



## Red Hat Data Grid 8.4

# Using the Data Grid Command Line Interface

Access and manage remote caches with the Data Grid CLI



## Red Hat Data Grid 8.4 Using the Data Grid Command Line Interface

---

Access and manage remote caches with the Data Grid CLI

## Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

Connect to Data Grid Server clusters with the command line interface (CLI) to access data and perform management operations with remote caches.

# Table of Contents

<b>RED HAT DATA GRID</b> .....	<b>9</b>
<b>DATA GRID DOCUMENTATION</b> .....	<b>10</b>
<b>DATA GRID DOWNLOADS</b> .....	<b>11</b>
<b>MAKING OPEN SOURCE MORE INCLUSIVE</b> .....	<b>12</b>
<b>CHAPTER 1. GETTING STARTED WITH DATA GRID CLI</b> .....	<b>13</b>
1.1. CREATING DATA GRID USERS	13
1.1.1. Granting roles to users	13
1.1.2. Adding users to groups	14
1.1.3. Data Grid user roles and permissions	15
1.2. CONNECTING TO DATA GRID SERVERS	16
1.3. NAVIGATING CLI RESOURCES	16
1.3.1. CLI Resources	17
1.4. SHUTTING DOWN DATA GRID SERVER	18
1.4.1. Shutdown and restart of Data Grid clusters	19
Cluster shutdown	19
Server shutdown	19
<b>CHAPTER 2. PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI</b> .....	<b>21</b>
2.1. CREATING REMOTE CACHES WITH THE DATA GRID CLI	21
2.1.1. Cache configuration	21
Distributed caches	22
Replicated caches	24
Multiple caches	26
2.2. MODIFYING DATA GRID CACHE CONFIGURATION	28
2.3. ADDING CACHE ENTRIES	28
2.4. CLEARING CACHES AND DELETING ENTRIES	29
2.5. DELETING CACHES	29
2.6. CONFIGURING AUTOMATIC CACHE REBALANCING	30
2.7. SET A STABLE TOPOLOGY	30
<b>CHAPTER 3. PERFORMING BATCH OPERATIONS</b> .....	<b>31</b>
3.1. PERFORMING BATCH OPERATIONS WITH FILES	31
3.2. PERFORMING BATCH OPERATIONS INTERACTIVELY	31
<b>CHAPTER 4. CONFIGURING THE DATA GRID CLI</b> .....	<b>33</b>
4.1. SETTING DATA GRID CLI PROPERTIES AND PERSISTENT STORAGE	33
4.2. CREATING COMMAND ALIASES	33
4.3. TRUSTING DATA GRID SERVER CONNECTIONS	34
4.4. DATA GRID CLI STORAGE DIRECTORY	35
<b>CHAPTER 5. WORKING WITH COUNTERS</b> .....	<b>36</b>
5.1. CREATING COUNTERS	36
5.2. ADDING DELTAS TO COUNTERS	37
<b>CHAPTER 6. PERFORMING CROSS-SITE REPLICATION OPERATIONS</b> .....	<b>38</b>
6.1. BRINGING BACKUP LOCATIONS OFFLINE AND ONLINE	38
6.2. CONFIGURING CROSS-SITE STATE TRANSFER MODES	39
6.3. PUSHING STATE TO BACKUP LOCATIONS	39
<b>CHAPTER 7. BACKING UP AND RESTORING DATA GRID CLUSTERS</b> .....	<b>40</b>

7.1. BACKING UP DATA GRID CLUSTERS	40
7.2. RESTORING DATA GRID CLUSTERS FROM BACKUP ARCHIVES	41
<b>CHAPTER 8. COMMAND REFERENCE</b> .....	<b>42</b>
8.1. ADD(1)	42
8.1.1. NAME	42
8.1.2. SYNOPSIS	42
8.1.3. OPTIONS	42
8.1.4. EXAMPLES	42
8.1.5. SEE ALSO	42
8.2. ALIAS(1)	42
8.2.1. NAME	42
8.2.2. SYNOPSIS	42
8.2.3. EXAMPLES	43
8.2.4. SEE ALSO	43
8.3. ALTER(1)	43
8.3.1. NAME	43
8.3.2. SYNOPSIS	43
8.3.3. ALTER CACHE OPTIONS	43
8.3.4. EXAMPLES	43
8.3.5. SEE ALSO	44
8.4. AVAILABILITY(1)	44
8.4.1. NAME	44
8.4.2. SYNOPSIS	44
8.4.3. OPTIONS	44
8.4.4. EXAMPLES	44
8.5. BACKUP(1)	44
8.5.1. NAME	44
8.5.2. SYNOPSIS	44
8.5.3. BACKUP CREATE OPTIONS	44
8.5.4. BACKUP GET OPTIONS	45
8.5.5. BACKUP RESTORE OPTIONS	45
8.5.6. EXAMPLES	45
8.5.7. SEE ALSO	46
8.6. BENCHMARK(1)	46
8.6.1. NAME	46
8.6.2. SYNOPSIS	46
8.6.3. BENCHMARK OPTIONS	46
8.6.4. EXAMPLES	47
8.7. CACHE(1)	47
8.7.1. NAME	47
8.7.2. SYNOPSIS	48
8.7.3. EXAMPLE	48
8.7.4. SEE ALSO	48
8.8. CAS(1)	48
8.8.1. NAME	48
8.8.2. SYNOPSIS	48
8.8.3. OPTIONS	48
8.8.4. EXAMPLE	48
8.8.5. SEE ALSO	48
8.9. CD(1)	48
8.9.1. NAME	48
8.9.2. DESCRIPTION	48

---

8.9.3. SYNOPSIS	49
8.9.4. EXAMPLE	49
8.9.5. SEE ALSO	49
8.10. CLEARCACHE(1)	49
8.10.1. NAME	49
8.10.2. SYNOPSIS	49
8.10.3. EXAMPLES	49
8.10.4. SEE ALSO	49
8.11. CONFIG(1)	49
8.11.1. NAME	49
8.11.2. SYNOPSIS	49
8.11.3. DESCRIPTION	49
8.11.4. COMMAND SYNOPSIS	50
8.11.5. COMMON OPTIONS	50
8.11.6. CONVERT OPTIONS	50
8.11.7. PROPERTIES	50
8.11.8. EXAMPLES	51
8.11.9. SEE ALSO	51
8.12. CONNECT(1)	51
8.12.1. NAME	51
8.12.2. DESCRIPTION	51
8.12.3. SYNOPSIS	51
8.12.4. OPTIONS	51
8.12.5. EXAMPLE	52
8.12.6. SEE ALSO	52
8.13. CONTAINER(1)	52
8.13.1. NAME	52
8.13.2. SYNOPSIS	52
8.13.3. EXAMPLE	52
8.13.4. SEE ALSO	52
8.14. COUNTER(1)	52
8.14.1. NAME	52
8.14.2. SYNOPSIS	53
8.14.3. EXAMPLE	53
8.14.4. SEE ALSO	53
8.15. CREATE(1)	53
8.15.1. NAME	53
8.15.2. SYNOPSIS	53
8.15.3. CREATE CACHE OPTIONS	53
8.15.4. CREATE COUNTER OPTIONS	53
8.15.5. EXAMPLES	54
8.15.6. SEE ALSO	54
8.16. CREDENTIALS(1)	54
8.16.1. NAME	54
8.16.2. SYNOPSIS	54
8.16.3. DESCRIPTION	54
8.16.4. SYNOPSIS	54
8.16.5. OPTIONS	54
8.16.6. CREDENTIALS ADD OPTIONS	55
8.16.7. CREDENTIALS MASK OPTIONS	55
8.16.8. EXAMPLES	55
8.17. DESCRIBE(1)	55
8.17.1. NAME	55

---

8.17.2. SYNOPSIS	56
8.17.3. EXAMPLES	56
8.17.4. SEE ALSO	56
8.18. DISCONNECT(1)	56
8.18.1. NAME	56
8.18.2. SYNOPSIS	56
8.18.3. EXAMPLE	56
8.18.4. SEE ALSO	56
8.19. DROP(1)	56
8.19.1. NAME	56
8.19.2. SYNOPSIS	56
8.19.3. EXAMPLES	57
8.19.4. SEE ALSO	57
8.20. ENCODING(1)	57
8.20.1. NAME	57
8.20.2. DESCRIPTION	57
8.20.3. SYNOPSIS	57
8.20.4. EXAMPLE	57
8.20.5. SEE ALSO	57
8.21. GET(1)	57
8.21.1. NAME	57
8.21.2. SYNOPSIS	58
8.21.3. OPTIONS	58
8.21.4. EXAMPLE	58
8.21.5. SEE ALSO	58
8.22. HELP(1)	58
8.22.1. NAME	58
8.22.2. SYNOPSIS	58
8.22.3. EXAMPLE	58
8.22.4. SEE ALSO	58
8.23. INDEX(1)	58
8.23.1. NAME	58
8.23.2. SYNOPSIS	58
8.23.3. EXAMPLES	59
8.23.4. SEE ALSO	59
8.24. INSTALL(1)	59
8.24.1. NAME	59
8.24.2. DESCRIPTION	59
8.24.3. SYNOPSIS	59
8.24.4. ARTIFACT NAMES	59
8.24.5. CHECKSUM VALIDATION	60
8.24.6. PATCH LIST OPTIONS	60
8.24.7. EXAMPLES	60
8.25. LOGGING(1)	60
8.25.1. NAME	60
8.25.2. SYNOPSIS	61
8.25.3. LOGGING SET OPTIONS	61
8.25.4. EXAMPLES	61
8.26. LS(1)	61
8.26.1. NAME	61
8.26.2. SYNOPSIS	61
8.26.3. OPTIONS	61
8.26.4. EXAMPLES	62



