Red Hat Data Grid 8.3

Using the Data Grid Command Line Interface

Access and manage remote caches with the Data Grid CLI

Last Updated: 2023-11-24

Red Hat Data Grid 8.3 Using the Data Grid Command Line Interface

Access and manage remote caches with the Data Grid CLI

Legal Notice

Copyright © 2023 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 ® is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js ® 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 ® 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

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 8.3 Documentation
- Data Grid 8.3 Component Details
- Supported Configurations for Data Grid 8.3
- 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 AND MODIFYING DATA GRID USERS

Add Data Grid user credentials and assign permissions to control access to data.

Data Grid server installations use a property realm to authenticate users for the Hot Rod and REST endpoints. This means you need to create at least one user before you can access Data Grid.

By default, users also need roles with permissions to access caches and interact with Data Grid resources. You can assign roles to users individually or add users to groups that have role permissions.

You create users and assign roles with the **user** command in the Data Grid command line interface (CLI).

TIP

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

1.1.1. Adding credentials

You need an **admin** user for the Data Grid Console and full control over your Data Grid environment. For this reason you should create a user with **admin** permissions the first time you add credentials.

Procedure

1. Open a terminal in **$RHDG_HOME**.
2. Create an **admin** user with the **user create** command.
   - Add a user assigned to the **admin** group.

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

   - Use implicit authorization to gain **admin** permissions.

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

3. Open **user.properties** and **groups.properties** with any text editor to verify users and groups.

```
$ cat server/conf/users.properties

#$REALM_NAME=default$
#$ALGORITHM=encrypted$
myuser=scram-sha-1\:BYGcIAwvf6b...

$ cat server/conf/groups.properties

myuser=admin
```

1.1.2. Assigning roles to users

Assign roles to users so they have the correct permissions to access data and modify Data Grid resources.

Procedure

1. Start a CLI session with an **admin** user.

```
$ bin/cli.sh
```

2. Assign the **deployer** role to "katie".

```
[//containers/default]> user roles grant --roles=deployer katie
```

3. List roles for "katie".

```
[//containers/default]> user roles ls katie
["deployer"]
```

1.1.3. 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.

Procedure

1. Start a CLI session with an **admin** user.
2. Use the **user create** command to create a group.
   a. Specify "developers" as the group name with the **--groups** argument.
   b. Set a username and password for the group.
   In a property realm, a group is a special type of user that also requires a username and password.

```
[//containers/default]> user create --groups=developers developers -p changeme
```

3. List groups.

```
[//containers/default]> user ls --groups
["developers"]
```

4. Assign the **application** role to the "developers" group.

```
[//containers/default]> user roles grant --roles=application developers
```

5. List roles for the "developers" group.

```
[//containers/default]> user roles ls developers
["application"]
```

6. Add existing users, one at a time, to the group as required.

```
[//containers/default]> user groups john --groups=developers
```

1.1.4. User roles and permissions

Data Grid includes a default set of roles that grant users with permissions to access data and interact with Data Grid resources.

ClusterRoleMapper is the default mechanism that Data Grid uses to associate security principals to authorization roles.

IMPORTANT

ClusterRoleMapper matches principal names to role names. A user named **admin** gets **admin** permissions automatically, a user named **deployer** gets **deployer** permissions, and so on.

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

Reference

- org.infinispan.security.AuthorizationPermission Enumeration
- 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
```

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.

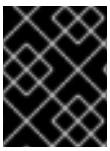
Procedure

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. Data Grid cluster restarts

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

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

If you shutdown a cluster with the **shutdown cluster** command, clusters become fully operational only after all nodes rejoin.
You can restart nodes in any order but the cluster remains in DEGRADED state until all nodes that were joined before shutdown are running.

CHAPTER 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 and enter your credentials when prompted.

