent commands.

9.3.2. SYNOPSIS

cache ['CACHE_NAME']

9.3.3. EXAMPLE

cache mycache
Selects mycache and is the same as navigating the resource tree using cd caches/mycache.

9.3.4. SEE ALSO

cd(1), clear(1), container(1), get(1), put(1), remove(1)

9.4. CAS(1)

9.4.1. NAME

cas - performs 'compare-and-swap' operations on strong counters.

9.4.2. SYNOPSIS

cas ['OPTIONS'] ['COUNTER_NAME']

9.4.3. OPTIONS

--expect='nnn'
  Specifies the expected value of the counter.

--value='nnn'
  Sets a new value for the counter.

-q, --quiet='[true|false]'
  Hides return values. The default is false.
9.4.4. EXAMPLE

    cas --expect=10 --value=20 cnt_a
Sets the value of cnt_a to 20 only if the current value is 10

9.4.5. SEE ALSO

    add(1), cas(1), reset(1)

9.5. CD(1)

9.5.1. NAME

cd - navigates the server resource tree.

9.5.2. DESCRIPTION

    PATH can be absolute or relative to the current resource. ../ specifies parent resources.

9.5.3. SYNOPSIS

    cd ['PATH']

9.5.4. EXAMPLE

    cd caches
Changes to the caches path in the resource tree.

9.5.5. SEE ALSO

    cache(1), ls(1), container(1)

9.6. CLEARCACHE(1)

9.6.1. NAME

    clearcache - removes all entries from a cache.

9.6.2. SYNOPSIS

    clearcache ['CACHE_NAME']

9.6.3. EXAMPLES

    clearcache mycache
Removes all entries from mycache.

9.6.4. SEE ALSO

    cache(1), drop(1), remove(1)
9.7. CONFIG(1)

9.7.1. NAME
config - manages CLI configuration properties.

9.7.2. SYNOPSIS
config
config set 'name' 'value'
config get 'name'

9.7.3. DESCRIPTION
Manage (list, set, get) CLI configuration properties.

9.7.4. COMMAND SYNOPSIS
config
   Lists all configuration properties that are set.
config set 'name' ['value']
   Sets the value of a specific property. If you do not specify a value, the property is not set.
config get 'name'
   Retrieves the value of a specific property.

9.7.5. COMMON OPTIONS
These options apply to all commands:

- -h, --help
   Displays a help page for the command or sub-command.

9.7.6. PROPERTIES
autoconnect-url
   Specifies the URL to which the CLI automatically connects on startup.
autoexec
   Specifies the path of a CLI batch file to execute on startup.
trustall
   Specifies whether to trust all server certificates. Values are false (default) and true.
truststore
   Defines the path to a keystore that contains a certificate chain that verifies server identity.
truststore-password
   Specifies a password to access the keystore.

9.7.7. EXAMPLES
**config set autoconnect-url** http://192.0.2.0:11222
Connects to a server at a custom IP address when you start the CLI.

**config get autoconnect-url**
Returns the value for the *autoconnect-url* configuration property.

**config set autoexec /path/to/mybatchfile**
Runs a batch file named "mybatchfile" when you start the CLI.

**config set trustall true**
Trusts all server certificates.

**config set truststore /home/user/my-trust-store.jks**
Specifies the path of a keystore named "my-trust-store.jks".

**config set truststore-password secret**
Sets the keystore password, if required.

### 9.7.8. SEE ALSO

alias(1), unalias(1)

### 9.8. CONNECT(1)

#### 9.8.1. NAME
connect - connects to running ${infinispan.brand.name} servers.

#### 9.8.2. DESCRIPTION
Defaults to **http://localhost:11222** and prompts for credentials if authentication is required.

#### 9.8.3. SYNOPSIS
connect ['OPTIONS'] ['SERVER_LOCATION']

#### 9.8.4. OPTIONS

- **-u, --username='USERNAME'**
  Specifies a username to authenticate with ${infinispan.brand.name} servers.