---

8.26.5. SEE ALSO	62
8.27. MIGRATE(1)	62
8.27.1. NAME	62
8.27.2. SYNOPSIS	62
8.27.3. DESCRIPTION	62
8.27.4. COMMAND SYNOPSIS	62
8.27.5. COMMON OPTIONS	63
8.27.6. CLUSTER CONNECTION OPTIONS	63
8.28. PATCH(1)	63
8.28.1. NAME	64
8.28.2. DESCRIPTION	64
8.28.3. SYNOPSIS	64
8.28.4. PATCH LIST OPTIONS	64
8.28.5. PATCH INSTALL OPTIONS	64
8.28.6. PATCH DESCRIBE OPTIONS	64
8.28.7. PATCH ROLLBACK OPTIONS	64
8.28.8. PATCH CREATE OPTIONS	64
8.28.9. EXAMPLES	65
8.29. PUT(1)	65
8.29.1. NAME	65
8.29.2. DESCRIPTION	65
8.29.3. SYNOPSIS	65
8.29.4. OPTIONS	65
8.29.5. EXAMPLES	66
8.29.6. SEE ALSO	66
8.30. QUERY(1)	66
8.30.1. NAME	66
8.30.2. SYNOPSIS	66
8.30.3. OPTIONS	66
8.30.4. EXAMPLES	66
8.30.5. SEE ALSO	66
8.31. QUIT(1)	66
8.31.1. NAME	67
8.31.2. SYNOPSIS	67
8.31.3. EXAMPLE	67
8.31.4. SEE ALSO	67
8.32. REBALANCE(1)	67
8.32.1. NAME	67
8.32.2. SYNOPSIS	67
8.32.3. EXAMPLES	67
8.33. REMOVE(1)	68
8.33.1. NAME	68
8.33.2. SYNOPSIS	68
8.33.3. OPTIONS	68
8.33.4. EXAMPLE	68
8.33.5. SEE ALSO	68
8.34. RESET(1)	68
8.34.1. NAME	68
8.34.2. SYNOPSIS	68
8.34.3. EXAMPLE	68
8.34.4. SEE ALSO	68
8.35. SCHEMA(1)	68
8.35.1. NAME	68

---

8.35.2. SYNOPSIS	68
8.35.3. DESCRIPTION	69
8.35.4. COMMAND SYNOPSIS	69
8.35.5. UPLOAD OPTIONS	69
8.35.6. EXAMPLE	69
8.35.7. SEE ALSO	69
8.36. SERVER(1)	69
8.36.1. NAME	69
8.36.2. DESCRIPTION	69
8.36.3. SYNOPSIS	70
8.36.4. SERVER CONNECTOR IPFILTER OPTIONS	70
8.36.5. EXAMPLES	70
8.37. SHUTDOWN(1)	71
8.37.1. NAME	71
8.37.2. SYNOPSIS	71
8.37.3. EXAMPLES	71
8.37.4. SEE ALSO	71
8.38. SITE(1)	71
8.38.1. NAME	71
8.38.2. SYNOPSIS	72
8.38.3. OPTIONS	72
8.38.4. STATE TRANSFER MODE OPTIONS	72
8.38.5. EXAMPLES	72
8.39. STATS(1)	73
8.39.1. NAME	73
8.39.2. SYNOPSIS	73
8.39.3. EXAMPLES	73
8.39.4. SEE ALSO	74
8.40. TASK(1)	74
8.40.1. NAME	74
8.40.2. SYNOPSIS	74
8.40.3. EXAMPLES	74
8.40.4. OPTIONS	74
8.40.5. SEE ALSO	74
8.41. UNALIAS(1)	74
8.41.1. NAME	74
8.41.2. SYNOPSIS	74
8.41.3. EXAMPLES	75
8.41.4. SEE ALSO	75
8.42. USER(1)	75
8.42.1. NAME	75
8.42.2. SYNOPSIS	75
8.42.3. DESCRIPTION	75
8.42.4. COMMAND SYNOPSIS	75
8.42.5. COMMON OPTIONS	76
8.42.6. USER CREATE/MODIFY OPTIONS	76
8.42.7. USER LS OPTIONS	77
8.42.8. USER ENCRYPT-ALL OPTIONS	77
8.42.9. USER ROLES OPTIONS	77
8.43. VERSION(1)	77
8.43.1. NAME	77
8.43.2. SYNOPSIS	77
8.43.3. EXAMPLE	77

8.43.4. SEE ALSO

77



# 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.

## DATA GRID DOCUMENTATION

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

- [Data Grid 8.4 Documentation](#)
- [Data Grid 8.4 Component Details](#)
- [Supported Configurations for Data Grid 8.4](#)
- [Data Grid 8 Feature Support](#)
- [Data Grid Deprecated Features and Functionality](#)

## 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.

## 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 1. GETTING STARTED WITH DATA GRID CLI

The command line interface (CLI) lets you remotely connect to Data Grid Server to access data and perform administrative functions. Complete the following procedures to learn basic CLI usage such as creating users, connecting to Data Grid, and navigating resources.

## 1.1. CREATING DATA GRID USERS

Add credentials to authenticate with Data Grid Server deployments through Hot Rod and REST endpoints. Before you can access the Data Grid Console or perform cache operations you must create at least one user with the Data Grid command line interface (CLI).

### TIP

Data Grid enforces security authorization with role-based access control (RBAC). Create an **admin** user the first time you add credentials to gain full **ADMIN** permissions to your Data Grid deployment.

### Prerequisites

- Download and install Data Grid Server.

### Procedure

1. Open a terminal in **\$RHDG\_HOME**.
2. Create an **admin** user with the **user create** command.

```
bin/cli.sh user create admin -p changeme
```

### TIP

Run **help user** from a CLI session to get complete command details.

### Verification

Open **user.properties** and confirm the user exists.

```
cat server/conf/users.properties
admin=scram-sha-1\:BYGclAwwf6b...
```



### NOTE

Adding credentials to a properties realm with the CLI creates the user only on the server instance to which you are connected. You must manually synchronize credentials in a properties realm to each node in the cluster.

### 1.1.1. Granting roles to users

Assign roles to users and grant them permissions to perform cache operations and interact with Data Grid resources.

## TIP

Grant roles to groups instead of users if you want to assign the same role to multiple users and centrally maintain their permissions.

## Prerequisites

- Have **ADMIN** permissions for Data Grid.
- Create Data Grid users.

## Procedure

1. Create a CLI connection to Data Grid.
2. Assign roles to users with the **user roles grant** command, for example:

```
user roles grant --roles=deployer katie
```

## Verification

List roles that you grant to users with the **user roles ls** command.

```
user roles ls katie  
["deployer"]
```

### 1.1.2. Adding users to groups

Groups let you change permissions for multiple users. You assign a role to a group and then add users to that group. Users inherit permissions from the group role.



#### NOTE

You use groups as part of a property realm in the Data Grid Server configuration. Each group is a special type of user that also requires a username and password.

## Prerequisites

- Have **ADMIN** permissions for Data Grid.
- Create Data Grid users.

## Procedure

1. Create a CLI connection to Data Grid.
2. Use the **user create** command to create a group.
  - a. Specify a group name with the **--groups** argument.
  - b. Set a username and password for the group.

```
user create --groups=developers developers -p changeme
```

3. List groups.

```
user ls --groups
```

4. Grant a role to the group.

```
user roles grant --roles=application developers
```

5. List roles for the group.

```
user roles ls developers
```

6. Add users to the group one at a time.

```
user groups john --groups=developers
```

## Verification

Open **groups.properties** and confirm the group exists.

```
cat server/conf/groups.properties
```

### 1.1.3. Data Grid user roles and permissions

Data Grid includes several roles that provide users with permissions to access caches and Data Grid resources.

Role	Permissions	Description
<b>admin</b>	ALL	Superuser with all permissions including control of the Cache Manager lifecycle.
<b>deployer</b>	ALL_READ, ALL_WRITE, LISTEN, EXEC, MONITOR, CREATE	Can create and delete Data Grid resources in addition to <b>application</b> permissions.
<b>application</b>	ALL_READ, ALL_WRITE, LISTEN, EXEC, MONITOR	Has read and write access to Data Grid resources in addition to <b>observer</b> permissions. Can also listen to events and execute server tasks and scripts.
<b>observer</b>	ALL_READ, MONITOR	Has read access to Data Grid resources in addition to <b>monitor</b> permissions.
<b>monitor</b>	MONITOR	Can view statistics via JMX and the <b>metrics</b> endpoint.

### Additional resources

- [org.infinispan.security.AuthorizationPermission Enum](#)
- [Data Grid configuration schema reference](#)

## 1.2. 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:**

```
bin/cli.sh
```
  - **Microsoft Windows:**

```
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
```

## 1.3. 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.

```
cd caches
```

- Use **describe** to view information about resources.

```
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
  } ]
}
```

### 1.3.1. 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
 tasks
```

### **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.

## **1.4. 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:

```
 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:

```
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
ISPN000390: Persisted state, version=<$version> timestamp=YYYY-MM-DDTHH:MM:SS
ISPN080003: Data Grid Server stopped
```

## 1.4.1. Shutdown and restart of Data Grid clusters

Prevent data loss and ensure consistency of your cluster by properly shutting down and restarting nodes.

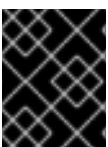
### Cluster shutdown

Data Grid recommends using the **shutdown cluster** command to stop all nodes in a cluster while saving cluster state and persisting all data in the cache. You can use the **shutdown cluster** command also for clusters with a single node.

When you bring Data Grid clusters back online, all nodes and caches in the cluster will be unavailable until all nodes rejoin. To prevent inconsistencies or data loss, Data Grid restricts access to the data stored in the cluster and modifications of the cluster state until the cluster is fully operational again. Additionally, Data Grid disables cluster rebalancing and prevents local cache stores purging on startup.

During the cluster recovery process, the coordinator node logs messages for each new node joining, indicating which nodes are available and which are still missing. Other nodes in the Data Grid cluster have the view from the time they join. You can monitor availability of caches using the Data Grid Console or REST API.

However, in cases where waiting for all nodes is not necessary nor desired, it is possible to set a cache available with the current topology. This approach is possible through the CLI, see below, or the REST API.



### IMPORTANT

Manually installing a topology can lead to data loss, only perform this operation if the initial topology cannot be recreated.

### Server shutdown

After using the **shutdown server** command to bring nodes down, the first node to come back online will be available immediately without waiting for other members. The remaining nodes join the cluster immediately, triggering state transfer but loading the local persistence first, which might lead to stale entries. Local cache stores configured to purge on startup will be emptied when the server starts. Local cache stores marked as **purge=false** will be available after a server restarts but might contain stale entries.