```
bin/cli.sh
```

2. 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
```

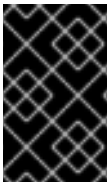
2. View cache configuration with the **describe** command.

```
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, the Data Grid server might abruptly stop without issuing an exception message. 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 cache names and cache template 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"/>
  </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"
      }
    },
    "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"
  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"/>
  </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"
      }
    },
    "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"
  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:13.0 https://infinispan.org/schemas/infinispan-config-
13.0.xsd
                          urn:infinispan:server:13.0 https://infinispan.org/schemas/infinispan-server-13.0.xsd"
      xmlns="urn:infinispan:config:13.0"
      xmlns:server="urn:infinispan:server:13.0">
  <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>
```

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"
```

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"
            }
          }
        }
      }
    }
  }
}
```

Additional resources

- Data Grid configuration schema reference
- infinispan-config-13.0.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**.

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
```

TIP

Use **echo** to add commands in interactive mode.

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

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

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

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

3. Verify your CLI configuration.

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

```
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	**$HOME/.config/red_hat_data_grid**
Microsoft Windows	**%APPDATA%/Sun/Java/red_hat_data_grid**
Mac OS	**$HOME/Library/Java/red_hat_data_grid**

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

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'

8.16.3. DESCRIPTION

List, create, and remove credentials inside a keystore. 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.

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. 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.

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. LOGGING(1)

8.23.1. NAME

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

8.23.2. SYNOPSIS

logging list-loggers

logging list-appenders

logging set ['OPTIONS'] [**LOGGER_NAME**]

logging remove LOGGER_NAME

8.23.3. 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.

8.23.4. 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.

8.24. LS(1)

8.24.1. NAME

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

8.24.2. SYNOPSIS

ls ['PATH']

8.24.3. EXAMPLES

ls caches
Lists the available caches.

ls ../
Lists parent resources.

8.24.4. SEE ALSO

cd(1)

8.25. MIGRATE(1)

8.25.1. NAME

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

8.25.2. SYNOPSIS

migrate cluster connect

migrate cluster synchronize

migrate cluster disconnect

migrate cluster source-connection

8.25.3. DESCRIPTION

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

8.25.4. 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.

8.25.5. 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.

8.25.6. CLUSTER CONNECTION OPTIONS

-c, --cache='name'

The name of the cache to obtain the connection configuration.

8.26. PATCH(1)

8.26.1. NAME

patch - manages server patches.

8.26.2. 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.

8.26.3. 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'...]

8.26.4. 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.

8.26.5. PATCH INSTALL OPTIONS

--dry-run

Shows the operations that the patch peforms without applying any changes.

--server='path/to/server'

Sets the path to a target server outside the current server home directory.

8.26.6. PATCH DESCRIBE OPTIONS

-v, --verbose

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

8.26.7. PATCH ROLLBACK OPTIONS

--dry-run

Shows the operations that the patch peforms without applying any changes.

--server='path/to/server'

Sets the path to a target server outside the current server home directory.

8.26.8. PATCH CREATE OPTIONS

-q, --qualifier='name'

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

8.26.9. 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.

8.27. PUT(1)

8.27.1. NAME

put - adds or updates cache entries.

8.27.2. DESCRIPTION

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

8.27.3. SYNOPSIS

put ['OPTIONS'] **KEY** [**VALUE**]

8.27.4. 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.

8.27.5. 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.

8.27.6. SEE ALSO

get(1), remove(1)

8.28. QUERY(1)

8.28.1. NAME

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

8.28.2. SYNOPSIS

query ['OPTIONS'] **QUERY_STRING**

8.28.3. 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**.

8.28.4. 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**.

8.28.5. SEE ALSO

schema(1)

8.29. QUIT(1)

8.29.1. NAME

quit - exits the command line interface.

8.29.2. SYNOPSIS

quit

exit and **bye** are command aliases.

8.29.3. EXAMPLE

quit
Ends the CLI session.

exit
Ends the CLI session.

bye
Ends the CLI session.