- **-p, --password='PASSWORD'**
  Specifies passwords.

#### 9.8.5. EXAMPLE
connect 127.0.0.1:11322 -u test -p changeme
Connects to a locally running server using a port offset of 100 and example credentials.

#### 9.8.6. SEE ALSO

disconnect(1)
9.9. CONTAINER(1)

9.9.1. NAME
container - selects the container for running subsequent commands.

9.9.2. SYNOPSIS
container ['CONTAINER_NAME']

9.9.3. EXAMPLE
container default
Selects the default container and is the same as navigating the resource tree using cd containers/default.

9.9.4. SEE ALSO
cd(1), clear(1), container(1), get(1), put(1), remove(1)

9.10. COUNTER(1)

9.10.1. NAME
counter - selects the default counter for subsequent commands.

9.10.2. SYNOPSIS
counter ['COUNTER_NAME']

9.10.3. EXAMPLE
counter cnt_a
Selects cnt_a and is the same as navigating the resource tree using cd counters/cnt_a.

9.10.4. SEE ALSO
add(1), cas(1)

9.11. CREATE(1)

9.11.1. NAME
create - creates caches and counters on ${infinispan.brand.name} servers.

9.11.2. SYNOPSIS
create cache ['OPTIONS'] CACHE_NAME
create counter ['OPTIONS'] COUNTER_NAME
9.11.3. CREATE CACHE OPTIONS

- \texttt{-f, \textemdash \text{-file='FILE'}}
  Specifies a configuration file in JSON or XML format.

- \texttt{-t, \textemdash \text{-template='TEMPLATE'}}
  Specifies a configuration template. Use tab autocompletion to see available templates.

- \texttt{-v, \textemdash \text{-volatile='[true|false]'}}
  Specifies whether the cache is persistent or volatile. The default is false.

9.11.4. CREATE COUNTER OPTIONS

- \texttt{-t, \textemdash \text{-type='[weak|strong]'}}
  Specifies if the counter is weak or strong.

- \texttt{-s, \textemdash \text{-storage='[PERSISTENT|VOLATILE]'}}
  Specifies whether the counter is persistent or volatile.

- \texttt{-c, \textemdash \text{-concurrency-level='nnn'}}
  Sets the concurrency level of the counter.

- \texttt{-i, \textemdash \text{-initial-value='nnn'}}
  Sets the initial value of the counter.

- \texttt{-l, \textemdash \text{-lower-bound='nnn'}}
  Sets the lower bound of a \texttt{strong} counter.

- \texttt{-u, \textemdash \text{-upper-bound='nnn'}}
  Sets the upper bound of a \texttt{strong} counter.

9.11.5. EXAMPLES

\texttt{create cache \textemdash \text{-template=org.infinispan.DIST\_SYNC} mycache}
Creates a cache named \texttt{mycache} from the \texttt{DIST\_SYNC} template.

\texttt{create counter \textemdash \text{-initial-value=3 \text{-storage=PERSISTENT \text{-type=strong}} cnt\_a}
Creates a strong counter named \texttt{cnt\_a}.

9.11.6. SEE ALSO

drop(1)

9.12. DESCRIBE(1)

9.12.1. NAME

describe - displays information about resources.

9.12.2. SYNOPSIS

describe ['PATH']

9.12.3. EXAMPLES
describe //containers/default
Displays information about the default container.

describe //containers/default/caches/mycache
Displays information about the mycache cache.

describe //containers/default/caches/mycache/k1
Displays information about the k1 key.

describe //containers/default/counters/cnt1
Displays information about the cnt1 counter.