If you shutdown clustered nodes with the **shutdown server** command, you must restart each server in reverse order to avoid potential issues related to data loss and stale entries in the cache. For example, if you shutdown **server1** and then shutdown **server2**, you should first start **server2** and then start **server1**. However, restarting clustered nodes in reverse order does not completely prevent data loss and stale entries.



## CHAPTER 2. PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

Use the command line interface (CLI) to perform operations on remote caches such as creating caches, manipulating data, and rebalancing.

### 2.1. CREATING REMOTE CACHES WITH THE DATA GRID CLI

Use the Data Grid Command Line Interface (CLI) to add remote caches on Data Grid Server.

#### Prerequisites

- Create a Data Grid user with **admin** permissions.
- Start at least one Data Grid Server instance.
- Have a Data Grid cache configuration.

#### Procedure

1. Start the CLI.

```
bin/cli.sh
```

2. Run the **connect** command and enter your username and password when prompted.
3. Use the **create cache** command to create remote caches.  
For example, create a cache named "mycache" from a file named **mycache.xml** as follows:

```
create cache --file=mycache.xml mycache
```

#### Verification

1. List all remote caches with the **ls** command.

```
ls caches  
mycache
```

2. View cache configuration with the **describe** command.

```
describe caches/mycache
```

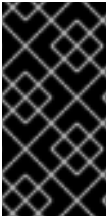
#### 2.1.1. Cache configuration

You can create declarative cache configuration in XML, JSON, and YAML format.

All declarative caches must conform to the Data Grid schema. Configuration in JSON format must follow the structure of an XML configuration, elements correspond to objects and attributes correspond to fields.

**IMPORTANT**

Data Grid restricts characters to a maximum of **255** for a cache name or a cache template name. If you exceed this character limit, Data Grid throws an exception. Write succinct cache names and cache template names.

**IMPORTANT**

A file system might set a limitation for the length of a file name, so ensure that a cache's name does not exceed this limitation. If a cache name exceeds a file system's naming limitation, general operations or initialing operations towards that cache might fail. Write succinct file names.

**Distributed caches****XML**

```
<distributed-cache owners="2"
  segments="256"
  capacity-factor="1.0"
  l1-lifespan="5000"
  mode="SYNC"
  statistics="true">
  <encoding media-type="application/x-protostream"/>
  <locking isolation="REPEATABLE_READ"/>
  <transaction mode="FULL_XA"
    locking="OPTIMISTIC"/>
  <expiration lifespan="5000"
    max-idle="1000" />
  <memory max-count="1000000"
    when-full="REMOVE"/>
  <indexing enabled="true"
    storage="local-heap">
    <index-reader refresh-interval="1000"/>
    <indexed-entities>
      <indexed-entity>org.infinispan.Person</indexed-entity>
    </indexed-entities>
  </indexing>
  <partition-handling when-split="ALLOW_READ_WRITES"
    merge-policy="PREFERRED_NON_NULL"/>
  <persistence passivation="false">
    <!-- Persistent storage configuration. -->
  </persistence>
</distributed-cache>
```

**JSON**

```
{
  "distributed-cache": {
    "mode": "SYNC",
    "owners": "2",
    "segments": "256",
    "capacity-factor": "1.0",
    "l1-lifespan": "5000",
```

```

"statistics": "true",
"encoding": {
  "media-type": "application/x-protostream"
},
"locking": {
  "isolation": "REPEATABLE_READ"
},
"transaction": {
  "mode": "FULL_XA",
  "locking": "OPTIMISTIC"
},
"expiration" : {
  "lifespan" : "5000",
  "max-idle" : "1000"
},
"memory": {
  "max-count": "1000000",
  "when-full": "REMOVE"
},
"indexing" : {
  "enabled" : true,
  "storage" : "local-heap",
  "index-reader" : {
    "refresh-interval" : "1000"
  },
  "indexed-entities": [
    "org.infinispan.Person"
  ]
},
"partition-handling" : {
  "when-split" : "ALLOW_READ_WRITES",
  "merge-policy" : "PREFERRED_NON_NULL"
},
"persistence" : {
  "passivation" : false
}
}
}

```

## YAML

```

distributedCache:
  mode: "SYNC"
  owners: "2"
  segments: "256"
  capacityFactor: "1.0"
  l1Lifespan: "5000"
  statistics: "true"
  encoding:
    mediaType: "application/x-protostream"
  locking:
    isolation: "REPEATABLE_READ"
  transaction:
    mode: "FULL_XA"
    locking: "OPTIMISTIC"

```

```

expiration:
  lifespan: "5000"
  maxIdle: "1000"
memory:
  maxCount: "1000000"
  whenFull: "REMOVE"
indexing:
  enabled: "true"
  storage: "local-heap"
  indexReader:
    refreshInterval: "1000"
  indexedEntities:
    - "org.infinispan.Person"
partitionHandling:
  whenSplit: "ALLOW_READ_WRITES"
  mergePolicy: "PREFERRED_NON_NULL"
persistence:
  passivation: "false"
  # Persistent storage configuration.

```

## Replicated caches

### XML

```

<replicated-cache segments="256"
  mode="SYNC"
  statistics="true">
  <encoding media-type="application/x-protostream"/>
  <locking isolation="REPEATABLE_READ"/>
  <transaction mode="FULL_XA"
    locking="OPTIMISTIC"/>
  <expiration lifespan="5000"
    max-idle="1000" />
  <memory max-count="1000000"
    when-full="REMOVE"/>
  <indexing enabled="true"
    storage="local-heap">
    <index-reader refresh-interval="1000"/>
    <indexed-entities>
      <indexed-entity>org.infinispan.Person</indexed-entity>
    </indexed-entities>
  </indexing>
  <partition-handling when-split="ALLOW_READ_WRITES"
    merge-policy="PREFERRED_NON_NULL"/>
  <persistence passivation="false">
    <!-- Persistent storage configuration. -->
  </persistence>
</replicated-cache>

```

### JSON

```

{
  "replicated-cache": {
    "mode": "SYNC",

```

```

"segments": "256",
"statistics": "true",
"encoding": {
  "media-type": "application/x-protostream"
},
"locking": {
  "isolation": "REPEATABLE_READ"
},
"transaction": {
  "mode": "FULL_XA",
  "locking": "OPTIMISTIC"
},
"expiration" : {
  "lifespan" : "5000",
  "max-idle" : "1000"
},
"memory": {
  "max-count": "1000000",
  "when-full": "REMOVE"
},
"indexing" : {
  "enabled" : true,
  "storage" : "local-heap",
  "index-reader" : {
    "refresh-interval" : "1000"
  },
  "indexed-entities": [
    "org.infinispan.Person"
  ]
},
"partition-handling" : {
  "when-split" : "ALLOW_READ_WRITES",
  "merge-policy" : "PREFERRED_NON_NULL"
},
"persistence" : {
  "passivation" : false
}
}
}

```

## YAML

```

replicatedCache:
  mode: "SYNC"
  segments: "256"
  statistics: "true"
  encoding:
    mediaType: "application/x-protostream"
  locking:
    isolation: "REPEATABLE_READ"
  transaction:
    mode: "FULL_XA"
    locking: "OPTIMISTIC"
  expiration:
    lifespan: "5000"

```

```

maxIdle: "1000"
memory:
  maxCount: "1000000"
  whenFull: "REMOVE"
indexing:
  enabled: "true"
  storage: "local-heap"
indexReader:
  refreshInterval: "1000"
indexedEntities:
  - "org.infinispan.Person"
partitionHandling:
  whenSplit: "ALLOW_READ_WRITES"
  mergePolicy: "PREFERRED_NON_NULL"
persistence:
  passivation: "false"
# Persistent storage configuration.

```

## Multiple caches

### XML

```

<infinispan
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:infinispan:config:14.0 https://infinispan.org/schemas/infinispan-config-14.0.xsd
                    urn:infinispan:server:14.0 https://infinispan.org/schemas/infinispan-server-14.0.xsd"
  xmlns="urn:infinispan:config:14.0"
  xmlns:server="urn:infinispan:server:14.0">
  <cache-container name="default"
    statistics="true">
    <distributed-cache name="mycacheone"
      mode="ASYNC"
      statistics="true">
      <encoding media-type="application/x-protostream"/>
      <expiration lifespan="300000"/>
      <memory max-size="400MB"
        when-full="REMOVE"/>
    </distributed-cache>
    <distributed-cache name="mycachetwo"
      mode="SYNC"
      statistics="true">
      <encoding media-type="application/x-protostream"/>
      <expiration lifespan="300000"/>
      <memory max-size="400MB"
        when-full="REMOVE"/>
    </distributed-cache>
  </cache-container>
</infinispan>

```

### JSON

```

{
  "infinispan" : {

```

```

"cache-container" : {
  "name" : "default",
  "statistics" : "true",
  "caches" : {
    "mycacheone" : {
      "distributed-cache" : {
        "mode": "ASYNC",
        "statistics": "true",
        "encoding": {
          "media-type": "application/x-protostream"
        },
        "expiration" : {
          "lifespan" : "300000"
        },
        "memory": {
          "max-size": "400MB",
          "when-full": "REMOVE"
        }
      }
    },
    "mycachetwo" : {
      "distributed-cache" : {
        "mode": "SYNC",
        "statistics": "true",
        "encoding": {
          "media-type": "application/x-protostream"
        },
        "expiration" : {
          "lifespan" : "300000"
        },
        "memory": {
          "max-size": "400MB",
          "when-full": "REMOVE"
        }
      }
    }
  }
}

```

## YAML

```

infinispan:
  cacheContainer:
    name: "default"
    statistics: "true"
  caches:
    mycacheone:
      distributedCache:
        mode: "ASYNC"
        statistics: "true"
        encoding:
          mediaType: "application/x-protostream"
        expiration:

```

```

    lifespan: "300000"
    memory:
      maxSize: "400MB"
      whenFull: "REMOVE"
  mycachetwo:
    distributedCache:
      mode: "SYNC"
      statistics: "true"
      encoding:
        mediaType: "application/x-protostream"
      expiration:
        lifespan: "300000"
      memory:
        maxSize: "400MB"
        whenFull: "REMOVE"

```

### Additional resources

- [Data Grid configuration schema reference](#)
- [infinispan-config-14.0.xsd](#)

## 2.2. MODIFYING DATA GRID CACHE CONFIGURATION

Make changes to your remote cache configuration with the Data Grid CLI. You can modify attributes in your cache configuration either one at a time or provide a cache configuration in XML, JSON or YAML format to modify several attributes at once.