8.29.4. SEE ALSO

disconnect(1), shutdown(1)

8.30. REBALANCE(1)

8.30.1. NAME

rebalance - manages automatic rebalancing for caches

8.30.2. SYNOPSIS

rebalance enable ['PATH']

rebalance disable ['PATH']

8.30.3. 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**.

8.31. REMOVE(1)

8.31.1. NAME

remove - deletes entries from a cache.

8.31.2. SYNOPSIS

remove KEY ['OPTIONS']

8.31.3. OPTIONS

--cache='NAME'

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

8.31.4. EXAMPLE

remove --cache=mycache hola
Deletes the **hola** entry from the **mycache** cache.

8.31.5. SEE ALSO

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

8.32. RESET(1)

8.32.1. NAME

reset - restores the initial values of counters.

8.32.2. SYNOPSIS

reset ['COUNTER_NAME']

8.32.3. EXAMPLE

reset cnt_a
Resets the **cnt_a** counter.

8.32.4. SEE ALSO

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

8.33. SCHEMA(1)

8.33.1. NAME

schema - uploads and registers protobuf schemas.

8.33.2. SYNOPSIS

schema ['OPTIONS'] **SCHEMA_NAME**

8.33.3. OPTIONS

-u, --upload='FILE'

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

8.33.4. EXAMPLE

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

8.33.5. SEE ALSO

query(1)

8.34. SERVER(1)

8.34.1. NAME

server - server configuration and state management.

8.34.2. 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.

8.34.3. SYNOPSIS

server report

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'

8.34.4. SERVER CONNECTOR IPFILTER OPTIONS

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

One or more IP filtering rules.

8.34.5. EXAMPLES

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

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.

8.35. SHUTDOWN(1)

8.35.1. NAME

shutdown - stops server instances and clusters.

8.35.2. SYNOPSIS

shutdown server ['SERVERS']

shutdown cluster

shutdown container

8.35.3. 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.

8.35.4. SEE ALSO

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

8.36. SITE(1)

8.36.1. NAME

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

8.36.2. 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

8.36.3. 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.

8.36.4. STATE TRANSFER MODE OPTIONS

--mode='MODE'
Sets the state transfer mode. Values are **MANUAL** (default) or **AUTO**.

8.36.5. 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.

8.37. STATS(1)

8.37.1. NAME

stats - displays statistics about resources.

8.37.2. SYNOPSIS

stats ['PATH']

8.37.3. EXAMPLES

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

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

8.37.4. SEE ALSO

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

8.38. TASK(1)

8.38.1. NAME

task - executes and uploads server-side tasks and scripts

8.38.2. SYNOPSIS

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

task exec ['TASK_NAME']

8.38.3. 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**.

8.38.4. OPTIONS

-P, --parameters='PARAMETERS'

Passes parameter values to tasks and scripts.

-f, --file='FILE'

Uploads script files with the given names.

8.38.5. SEE ALSO

ls(1)

8.39. UNALIAS(1)

8.39.1. NAME

unalias - deletes aliases.

8.39.2. SYNOPSIS

unalias 'ALIAS-NAME'

8.39.3. EXAMPLES

unalias q
Deletes the **q** alias.

8.39.4. SEE ALSO

config(1), alias(1)

8.40. USER(1)

8.40.1. NAME

user - manages Data Grid users in property security realms.

8.40.2. 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'

8.40.3. 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.

8.40.4. 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.

8.40.5. 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**.

8.40.6. 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).

8.40.7. USER LS OPTIONS

--groups

Shows a list of groups instead of the users.

8.40.8. USER ENCRYPT-ALL OPTIONS

-a, --algorithms

Specifies the algorithms used to hash the password.

8.41. VERSION(1)

8.41.1. NAME

version - displays the server version and CLI version.

8.41.2. SYNOPSIS

version

8.41.3. EXAMPLE

version
Returns the version for the server and the CLI.

8.41.4. SEE ALSO

help(1)