9.12.4. SEE ALSO

cd(1), ls(1)

9.13. DISCONNECT(1)

9.13.1. NAME
disconnect - ends CLI sessions with ${infinispan.brand.name} servers.

9.13.2. SYNOPSIS
disconnect

9.13.3. EXAMPLE
disconnect
Ends the current CLI session.

9.13.4. SEE ALSO
connect(1)

9.14. DROP(1)

9.14.1. NAME
drop - deletes caches and counters.

9.14.2. SYNOPSIS
drop cache CACHE_NAME
drop counter COUNTER_NAME

9.14.3. EXAMPLES
drop cache mycache
Deletes the mycache cache.
drop counter cnt_a
Deletes the cnt_a counter.

9.14.4. SEE ALSO
create(1), clearcache(1)

9.15. ENCODING(1)

9.15.1. NAME
eencoding - displays and sets the encoding for cache entries.

9.15.2. DESCRIPTION
Sets a default encoding for put and get operations on a cache. If no argument is specified, the encoding command displays the current encoding.

Valid encodings use standard MIME type (IANA media types) naming conventions, such as the following:

- text/plain
- application/json
- application/xml
- application/octet-stream

9.15.3. SYNOPSIS
eencoding ['ENCODING']

9.15.4. EXAMPLE
eencoding application/json
Configures the currently selected cache to encode entries as application/json.

9.15.5. SEE ALSO
get(1), put(1)

9.16. GET(1)

9.16.1. NAME
get - retrieves entries from a cache.

9.16.2. SYNOPSIS
get ['OPTIONS'] KEY

9.16.3. OPTIONS
-c, --cache='NAME'
  Specifies the cache from which to retrieve entries. Defaults to the currently selected cache.

9.16.4. EXAMPLE

get hello -c mycache
Retrieves the value of the key named hello from mycache.

9.16.5. SEE ALSO

query(1), put(1)

9.17. HELP(1)

9.17.1. NAME

help - prints manual pages for commands.

9.17.2. SYNOPSIS

help ['COMMAND']

9.17.3. EXAMPLE

help get
Prints the manual page for the get command.

9.17.4. SEE ALSO

version(1)

9.18. LOGGING(1)

9.18.1. NAME

logging - inspects and manipulates the $infinispan.brand.name server runtime logging configuration.

9.18.2. SYNOPSIS

logging list-loggers
logging list-appenders
logging set ['OPTIONS'] [LOGGER_NAME]
logging remove LOGGER_NAME

9.18.3. LOGGING SET OPTIONS

-l, --level='OFF|TRACE|DEBUG|INFO|WARN|ERROR|ALL'
  Specifies the logging level for the specific logger.
Specifies an appenders to set on the specific logger. The option can be repeated for multiple appenders.

NOTE

calling logging set without a logger name will modify the root logger.

9.18.4. EXAMPLES

logging list-loggers
Lists all available loggers

logging set --level=DEBUG --appenders=FILE org.infinispan
Sets the log level for the org.infinispan logger to DEBUG and configures it to use the FILE appender.

9.19. LS(1)

9.19.1. NAME
ls - lists resources for the current path or a given path.

9.19.2. SYNOPSIS
ls ['PATH']

9.19.3. EXAMPLES
ls caches
Lists the available caches.

ls ../
Lists parent resources.

9.19.4. SEE ALSO
cd(1)

9.20. MIGRATE(1)

9.20.1. NAME
migrate - migrates data from one version of $\{infinispan.brand.name\} to another.

9.20.2. SYNOPSIS
migrate cluster synchronize
migrate cluster disconnect

9.20.3. DESCRIPTION
Use the **migrate** command to migrate data from one version of `${infinispan.brand.name}` to another.

9.20.4. COMMAND SYNOPSIS

Migrate clusters

**migrate cluster synchronize**

Synchronize data between the source cluster and the target cluster.

**migrate cluster disconnect**

Disconnects the target cluster from the source cluster.

9.20.5. COMMON OPTIONS

These options apply to all commands:

- **-h, --help**
  
  Displays a help page for the command or sub-command.

9.20.6. CLUSTER SYNCHRONIZE OPTIONS

- **-c, --cache='name'**
  
  The name of the cache to synchronize.