### Prerequisites

- Create at least one remote cache on your Data Grid cluster.

### Procedure

1. Create a CLI connection to Data Grid.
2. Modify the cache configuration with the **alter** command in one of the following ways:
  - Use the **--file** option to specify a configuration file with one or more attribute modifications.
  - Use the **--attribute** and **--value** option to modify a specific configuration attribute.

### TIP

For more information and examples, run the **help alter** command.

3. Verify your changes with the **describe** command, for example:

```
describe caches/mycache
```

## 2.3. 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 **--cache=** with the **put** command:

```
put --cache=mycache hello world
```

- Use the **put** command from the context of a cache:

```
[/containers/default/caches/mycache]> put hello world
```

3. Use the **get** command to verify entries.

```
[/containers/default/caches/mycache]> get hello  
world
```

## 2.4. 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.

```
clearcache mycache
```

- Remove specific entries with the **remove** command.

```
remove --cache=mycache hello
```

## 2.5. 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.

```
drop cache mycache
```

## 2.6. CONFIGURING AUTOMATIC CACHE REBALANCING

By default, Data Grid automatically rebalances caches as nodes join and leave the cluster. You can configure automatic cache rebalancing by disabling or enabling it at the Cache Manager level or on a per-cache basis.

### Procedure

1. Create a CLI connection to Data Grid.
2. Disable automatic rebalancing for all caches with the **rebalance disable** command.

```
rebalance disable
```

3. Enable automatic rebalancing for a specific cache with the **rebalance enable** command. The following example enables rebalancing for the cache named "mycache" only.

```
rebalance enable caches/mycache
```

4. Re-enable automatic rebalancing for all caches.

```
rebalance enable
```

For more information about the **rebalance** command, run **help rebalance**.

## 2.7. SET A STABLE TOPOLOGY

By default, after a cluster shutdown, Data Grid waits for all nodes to join the cluster and restore the topology. However, we offer a CLI command to mark the current topology stable for a specific cache.

### Procedure

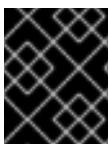
1. Create a CLI connection to Data Grid.
2. Do one of the following:
  - Set the current topology as stable for the given cache.

```
topology set-stable cacheName
```

- If the current topology is missing more nodes than the number of owners, the force flag is necessary to confirm the operation.

```
topology set-stable cacheName -f
```

For more information about the **topology set-stable** command, run **topology set-stable -h**.



### IMPORTANT

Manually installing a topology can lead to data loss, only perform this operation if the initial topology cannot be recreated.

## CHAPTER 3. PERFORMING BATCH OPERATIONS

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

### Prerequisites

- A running Data Grid cluster.

### 3.1. 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.

```
connect --username=<username> --password=<password> <hostname>:11222
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
ls caches/mybatch
disconnect
```

#### TIP

Configure the CLI with the **autoconnect-url** property instead of using the **connect** command directly in your batch files.

2. Run the CLI and specify the file as input.

```
bin/cli.sh -f batch
```



#### NOTE

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

### 3.2. 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 -
```

**TIP**

You can configure the CLI connection with the **autoconnect-url** property instead of using the **-c** argument.

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
```

## CHAPTER 4. CONFIGURING THE DATA GRID CLI

Define configuration properties for the Data Grid CLI.

### 4.1. 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. For example, set the **autoconnect-url** property so that the CLI automatically connects to that URL.



#### NOTE

For remote connections, specify the URL and provide credentials:

- **http[s]://<username>:<password>@<hostname>:<port>** for basic authentication.
- **http[s]://<token>@<hostname>:<port>** for OAuth authentication.

```
bin/cli.sh config set autoconnect-url http://<username>:<password>@<hostname>:11222
```

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

#### TIP

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

### 4.2. 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:

```
alias q=quit
```

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

```
alias  
alias q='quit'
```

3. Delete aliases with the **unalias** command, for example:

```
unalias q
```

### 4.3. 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. *Optional:* Define a trust store password. The following example sets **secret** as the trust store password:

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

3. *Optional:* If you require client certificate authentication for your server, specify the location of the client key store. Considering the following example, replace **<path>** with the absolute directory path to your key store file, and replace **<key\_store\_file>** with the name of your key store file:

```
bin/cli.sh config set keystore /<emphasis><path></emphasis>/<emphasis><key_store_file></emphasis>
```

4. *Optional:* Define a key store password. The following example sets **secret** as the key store password:

```
bin/cli.sh config set keystore-password secret
```

5. Verify your CLI configuration.

```
bin/cli.sh config get truststore
```

```
bin/cli.sh config get truststore-password
```

#### Additional resources

- [Setting Up SSL Identities for Data Grid Server](#)

## 4.4. DATA GRID CLI STORAGE DIRECTORY

Data Grid CLI stores configuration in the following default directory:

Operating System	Default Path
Linux/Unix	<code>\$HOME/.config/red_hat_data_grid</code>
Microsoft Windows	<code>%APPDATA%/Sun/Java/red_hat_data_grid</code>
Mac OS	<code>\$HOME/Library/Java/red_hat_data_grid</code>

This directory contains the following files:

#### **cli.properties**

Stores values for CLI configuration properties.

#### **aliases**

Stores command aliases.

#### **history**

Stores CLI history.

## CHAPTER 5. 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.

### 5.1. 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**.

```
create counter --concurrency-level=1 --initial-value=5 --storage=PERSISTENT --  
type=weak my-weak-counter
```

- b. Create **my-strong-counter**.

```
create counter --initial-value=3 --storage=PERSISTENT --type=strong my-strong-  
counter
```

3. List available counters.

```
ls counters
```

4. Verify counter configurations.

- a. Describe **my-weak-counter**.

```
describe counters/my-weak-counter
```

```
{  
  "weak-counter":{  
    "initial-value":5,  
    "storage":"PERSISTENT",  
    "concurrency-level":1  
  }  
}
```

- b. Describe **my-strong-counter**.

```
describe counters/my-strong-counter
```



```
{
  "strong-counter":{
    "initial-value":3,
    "storage":"PERSISTENT",
    "upper-bound":5
  }
}
```

## 5.2. ADDING DELTAS TO COUNTERS

Increment or decrement counters with arbitrary values.

### Procedure

1. Select a counter.

```
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, **add --delta=3 --quiet=true**.

Weak counters return empty responses.

## CHAPTER 6. 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.

### 6.1. BRINGING BACKUP LOCATIONS OFFLINE AND ONLINE

Take backup locations offline manually and bring them back online.

### Prerequisites

- Create a CLI connection to Data Grid.

### Procedure

1. Check if backup locations are online or offline with the **site status** command:

```
site status --cache=cacheName --site=NYC
```



#### NOTE

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

#### TIP

Use the **--all-caches** option to get the backup location status for all caches.

2. Manage backup locations as follows:

- Bring backup locations online with the **bring-online** command:

```
site bring-online --cache=customers --site=NYC
```

- Take backup locations offline with the **take-offline** command:

```
site take-offline --cache=customers --site=NYC
```

#### TIP

Use the **--all-caches** option to bring a backup location online, or take a backup location offline, for all caches.

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

## 6.2. CONFIGURING CROSS-SITE STATE TRANSFER MODES

You can configure cross-site state transfer operations to happen automatically when Data Grid detects that backup locations come online. Alternatively you can use the default mode, which is to manually perform state transfer.

### Prerequisites

- Create a CLI connection to Data Grid.

### Procedure

1. Use the **site** command to configure state transfer modes, as in the following examples:

- Retrieve the current state transfer mode.

```
site state-transfer-mode get --cache=cacheName --site=NYC
```

- Configure automatic state transfer operations for a cache and backup location.

```
site state-transfer-mode set --cache=cacheName --site=NYC --mode=AUTO
```

### TIP

Run the **help site** command for more information and examples.

## 6.3. PUSHING STATE TO BACKUP LOCATIONS

Transfer cache state to backup locations.

### Prerequisites

- Create a CLI connection to Data Grid.

### Procedure

- Use the **site push-site-state** command to push state transfer, as in the following example:

```
site push-site-state --cache=cacheName --site=NYC
```

### TIP

Use the **--all-caches** option to push state transfer for all caches.

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

## CHAPTER 7. BACKING UP AND RESTORING DATA GRID CLUSTERS

Create archives of Data Grid resources that include cached entries, cache configurations, Protobuf schemas, and server scripts. You can then use the backup archives to restore Data Grid Server clusters after a restart or migration.

### Prerequisites

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

## 7.1. BACKING UP DATA GRID CLUSTERS

Create backup archives in **.zip** format that you can download or store on Data Grid Server.

### Prerequisites

Backup archives should reflect the most recent cluster state. For this reason you should ensure the cluster is no longer accepting write requests before you create backup archives.

### Procedure

1. Create a CLI connection to Data Grid.
2. Run the **backup create** command with the appropriate options, for example:

- Back up all resources with an automatically generated name.

```
backup create
```

- Back up all resources in a backup archive named **example-backup**.

```
backup create -n example-backup
```

- Back up all resources to the **/some/server/dir** path on the server.

```
backup create -d /some/server/dir
```

- Back up only caches and cache templates.

```
backup create --caches=* --templates=*
```

- Back up named Protobuf schemas only.

```
backup create --proto-schemas=schema1,schema2
```

3. List available backup archives on the server.

```
backup ls
```

4. Download the **example-backup** archive from the server.  
If the backup operation is still in progress, the command waits for it to complete.

```
backup get example-backup
```

5. Optionally delete the **example-backup** archive from the server.

```
backup delete example-backup
```

## 7.2. RESTORING DATA GRID CLUSTERS FROM BACKUP ARCHIVES

Apply the content of backup archives to Data Grid clusters to restore them to the backed up state.

### Prerequisites

- Create a backup archive that is either local to the Data Grid CLI or stored on Data Grid Server.
- Ensure that the target container matches the container name in the backup archive. You cannot restore backups if the container names do not match.

### Procedure

1. Create a CLI connection to Data Grid.
2. Run the **backup restore** command with the appropriate options.
  - Restore all content from a backup archive accessible on the server.

```
backup restore /some/path/on/the/server
```

- Restore all content from a local backup archive.

```
backup restore -u /some/local/path
```

- Restore only cache content from a backup archive on the server.

```
backup restore /some/path/on/the/server --caches=*
```

## CHAPTER 8. 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
```

### 8.1. ADD(1)

#### 8.1.1. NAME

**add** - increments and decrements counters with arbitrary values.

#### 8.1.2. SYNOPSIS

```
add ['OPTIONS'] ['COUNTER_NAME']
```

#### 8.1.3. 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**.

#### 8.1.4. 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**.

#### 8.1.5. SEE ALSO

`cas(1)`, `reset(1)`

### 8.2. ALIAS(1)

#### 8.2.1. NAME

**alias** - creates or displays aliases.

#### 8.2.2. SYNOPSIS

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

### 8.2.3. EXAMPLES

#### **alias q=quit**

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

#### **alias**

Lists all defined aliases.

### 8.2.4. SEE ALSO

config(1), unalias(1)

## 8.3. ALTER(1)

### 8.3.1. NAME

alter - modifies the configuration of caches on Data Grid Server.

### 8.3.2. SYNOPSIS

**alter cache** [**'OPTIONS'**] **CACHE\_NAME**

You can modify a cache with the **alter** command only if the changes are compatible with the existing configuration.

For example you cannot use a replicated cache configuration to modify a distributed cache. Likewise if you create a cache configuration with a specific attribute, you cannot modify the configuration to use a different attribute instead. For example, attempting to modify cache configuration by specifying a value for the **max-count** attribute results in invalid configuration if the **max-size** is already set.

### 8.3.3. ALTER CACHE OPTIONS

**-f, --file='FILE'**

Specifies a configuration file in XML, JSON or YAML format that modifies an existing configuration. Mutually exclusive with the **--attribute** option.

**--attribute='ATTRIBUTE'**

Specifies an attribute to modify in an existing configuration. Press the tab key to display a list of attributes. Must be used in combination with the **--value** option. Mutually exclusive with the **--file** option.

**--value='VALUE'**

Specifies the new value for a configuration attribute. Must be used in combination with the **--attribute** option.

### 8.3.4. EXAMPLES

**alter cache mycache --file=/path/to/mycache.json**

Modifies the configuration of a cache named **mycache** with the **mycache.json** file.

**alter cache mycache --attribute=clustering.remote-timeout --value=5000**

Modifies the configuration of a cache named **mycache** so that the **clustering.remote-timeout** attribute has a value of '5000'.

### 8.3.5. SEE ALSO

`create(1)`, `drop(1)`

## 8.4. AVAILABILITY(1)

### 8.4.1. NAME

`availability` - manage availability of clustered caches in network partitions.

### 8.4.2. SYNOPSIS

`availability` ['OPTIONS'] ['CACHE\_NAME']

### 8.4.3. OPTIONS

`--mode='[AVAILABLE|DEGRADED_MODE]'`

Sets cache availability to `AVAILABLE` or `DEGRADED_MODE` when using either the `DENY_READ_WRITES` or `ALLOW_READS` partition handling strategy.

`AVAILABLE` makes caches available to all nodes in a network partition. `DEGRADED_MODE` prevents read and write operations on caches when network partitions occur.

### 8.4.4. EXAMPLES

#### `availability cache1`

Gets the current availability of the cache 'cache1'.

#### `availability --mode=AVAILABLE cache1`

Sets the availability of the cache 'cache1' to `AVAILABLE`.

## 8.5. BACKUP(1)

### 8.5.1. NAME

`backup` - manage container backup creation and restoration.

### 8.5.2. SYNOPSIS

`backup create` ['OPTIONS']

`backup delete` ['OPTIONS'] **BACKUP\_NAME**

`backup get` ['OPTIONS'] **BACKUP\_NAME**

`backup ls`

`backup restore` ['OPTIONS'] **BACKUP\_PATH**

### 8.5.3. BACKUP CREATE OPTIONS

`-d, --dir='PATH'`



Specifies a directory on the server to create and store the backup archive.

**-n, --name='NAME'**

Defines a name for the backup archive.

**--caches='cache1,cache2,...'**

Lists caches to back up. Use '\*' to back up all caches.

**--templates='template1,template2,...'**

Lists cache templates to back up. Use '\*' to back up all templates.

**--counters='counter1,counter2,...'**

Lists of counters to back up. Use '\*' to back up all counters.

**--proto-schemas='schema1,schema2,...'**

Lists Protobuf schemas to back up. Use '\*' to back up all schemas.

**--tasks='task1,task2,...'**

Lists server tasks to back up. Use '\*' to back up all tasks.

## 8.5.4. BACKUP GET OPTIONS

**--no-content**

Does not download content. The command returns only when the backup operation is complete.

## 8.5.5. BACKUP RESTORE OPTIONS

**-u, --upload**

Defines the path to a local backup archive that is uploaded to the server.

**-n, --name='NAME'**

Defines a name for the restore request.

**--caches='cache1,cache2,...'**

Lists caches to restore. Use '\*' to restore all caches from the backup archive.

**--templates='template1,template2,...'**

Lists cache templates to restore. Use '\*' to restore all templates from the backup archive.

**--counters='counter1,counter2,...'**

Lists counters to restore. Use '\*' to restore all counters from the backup archive.

**--proto-schemas='schema1,schema2,...'**

Lists Protobuf schemas to restore. Use '\*' to restore all schemas from the backup archive.

**--tasks='task1,task2,...'**

Lists server tasks to restore. Use '\*' to restore all tasks from the backup archive.

## 8.5.6. EXAMPLES

**backup create -n example-backup**

Initiates a backup of all container content with name **example-backup**.

**backup create -d /some/server/dir**

Initiates a backup of all container content and stores it on the server at path **/some/server/dir**.

**backup create --caches=\* --templates=\***

Initiates a backup that contains only cache and cache configuration resources.

**backup create --proto-schemas=schema1,schema2**

Initiates a backup that contains the named schema resources only.

**backup ls**

Lists all backups available on the server.

**backup get example-backup**

Downloads the **example-backup** archive from the server. If the backup operation is in progress, the command waits for it to complete.

**backup restore /some/path/on/the/server**

Restores all content from a backup archive on the server.

**backup restore -u /some/local/path**

Restores all content from a local backup archive that is uploaded to the server.

**backup restore /some/path/on/the/server --caches=\***

Restores only cache content from a backup archive on the server.

**backup restore /some/path/on/the/server --proto-schemas=schema1,schema2**

Restores only the named schema resources from a backup archive on the server.

**backup delete example-backup**

Deletes the **example-backup** archive from the server.

## 8.5.7. SEE ALSO

drop(1)

## 8.6. BENCHMARK(1)

### 8.6.1. NAME

benchmark - runs a performance benchmark against a cache.

You can run performance benchmarks for the following HTTP and Hot Rod protocols: **http**, **https**, **hotrod**, and **hotrods**. You specify the protocol for the benchmark with a URI. If you do not specify a protocol, the benchmark uses the URI of the current CLI connection.

Benchmarks for Hot Rod URIs connect to the entire cluster. For HTTP URIs, benchmarks connect to a single node only.

Benchmarks test performance against an existing cache. Before you run a benchmark, you should create a cache with the capabilities you want to measure. For example, if you want to evaluate the performance of cross-site replication, you should create a cache that has backup locations. If you want to test the performance of persistence, create a cache that uses an appropriate cache store.

### 8.6.2. SYNOPSIS

```
benchmark ['OPTIONS'] [uri]
```

### 8.6.3. BENCHMARK OPTIONS

```
-t, --threads='num'
```

Specifies the number of threads to create. Defaults to **10**.

**--cache='cache'**

Names the cache against which the benchmark is performed. Defaults to **benchmark**. You must create the cache before running the benchmark if it does not already exist.

**\*--key-size='num'**

Sets the size, in bytes, of the key. Defaults to 16 bytes.

**\*--value-size='num'**

Sets the size, in bytes, of the value. Defaults to 1000 bytes.

**\*--keyset-size='num'**

Defines the size, in bytes, of the test key set. Defaults to **1000**.

**--verbosity=['SILENT', 'NORMAL', 'EXTRA']**

Specifies the verbosity level of the output. Possible values, from least to most verbose, are **SILENT**, **NORMAL**, and **EXTRA**. The default is **NORMAL**.

**-c, --count='num'**

Specifies how many measurement iterations to perform. Defaults to **5**.

**--time='time'**

Sets the amount of time, in seconds, that each iteration takes. Defaults to **10**.

**--warmup-count='num'**

Specifies how many warmup iterations to perform. Defaults to **5**.

**--warmup-time='time'**

Sets the amount of time, in seconds, that each warmup iteration takes. Defaults to **1**.

**--mode='mode'**

Specifies the benchmark mode. Possible values are **Throughput**, **AverageTime**, **SampleTime**, **SingleShotTime**, and **All**. The default is **Throughput**.

**--time-unit='unit'**

Specifies the time unit for results in the benchmark report. Possible values are **NANOSECONDS**, **MICROSECONDS**, **MILLISECONDS**, and **SECONDS**. The default is **MICROSECONDS**.

## 8.6.4. EXAMPLES

**benchmark hotrod://localhost:11222**

Performs a benchmark test with the Hot Rod protocol.

**benchmark --value-size=10000 --cache=largecache hotrod://localhost:11222**

Performs a benchmark test with the Hot Rod protocol against the **largecache** cache using test values that are 10000 bytes in size.

**benchmark --mode=All --threads=20 https://user:password@server:11222**

Performs a benchmark test with the HTTPS protocol using 20 threads and includes all modes in the report.

## 8.7. CACHE(1)

### 8.7.1. NAME

cache - selects the default cache for subsequent commands.