- **-b, --read-batch='num'**
  
  The amount of entries to process in a batch. Defaults to 10000.

- **-t, --threads='num'**
  
  The number of threads to use. Defaults to the number of cores on the server.

9.20.7. CLUSTER DISCONNECT OPTIONS

- **-c, --cache='name'**
  
  The name of the cache to disconnect from the source.

9.21. PATCH(1)

9.21.1. NAME

patch - manages server patches.

9.21.2. DESCRIPTION

List, describe, install, rollback, and create server patches.

Patches are zip archive files that contain artifacts to upgrade servers and resolve issues or add new features. Patches can apply target versions to multiple server installations with different versions.

9.21.3. SYNOPSIS

patch ls

patch install 'patch-file'
patch describe 'patch-file'

patch rollback

patch create 'patch-file' 'target-server' 'source-server-1' ['source-server-2'...]

9.21.4. PATCH LIST OPTIONS
--server='path/to/server'
   Sets the path to a target server outside the current server home directory.
-v, --verbose
   Shows the content of each installed patch, including information about individual files.

9.21.5. PATCH INSTALL OPTIONS
--dry-run
   Shows the operations that the patch performs without applying any changes.
--server='path/to/server'
   Sets the path to a target server outside the current server home directory.

9.21.6. PATCH DESCRIBE OPTIONS
-v, --verbose
   Shows the content of the patch, including information about individual files

9.21.7. PATCH ROLLBACK OPTIONS
--dry-run
   Shows the operations that the patch performs without applying any changes.
--server='path/to/server'
   Sets the path to a target server outside the current server home directory.

9.21.8. PATCH CREATE OPTIONS
-q, --qualifier='name'
   Specifies a descriptive qualifier string for the patch; for example, 'one-off for issue nnnn'.

9.21.9. EXAMPLES
patch ls
Lists the patches currently installed on a server in order of installation.

patch install mypatch.zip
Installs "mypatch.zip" on a server in the current directory.

patch install mypatch.zip --server=/path/to/server/home
Installs "mypatch.zip" on a server in a different directory.

patch describe mypatch.zip
Displays the target version and list of source versions for "mypatch.zip".
patch create mypatch.zip 'target-server' 'source-server-1' ['source-server-2'...]
Creates a patch file named "mypatch.zip" that uses the version of the target server and applies to the source server versions.

patch rollback
Rolls back the last patch that was applied to a server and restores the previous version.

9.22. PUT(1)

9.22.1. NAME
put - adds or updates cache entries.

9.22.2. DESCRIPTION
Creates entries for new keys. Replaces values for existing keys.

9.22.3. SYNOPSIS
put ['OPTIONS'] KEY [VALUE]

9.22.4. OPTIONS
-c, --cache='NAME'
   Specifies the name of the cache. Defaults to the currently selected cache.
-e, --encoding='ENCODING'
   Sets the media type for the value.
-f, --file='FILE'
   Specifies a file that contains the value for the entry.
-l, --ttl='TTL'
   Sets the number of seconds before the entry is automatically deleted (time-to-live). Defaults to the value for lifespan in the cache configuration if 0 or not specified. If you set a negative value, the entry is never deleted.
-i, --max-idle='MAXIDLE'
   Sets the number of seconds that the entry can be idle. If a read or write operation does not occur for an entry after the maximum idle time elapses, the entry is automatically deleted. Defaults to the value for maxIdle in the cache configuration if 0 or not specified. If you set a negative value, the entry is never deleted.
-a, --if-absent=[true|false]
   Puts an entry only if it does not exist.