## 8.7.2. SYNOPSIS

```
cache ['CACHE_NAME']
```

## 8.7.3. EXAMPLE

### **cache mycache**

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

## 8.7.4. SEE ALSO

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

## 8.8. CAS(1)

### 8.8.1. NAME

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

### 8.8.2. SYNOPSIS

```
cas ['OPTIONS'] ['COUNTER_NAME']
```

### 8.8.3. 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.

### 8.8.4. EXAMPLE

```
cas --expect=10 --value=20 cnt_a
```

Sets the value of **cnt\_a** to **20** only if the current value is **10**

### 8.8.5. SEE ALSO

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

## 8.9. CD(1)

### 8.9.1. NAME

cd - navigates the server resource tree.

### 8.9.2. DESCRIPTION

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

### 8.9.3. SYNOPSIS

```
cd ['PATH']
```

### 8.9.4. EXAMPLE

#### **cd caches**

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

### 8.9.5. SEE ALSO

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

## 8.10. CLEARCACHE(1)

### 8.10.1. NAME

clearcache - removes all entries from a cache.

### 8.10.2. SYNOPSIS

```
clearcache ['CACHE_NAME']
```

### 8.10.3. EXAMPLES

#### **clearcache mycache**

Removes all entries from **mycache**.

### 8.10.4. SEE ALSO

cache(1), drop(1), remove(1)

## 8.11. CONFIG(1)

### 8.11.1. NAME

config - manages CLI configuration properties.

### 8.11.2. SYNOPSIS

```
config
```

```
config set 'name' 'value'
```

```
config get 'name'
```

```
config convert --outputFormat=[xml|json|yaml] [-o outputFile] [inputFile]
```

### 8.11.3. DESCRIPTION

Manage (list, set, get) CLI configuration properties and provide configuration conversion between the different formats (XML, JSON, YAML)

### 8.11.4. 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.

#### **config reset**

Resets all properties to their default values.

#### **config convert --format=[xml|json|yaml] [-o outputFile] [inputFile]**

Converts a configuration file to a different format.

### 8.11.5. COMMON OPTIONS

These options apply to all commands:

#### **-h, --help**

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

### 8.11.6. CONVERT OPTIONS

The following options apply to the **convert** command:

#### **-f, --format='xml|json|yaml'**

Specifies the format for the conversion.

#### **-o, --output='path'**

Specifies the path to the output file. Uses standard output (**stdout**) if you do not specify a path.

### 8.11.7. 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 truststore.

#### **keystore**

Defines a path to the keystore, which contains a certificate. The certificate identifies the client. Use the **keystore** property when the server requires client certificate authentication.

#### **keystore-password**

Specifies a password to access the keystore.

### 8.11.8. 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.

**config convert -f yaml -o infinispam.yaml infinispam.xml**

Converts the **infinispam.xml** file to YAML and writes the output to the **infinispam.yaml** file.

**config convert -f json**

Converts the configuration from standard input to JSON, and writes the output to standard output.

### 8.11.9. SEE ALSO

[alias\(1\)](#), [unalias\(1\)](#)

## 8.12. CONNECT(1)

### 8.12.1. NAME

**connect** - connects to running Data Grid servers.

### 8.12.2. DESCRIPTION

Defaults to <http://localhost:11222> and prompts for credentials if authentication is required.

### 8.12.3. SYNOPSIS

```
connect ['OPTIONS'] ['SERVER_LOCATION']
```

### 8.12.4. OPTIONS

**-u, --username='USERNAME'**

Specifies a username to authenticate with Data Grid servers.

**-p, --password='PASSWORD'**

Specifies passwords.

**-t, --truststore='PATH'**

Specifies a truststore.

**-s, --truststore-password='PASSWORD'**

Specifies a password for the truststore.

**-k, --keystore='PATH'**

Specifies a keystore that contains a client certificate.

**-w, --keystore-password='PASSWORD'**

Specifies a password for the keystore.

**--hostname-verifier='REGEX'**

A regular expression that matches hostnames during a connection to an SSL/TLS-enabled server.

**--trustall**

Trusts all certificates.

**--context-path='PATH'**

The context path for the server REST connector. If unspecified, defaults to **/rest**.

### 8.12.5. 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.

### 8.12.6. SEE ALSO

disconnect(1)

## 8.13. CONTAINER(1)

### 8.13.1. NAME

container - selects the container for running subsequent commands.

### 8.13.2. SYNOPSIS

container ['CONTAINER\_NAME']

### 8.13.3. EXAMPLE

**container default**

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

### 8.13.4. SEE ALSO

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

## 8.14. COUNTER(1)

### 8.14.1. NAME

counter - selects the default counter for subsequent commands.



## 8.14.2. SYNOPSIS

```
counter ['COUNTER_NAME']
```

## 8.14.3. EXAMPLE

**counter cnt\_a**

Selects **cnt\_a** and is the same as navigating the resource tree using **cd counters/cnt\_a**.

## 8.14.4. SEE ALSO

add(1), cas(1)

## 8.15. CREATE(1)

### 8.15.1. NAME

create - creates caches and counters on Data Grid servers.

### 8.15.2. SYNOPSIS

```
create cache ['OPTIONS'] CACHE_NAME
```

```
create counter ['OPTIONS'] COUNTER_NAME
```

### 8.15.3. CREATE CACHE OPTIONS

**-f, --file='FILE'**

Specifies a configuration file in XML, JSON or YAML format.

**-t, --template='TEMPLATE'**

Specifies a configuration template. Use tab autocompletion to see available templates.

**-v, --volatile='[true|false]'**

Specifies whether the cache is persistent or volatile. The default is false.

### 8.15.4. CREATE COUNTER OPTIONS

**-t, --type='[weak|strong]'**

Specifies if the counter is weak or strong.

**-s, --storage='[PERSISTENT|VOLATILE]'**

Specifies whether the counter is persistent or volatile.

**-c, --concurrency-level='nnn'**

Sets the concurrency level of the counter.

**-i, --initial-value='nnn'**

Sets the initial value of the counter.

**-l, --lower-bound='nnn'**

Sets the lower bound of a **strong** counter.

**-u, --upper-bound='nnn'**

Sets the upper bound of a **strong** counter.

### 8.15.5. EXAMPLES

**create cache --template=org.infinispan.DIST\_SYNC mycache**

Creates a cache named **mycache** from the **DIST\_SYNC** template.

**create counter --initial-value=3 --storage=PERSISTENT --type=strong cnt\_a**

Creates a strong counter named **cnt\_a**.

### 8.15.6. SEE ALSO

drop(1)

## 8.16. CREDENTIALS(1)

### 8.16.1. NAME

credentials - manages keystores that contain Data Grid Server credentials

### 8.16.2. SYNOPSIS

**credentials ls**

**credentials add 'alias'**

**credentials remove 'alias'**

**credentials mask -i iterations -s salt secret**

### 8.16.3. DESCRIPTION

List, create, and remove credentials inside a keystore and mask keystore passwords. By default, commands manage the **credentials.pfx** keystore in the server configuration directory.

### 8.16.4. SYNOPSIS

**credentials ls**

Lists credential aliases stored in the keystore.

Add a credential

**credentials add 'alias'**

Adds an alias and corresponding credential to the keystore.

Remove a credential

**credentials remove 'alias'**

Deletes an alias and corresponding credential from the keystore.

**credentials mask -i iterations -s salt 'secret'**

Obscure the keystore password with a mask for additional security.

### 8.16.5. OPTIONS

**-h, --help**

Prints command help.

**-s, --server-root='path-to-server-root'**

Specifies the path to the server root directory. Defaults to **server**.

**--path='credentials.pfx'**

Specifies the path to the credential keystore. Defaults to the server configuration directory, **server/conf**.

**-p, --password='password'**

Specifies a password for the credential keystore.

**-t, --type='PKCS12'**

Specifies the type of keystore that contains credentials. Supported types are **PKCS12** or **JCEKS**. Defaults to **PKCS12**.

### 8.16.6. CREDENTIALS ADD OPTIONS

**-c, --credential='credential'**

Specifies the credential to store.

### 8.16.7. CREDENTIALS MASK OPTIONS

**-i, --iteration='n'**

Sets the number of iterations.

**-s, --salt='salt'**

Sets the salt and must be of length 8.

### 8.16.8. EXAMPLES

**credentials add dbpassword -c changeme -p "secret1234!"**

Creates a new default credential keystore, if does not already exist, and adds an alias of "dbpassword" for a password of "changeme". This command also sets "secret1234!" as the password for the credential keystore, which must match the password in the server configuration: **<clear-text-credential clear-text="secret1234!"/>**

**credentials ls -p "secret1234!"**

Lists all aliases in the default credential keystore.

**credentials add ldappassword -t JCEKS -p "secret1234!"**

Creates a credential keystore in JCEKS format and adds an alias "ldappassword". This command prompts you to specify the password that corresponds to the alias.

**credentials mask "secret1234!" -i 100 -s pepper99** Creates a masked representation of the credential "secret1234!" using 100 iterations using the string **pepper99** as salt.

## 8.17. DESCRIBE(1)

### 8.17.1. NAME

describe - displays information about resources.

## 8.17.2. SYNOPSIS

**describe** ['PATH']

## 8.17.3. 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.

## 8.17.4. SEE ALSO

cd(1), ls(1)

## 8.18. DISCONNECT(1)

### 8.18.1. NAME

disconnect - ends CLI sessions with Data Grid servers.

### 8.18.2. SYNOPSIS

**disconnect**

### 8.18.3. EXAMPLE

**disconnect**

Ends the current CLI session.

### 8.18.4. SEE ALSO

connect(1)

## 8.19. DROP(1)

### 8.19.1. NAME

drop - deletes caches and counters.

### 8.19.2. SYNOPSIS

**drop cache CACHE\_NAME**

**drop counter COUNTER\_NAME**

### 8.19.3. EXAMPLES

#### **drop cache mycache**

Deletes the **mycache** cache.

#### **drop counter cnt\_a**

Deletes the **cnt\_a** counter.

### 8.19.4. SEE ALSO

create(1), clearcache(1)

## 8.20. ENCODING(1)

### 8.20.1. NAME

encoding - displays and sets the encoding for cache entries.

### 8.20.2. 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**

### 8.20.3. SYNOPSIS

```
encoding ['ENCODING']
```

### 8.20.4. EXAMPLE

#### **encoding application/json**

Configures the currently selected cache to encode entries as **application/json**.