9.22.5. EXAMPLES
put -c mycache hello world
   Adds the hello key with a value of world to the mycache cache.

put -c mycache -f myfile -i 500 hola
   Adds the hola key with the value from the contents of myfile. Also sets a maximum idle of 500 seconds.
9.22.6. SEE ALSO
get(1), remove(1)

9.23. QUERY(1)

9.23.1. NAME
query - retrieves entries that match Ickle query strings.

9.23.2. SYNOPSIS
query ['OPTIONS'] QUERY_STRING

9.23.3. OPTIONS
- c, --cache='NAME'
  Specifies the cache to query. Defaults to the currently selected cache.
--max-results='MAX_RESULTS'
  Sets the number of results to return. The default is 10.
-o, --offset='OFFSET'
  Specifies the index of the first result to return. The default is 0.
--query-mode='QUERY_MODE'
  Specifies how the server executes the query. Values are FETCH and BROADCAST. The default is FETCH.

9.23.4. EXAMPLES
query "from org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
Queries the currently selected cache to return entries from a Protobuf Person entity where the gender datatype is MALE.

9.23.5. SEE ALSO
schema(1)

9.24. QUIT(1)

9.24.1. NAME
quit - exits the command line interface.

9.24.2. SYNOPSIS
quit

9.24.3. EXAMPLE
9.24.4. SEE ALSO
disconnect(1), shutdown(1)

9.25. REMOVE(1)

9.25.1. NAME
remove - deletes entries from a cache.

9.25.2. SYNOPSIS
remove KEY ['OPTIONS']

9.25.3. OPTIONS
--cache='NAME'
    Specifies the cache from which to remove entries. Defaults to the currently selected cache.

9.25.4. EXAMPLE
remove --cache=mycache hola
Deletes the hola entry from the mycache cache.

9.25.5. SEE ALSO
cache(1), drop(1), clearcache(1)

9.26. RESET(1)

9.26.1. NAME
reset - restores the initial values of counters.

9.26.2. SYNOPSIS
reset ['COUNTER_NAME']

9.26.3. EXAMPLE
reset cnt_a
Resets the cnt_a counter.

9.26.4. SEE ALSO
add(1), cas(1), drop(1)
9.27. SCHEMA(1)

9.27.1. NAME

schema - uploads and registers protobuf schemas.

9.27.2. SYNOPSIS

schema ['OPTIONS'] SCHEMA_NAME

9.27.3. OPTIONS

-u, --upload='FILE'
   Uploads a file as a protobuf schema with the given name.

9.27.4. EXAMPLE

    schema --upload=person.proto person.proto
    Registers a person.proto Protobuf schema.

9.27.5. SEE ALSO

    query(1)

9.28. SHUTDOWN(1)

9.28.1. NAME

shutdown - stops running servers or brings clusters down gracefully.

9.28.2. SYNOPSIS

shutdown server ['SERVERS']
shutdown cluster

9.28.3. EXAMPLES

    shutdown server
    Stops the server to which the CLI is connected.

    shutdown server my_server01
    Stops the server with hostname my_server01.

    shutdown cluster
    Stores cluster state, persists entries if you use a cache store, and stops all nodes.

9.28.4. SEE ALSO

    connect(1), disconnect(1), quit(1)
9.29. SITE(1)

9.29.1. NAME

site - manages backup locations and performs cross-site replication operations.

9.29.2. SYNOPSIS

site status ['OPTIONS']

site bring-online ['OPTIONS']

site take-offline ['OPTIONS']

site push-site-state ['OPTIONS']

site cancel-push-state ['OPTIONS']

site cancel-receive-state ['OPTIONS']

site push-site-status ['OPTIONS']

9.29.3. OPTIONS

--cache='CACHE_NAME'

Specifies a cache.

--site='SITE_NAME'

Specifies a backup location.

9.29.4. EXAMPLES

site status --cache=mycache
Returns the status of all backup locations for mycache.

site status --cache=mycache --site=NYC
Returns the status of NYC for mycache.

site bring-online --cache=mycache --site=NYC
Brings the site NYC online for mycache.

site take-offline --cache=mycache --site=NYC
Takes the site NYC offline for mycache.

site push-site-state --cache=mycache --site=NYC
Backs up caches to remote backup locations.

site push-site-status --cache=mycache
Displays the status of the operation to backup mycache.

site cancel-push-state --cache=mycache --site=NYC
 Cancels the operation to backup mycache to NYC.

site cancel-receive-state --cache=mycache --site=NYC
 Cancels the operation to receive state from NYC.
site clear-push-state-status --cache=myCache
Clears the status of the push state operation for mycache.