### 8.20.5. SEE ALSO

get(1), put(1)

## 8.21. GET(1)

### 8.21.1. NAME

get - retrieves entries from a cache.

## 8.21.2. SYNOPSIS

**get** ['OPTIONS'] **KEY**

## 8.21.3. OPTIONS

**-c, --cache='NAME'**

Specifies the cache from which to retrieve entries. Defaults to the currently selected cache.

## 8.21.4. EXAMPLE

**get hello -c mycache**

Retrieves the value of the key named **hello** from **mycache**.

## 8.21.5. SEE ALSO

query(1), put(1)

## 8.22. HELP(1)

### 8.22.1. NAME

help - prints manual pages for commands.

### 8.22.2. SYNOPSIS

**help** ['COMMAND']

### 8.22.3. EXAMPLE

**help get**

Prints the manual page for the **get** command.

### 8.22.4. SEE ALSO

version(1)

## 8.23. INDEX(1)

### 8.23.1. NAME

index - manages cache indexes.

### 8.23.2. SYNOPSIS

**index reindex** 'cache-name'

**index clear** 'cache-name'

**index update-schema** 'cache-name'

**index stats** 'cache-name'

**index clear-stats** 'cache-name'

### 8.23.3. EXAMPLES

#### **index reindex mycache**

Reindexes a cache.

#### **index clear mycache**

Clears a cache index.

#### **index update-schema mycache**

Updates the index schema for a cache.

#### **index stats mycache**

Shows indexing and search statistics for a cache.

#### **index clear-stats mycache**

Clears indexing and search statistics for a cache.

### 8.23.4. SEE ALSO

query(1)

## 8.24. INSTALL(1)

### 8.24.1. NAME

install - download and install artifacts for Data Grid Server.

### 8.24.2. DESCRIPTION

Download and install artifacts to the **server/lib** directory. You can specify the download location for artifacts as Maven artifact coordinates, a URL, or a local file path.

When downloading Maven artifacts, an optional Maven **settings.xml** file determines the location of the remote and local repositories as well as credentials and proxy configuration.

If you download artifacts as **zip**, **tar.gz**, or **tgz** archives, the content is extracted.

### 8.24.3. SYNOPSIS

```
install 'artifact-1[[[algorithm]]checksum]' ['artifact-2[[[algorithm]]checksum]' ...]
```

### 8.24.4. ARTIFACT NAMES

Artifact names can be any of the following:

- Maven coordinates using the **groupId:artifactId:version** format, for example **org.postgresql:postgresql:42.3.1**.
- HTTP, HTTPS, or FTP URLs

- Local paths

### 8.24.5. CHECKSUM VALIDATION

You can validate the checksum of an artifact after download. The algorithm defaults to **SHA-256** but it can also be **MD-5**, **SHA-1**, **SHA-256**, **SHA-384**, or `'SHA-512'`.

### 8.24.6. PATCH LIST OPTIONS

`--server-home='path/to/server'`

Sets the path of the server installation.

`--server-root='server'`

Sets the server root directory relative to the server home.

`*--maven-settings='$HOME/.m2/maven-settings.xml'`

Sets the path of a Maven **settings.xml** file and uses the default location, if not specified.

`-o, --overwrite`

Forces overwriting of artifacts in the **server/lib** directory. By default artifacts are not overwritten, which causes the installation to fail if an artifact already exists.

`-v, --verbose`

Shows verbose information about artifact downloads.

`-f, --force`

Forces download of remote artifacts, even if they are already present locally.

`-r, --retries=num`

The number of retries in case the downloaded artifacts do not match the supplied checksums.

`--clean`

Deletes all the contents of the **server/lib** directory before downloading artifacts.

### 8.24.7. EXAMPLES

`install -o org.postgresql:postgresql:42.3.1`

Installs the PostgreSQL JDBC driver JAR and overwrites if it already exists.

`install https://example.org/artifact.zip`

Downloads the **artifact.zip** and extracts the contents.

`install`

`https://example.org/artifact.zip|52d73f9b3611610ebbb4dca7c2ac1171218eb09891c1faba10f5f54c1d2acc13`

Downloads the **artifact.zip**, verifies its SHA-256 checksum, and extracts the contents.

`install https://example.org/artifact.zip|MD5|2b48d1871ee26f969d8481db94e103c2`

Downloads the **artifact.zip**, verifies its MD-5 checksum, and extracts the contents.

## 8.25. LOGGING(1)

### 8.25.1. NAME

logging - inspects and manipulates the Data Grid server runtime logging configuration.



## 8.25.2. SYNOPSIS

`logging list-loggers`

`logging list-appenders`

`logging set ['OPTIONS'] [LOGGER_NAME]`

`logging remove LOGGER_NAME`

## 8.25.3. LOGGING SET OPTIONS

`-l, --level='OFF|TRACE|DEBUG|INFO|WARN|ERROR|ALL'`

Specifies the logging level for the specific logger.

`-a, --appender='APPENDER'`

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.

## 8.25.4. 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.

## 8.26. LS(1)

### 8.26.1. NAME

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

### 8.26.2. SYNOPSIS

`ls ['PATH']`

### 8.26.3. OPTIONS

`-f, --format='[NAMES|VALUES|FULL]'`

This option currently only applies when listing caches.

- **NAMES**: only show the keys
- **VALUES**: show the keys and values
- **FULL**: show keys, values and metadata

`-l`

This option only applies when listing caches. Shortcut for **-f FULL**.

**-p, --pretty-print='[TABLE|CSV|JSON]'**

Prints the output using one of the following layouts:

- **TABLE**: tabular format. The column sizes are determined by the terminal width. This is the default.
- **CSV**: comma-separated values.
- **JSON**: JSON format.

**-m, --max-items='num'**

This option only applies when listing caches. The maximum number of items to show. Defaults to -1 (unlimited).

## 8.26.4. EXAMPLES

### **ls caches**

Lists the available caches.

### **ls ../**

Lists parent resources.

### **ls -l --pretty-print=CSV /containers/default/caches/mycache > mycache.csv**

Lists the content of a cache, including keys, values and metadata and redirects the contents to a file.

## 8.26.5. SEE ALSO

[cd\(1\)](#)

## 8.27. MIGRATE(1)

### 8.27.1. NAME

**migrate** - migrates data from one version of Data Grid to another.

### 8.27.2. SYNOPSIS

**migrate cluster connect**

**migrate cluster synchronize**

**migrate cluster disconnect**

**migrate cluster source-connection**

### 8.27.3. DESCRIPTION

Use the **migrate** command to migrate data from one version of Data Grid to another.

### 8.27.4. COMMAND SYNOPSIS

Migrate clusters

**migrate cluster connect**

Connects the target cluster to the source cluster.

**migrate cluster synchronize**

Synchronize data between the source cluster and the target cluster.

**migrate cluster disconnect**

Disconnects the target cluster from the source cluster.

**migrate cluster source-connection**

Gets connection configuration of the target cluster. The command will print "Not Found" if the connections hasn't been established.

**8.27.5. COMMON OPTIONS**

These options apply to all commands:

**-h, --help**

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

**CLUSTER CONNECT OPTIONS**

**\*-c, --cache\*='name'::**

The name of the cache to connect to the source.

**\*-f, --file\*='FILE'::**

Specifies a configuration file in JSON format, containing a single 'remote-store' element.

**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.

**CLUSTER DISCONNECT OPTIONS**

**-c, --cache='name'**

The name of the cache to disconnect from the source.

**8.27.6. CLUSTER CONNECTION OPTIONS**

**-c, --cache='name'**

The name of the cache to obtain the connection configuration.

**8.28. PATCH(1)**

### 8.28.1. NAME

`patch` - manages server patches.

### 8.28.2. 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.

### 8.28.3. 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'...]`

### 8.28.4. 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.

### 8.28.5. 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.

### 8.28.6. PATCH DESCRIBE OPTIONS

`-v, --verbose`

Shows the content of the patch, including information about individual files

### 8.28.7. 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.

### 8.28.8. PATCH CREATE OPTIONS

**-q, --qualifier='name'**

Specifies a descriptive qualifier string for the patch; for example, 'one-off for issue nnnn'.

## 8.28.9. 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.

## 8.29. PUT(1)

### 8.29.1. NAME

put - adds or updates cache entries.

### 8.29.2. DESCRIPTION

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

### 8.29.3. SYNOPSIS

```
put ['OPTIONS'] KEY [VALUE]
```

### 8.29.4. 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.

### 8.29.5. 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.

### 8.29.6. SEE ALSO

get(1), remove(1)

## 8.30. QUERY(1)

### 8.30.1. NAME

query - performs lckle queries to match entries in remote caches.

### 8.30.2. SYNOPSIS

query ['OPTIONS'] **QUERY\_STRING**

### 8.30.3. OPTIONS

**-c, --cache='NAME'**

Specifies the cache to query. Defaults to the currently selected cache.

**--max-results='MAX\_RESULTS'**

Sets the maximum 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**.

### 8.30.4. EXAMPLES

**query "from org.infinispan.example.Person p where p.gender = 'MALE'"**

Queries values in a remote cache to find entries from a Protobuf **Person** entity where the gender datatype is **MALE**.

### 8.30.5. SEE ALSO

index(1) schema(1)

## 8.31. QUIT(1)

### 8.31.1. NAME

quit - exits the command line interface.

### 8.31.2. SYNOPSIS

**quit**

**exit** and **bye** are command aliases.

### 8.31.3. EXAMPLE

**quit**

Ends the CLI session.

**exit**

Ends the CLI session.

**bye**

Ends the CLI session.

### 8.31.4. SEE ALSO

disconnect(1), shutdown(1)

## 8.32. REBALANCE(1)

### 8.32.1. NAME

rebalance - manages automatic rebalancing for caches

### 8.32.2. SYNOPSIS

**rebalance enable** ['PATH']

**rebalance disable** ['PATH']

### 8.32.3. EXAMPLES

**rebalance enable**

Enables automatic rebalancing in the current context. Running this command in the root context enables rebalancing for all caches.

**rebalance enable caches/mycache**

Enables automatic rebalancing for the cache named **mycache**.