9.30. STATS(1)

9.30.1. NAME
stats - displays statistics about resources.

9.30.2. SYNOPSIS
stats ['PATH']

9.30.3. EXAMPLES
stats //containers/default
Displays statistics about the default container.

stats //containers/default/caches/mycache
Displays statistics about the mycache cache.

9.30.4. SEE ALSO
cd(1), ls(1), describe(1)

9.31. TASK(1)

9.31.1. NAME
task - executes and uploads server-side tasks and scripts

9.31.2. SYNOPSIS
task upload --file='script' 'TASK_NAME'
task exec ['TASK_NAME']

9.31.3. EXAMPLES
task upload --file=hello.js hello
Uploads a script from a hello.js file and names it hello.

task exec @@cache@names
Runs a task that returns available cache names.

task exec hello -Pgreetee=world
Runs a script named hello and specifies the greetee parameter with a value of world.

9.31.4. OPTIONS
-P, --parameters='PARAMETERS'
Passes parameter values to tasks and scripts.
-f, --file='FILE'
   Uploads script files with the given names.

9.31.5. SEE ALSO
ls(1)

9.32. UNALIAS(1)

9.32.1. NAME
unalias - deletes aliases.

9.32.2. SYNOPSIS
unalias 'ALIAS-NAME'

9.32.3. EXAMPLES
unalias q
Deletes the q alias.

9.32.4. SEE ALSO
config(1), alias(1)

9.33. USER(1)

9.33.1. NAME
user - manages ${infinispan.brand.name} users in property security realms.

9.33.2. SYNOPSIS
user ls
user create 'username'
user describe 'username'
user remove 'username'
user password 'username'
user groups 'username'
user encrypt-all

9.33.3. DESCRIPTION
Manage (list, create, describe, remove, modify) users stored in a property security realm. Note: You can use this command only with property realms.
9.33.4. COMMAND SYNOPSIS

user ls
Lists the users or groups which are present in the property file.

user create 'username'
Creates a user after prompting for a password.

user describe 'username'
Describes a user, including its username, realm and any groups it belongs to.

user remove 'username'
Removes the specified user from the property file.

user password 'username'
Changes the password for a user.

user groups 'username'
Sets the groups to which a user belongs.

user encrypt-all
Encrypt all passwords in a plain-text user property file.

9.33.5. COMMON OPTIONS

These options apply to all commands:

-h, --help
Displays a help page for the command or sub-command.

-s, --server-root='path-to-server-root'
The path to the server root. Defaults to server.

-f, --users-file='users.properties'
The name of the property file which contains the user passwords. Defaults to users.properties.

-w, --groups-file='groups.properties'
The name of the property file which contains the user to groups mapping. Defaults to groups.properties.

9.33.6. USER CREATE/MODIFY OPTIONS

-a, --algorithms
Specifies the algorithms used to hash the password.

-g, --groups='group1,group2,...'
Specifies the groups to which the user belongs.

-p, --password='password'
Specifies the user’s password.

-r, --realm='realm'
Specifies the realm name.

--plain-text
Whether passwords should be stored in plain-text (not recommended).

9.33.7. USER LS OPTIONS
--groups
   Shows a list of groups instead of the users.

9.33.8. USER ENCRYPT-ALL OPTIONS
-a, --algorithms
   Specifies the algorithms used to hash the password.

9.34. VERSION(1)

9.34.1. NAME
version - displays the server version and CLI version.

9.34.2. SYNOPSIS
version

9.34.3. EXAMPLE
version
   Returns the version for the server and the CLI.

9.34.4. SEE ALSO
help(1)