**rebalance disable**

Disables automatic rebalancing in the current context. Running this command in the root context disables rebalancing for all caches.

**rebalance disable caches/mycache**

Disables automatic rebalancing for the cache named **mycache**.

## 8.33. REMOVE(1)

### 8.33.1. NAME

remove - deletes entries from a cache.

### 8.33.2. SYNOPSIS

```
remove KEY ['OPTIONS']
```

### 8.33.3. OPTIONS

```
--cache='NAME'
```

Specifies the cache from which to remove entries. Defaults to the currently selected cache.

### 8.33.4. EXAMPLE

```
remove --cache=mycache hola
```

Deletes the **hola** entry from the **mycache** cache.

### 8.33.5. SEE ALSO

cache(1), drop(1), clearcache(1)

## 8.34. RESET(1)

### 8.34.1. NAME

reset - restores the initial values of counters.

### 8.34.2. SYNOPSIS

```
reset ['COUNTER_NAME']
```

### 8.34.3. EXAMPLE

```
reset cnt_a
```

Resets the **cnt\_a** counter.

### 8.34.4. SEE ALSO

add(1), cas(1), drop(1)

## 8.35. SCHEMA(1)

### 8.35.1. NAME

schema - manipulates Protobuf schemas.

### 8.35.2. SYNOPSIS



`schema ls`

`schema upload --file=/path/to/schema.proto schema.proto`

`schema remove schema.proto`

`schema get schema.proto`

### 8.35.3. DESCRIPTION

Manage schemas with the **ls**, **upload**, **get**, **remove** subcommands.

### 8.35.4. COMMAND SYNOPSIS

`schema ls`

Lists the schemas installed in the server.

`schema upload --file='/path/to/schema.proto' 'schema.proto'`

Uploads a ProtoBuf schema file to the server.

`schema get 'schema.proto'`

Shows the content of the specified schema.

`schema remove 'schema.proto'`

Removes the specified schema from the server.

### 8.35.5. UPLOAD OPTIONS

`-f, --file='FILE'`

Uploads a file as a protobuf schema with the given name.

### 8.35.6. EXAMPLE

`schema upload --file=person.proto person.proto`

Registers a **person.proto** Protobuf schema.

### 8.35.7. SEE ALSO

`query(1)`

## 8.36. SERVER(1)

### 8.36.1. NAME

`server` - server configuration and state management.

### 8.36.2. DESCRIPTION

The **server** command describes and manages server endpoint connectors and datasources and retrieves aggregated diagnostic reports about both the server and host.

Reports provide details about CPU, memory, open files, network sockets and routing, threads, in addition to configuration and log files.

### 8.36.3. SYNOPSIS

**server report**

**server heap-dump** [--live]

**server connector ls**

**server connector describe** 'connector-name'

**server connector start** 'connector-name'

**server connector stop** 'connector-name'

**server connector ipfilter ls** 'connector-name'

**server connector ipfilter set** 'connector-name' --rules='[ACCEPT|REJECT]/cidr',...

**server connector ipfilter clear** 'connector-name'

**server datasource ls**

**server datasource test** 'datasource-name'

### 8.36.4. SERVER CONNECTOR IPFILTER OPTIONS

--rules='[ACCEPT|REJECT]/cidr',...

One or more IP filtering rules.

### 8.36.5. EXAMPLES

#### **server report**

Obtains a server report, including information about network, threads, memory, etc.

#### **server heap-dump**

Generates a JVM heap dump in the server data directory, returning the name of the generated file.

#### **server connector ls**

Lists all available connectors on the server.

#### **server connector describe endpoint-default**

Shows information about the specified connector, including host, port, local and global connections, IP filtering rules.

#### **server connector stop my-hotrod-connector**

Stops a connector dropping all established connections across the cluster. This command will be refused if attempting to stop the connector which is handling the request.

#### **server connector start my-hotrod-connector**

Starts a connector so that it can accept connections across the cluster.

#### **server connector ipfilter ls my-hotrod-connector**

Lists all IP filtering rules active on a connector across the cluster.

#### **server connector ipfilter set my-hotrod-connector --**

**rules=ACCEPT/192.168.0.0/16,REJECT/10.0.0.0/8** Sets IP filtering rules on a connector across the

cluster. Replaces all existing rules. This command will be refused if one of the rejection rules matches the address of the connection on which it is invoked.

**server connector ipfilter clear my-hotrod-connector**

Removes all IP filtering rules on a connector across the cluster.

**server datasource ls**

Lists all available datasources on the server.

**server datasource test my-datasource**

Performs a test connection on the datasource.

## 8.37. SHUTDOWN(1)

### 8.37.1. NAME

shutdown - stops server instances and clusters.

### 8.37.2. SYNOPSIS

shutdown server ['SERVERS']

shutdown cluster

shutdown container

### 8.37.3. 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**

Stops all nodes in the cluster after storing cluster state and persisting entries if there is a cache store.

**shutdown container**

Stops the data container without terminating the server process. Stores cluster state and persists entries if there is a cache store. Server instances remain running with active endpoints and clustering. REST calls to container resources will result in a 503 Service Unavailable response. The **shutdown container** command is intended for environments, such as Kubernetes, that automate resource lifecycle management. For self-managed environments you should use the **shutdown server** or **shutdown cluster** commands to stop servers.

### 8.37.4. SEE ALSO

connect(1), disconnect(1), quit(1)

## 8.38. SITE(1)

### 8.38.1. NAME

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

### 8.38.2. 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']

**site state-transfer-mode get|set** ['OPTIONS']

**site name**

**site view**

**site is-relay-node**

**site relay-nodes**

### 8.38.3. OPTIONS

**-c, --cache='CACHE\_NAME'**

Specifies a cache.

**-a, --all-caches**

Applies the command to all caches.

**-s, --site='SITE\_NAME'**

Specifies a backup location.

### 8.38.4. STATE TRANSFER MODE OPTIONS

**--mode='MODE'**

Sets the state transfer mode. Values are **MANUAL** (default) or **AUTO**.

### 8.38.5. EXAMPLES

**site status --cache=mycache**

Returns the status of all backup locations for **mycache**.

**site status --all-caches**

Returns the status of each backup location for all caches with backups.

**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**.

**site state-transfer-mode get --cache=myCache --site=NYC**

Retrieves the state transfer mode for **mycache** to **NYC**.

**site state-transfer-mode set --cache=myCache --site=NYC --mode=AUTO**

Configures automatic state transfer for **mycache** to **NYC**.

**site name**

Returns the name of the local site. If cross-site replication is not configured, the name of the local site is always "local".

**site view**

Returns a list of names for all sites or an empty list ("[]") if cross-site replication is not configured.

**site is-relay-node**

Returns true if the node handles RELAY messages between clusters.

**site relay-nodes**

Returns a list of relay nodes by their logical names.

## 8.39. STATS(1)

### 8.39.1. NAME

stats - displays statistics about resources.

### 8.39.2. SYNOPSIS

```
stats ['PATH']
```

### 8.39.3. EXAMPLES

**stats //containers/default**

Displays statistics about the default container.

**stats //containers/default/caches/mycache**

Displays statistics about the **mycache** cache.

**8.39.4. SEE ALSO**

cd(1), ls(1), describe(1)

**8.40. TASK(1)****8.40.1. NAME**

task - executes and uploads server-side tasks and scripts

**8.40.2. SYNOPSIS**

```
task upload --file='script' 'TASK_NAME'
```

```
task exec ['TASK_NAME']
```

**8.40.3. 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**.

**8.40.4. OPTIONS**

```
-P, --parameters='PARAMETERS'
```

Passes parameter values to tasks and scripts.

```
-f, --file='FILE'
```

Uploads script files with the given names.

**8.40.5. SEE ALSO**

ls(1)

**8.41. UNALIAS(1)****8.41.1. NAME**

unalias - deletes aliases.

**8.41.2. SYNOPSIS**

```
unalias 'ALIAS-NAME'
```

### 8.41.3. EXAMPLES

#### **unalias q**

Deletes the **q** alias.

### 8.41.4. SEE ALSO

config(1), alias(1)

## 8.42. USER(1)

### 8.42.1. NAME

user - manages Data Grid users in property security realms.

### 8.42.2. SYNOPSIS

**user ls**

**user create** 'username'

**user describe** 'username'

**user remove** 'username'

**user password** 'username'

**user groups** 'username'

**user encrypt-all**

**user roles ls** 'principal'

**user roles grant** --roles='role1'[, 'role2'...] 'principal'

**user roles deny** --roles='role1'[, 'role2'...] 'principal'

**user roles create** --permissions='perm1'[, 'perm2'...] 'role'

**user roles remove** 'role'

### 8.42.3. DESCRIPTION

Manage users in property realms with the **ls**, **create**, **describe**, **remove**, **password**, **groups** and **encrypt-all** subcommands. List and modify principal to role mappings with the **roles** subcommand when using the cluster role mapper for authorization.

### 8.42.4. 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.

**user roles ls 'principal'**

Lists all roles of the specified principal (user or group).

**user roles grant --roles='role1'[, 'role2'...] 'principal'**

Grants one or more roles to a principal.

**user roles deny --roles='role1'[, 'role2'...] 'principal'**

Denies one or more roles to a principal.

**user roles create --permissions='perm1'[, 'perm2'...] 'role'**

Creates a new role with the specified permissions.

**user roles remove 'role'**

Deletes an existing role.

### 8.42.5. 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**.

### 8.42.6. 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).

**8.42.7. USER LS OPTIONS****--groups**

Shows a list of groups instead of the users.

**8.42.8. USER ENCRYPT-ALL OPTIONS****-a, --algorithms**

Specifies the algorithms used to hash the password.

**8.42.9. USER ROLES OPTIONS****-p, --permissions**

Specifies one or more of the following permissions: **LIFECYCLE, READ, WRITE, EXEC, LISTEN, BULK\_READ, BULK\_WRITE, ADMIN, CREATE, MONITOR, ALL, ALL\_READ, ALL\_WRITE**

**8.43. VERSION(1)****8.43.1. NAME**

version - displays the server version and CLI version.

**8.43.2. SYNOPSIS**

version

**8.43.3. EXAMPLE****version**

Returns the version for the server and the CLI.

**8.43.4. SEE ALSO**

help(1)