Using the Infinispan Command Line Interface

Table of Contents

Chapter 1. Getting Started with the Infinispan CLI

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

Prerequisites

- At least one running Infinispan server.

1.1. Starting the Infinispan CLI

Start the Infinispan CLI as follows:

1. Open a terminal in `$ISPN_HOME`.
2. Run the CLI.

```
$ bin/cli.sh
[disconnected]>
```

1.2. Connecting to Infinispan Servers

Do one of the following:

- Run the `connect` command to connect to a Infinispan server on the default port of `11222`:

```
[disconnected]> connect
[hostname1@cluster//containers/default]>
```

- Specify the location of a Infinispan server. For example, connect to a local server that has a port offset of 100:

```
[disconnected]> connect 127.0.0.1:11322
[hostname2@cluster//containers/default]>
```

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

1.3. Navigating CLI Resources

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

When you connect to a Infinispan 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
```

- Use `cd` to navigate the resource tree.

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

- Use `describe` to view information about resources.

```
[//containers/default]> describe
{
  "name" : "default",
  "version" : "xx.x.x-FINAL",
  "cluster_name" : "cluster",
  "coordinator" : true,
  "cache_configuration_names" : [ "org.infinispan.REPL_ASYNC", "___protobuf_metadata",
"org.infinispan.DIST_SYNC", "org.infinispan.LOCAL",
"org.infinispan.INVALIDATION_SYNC", "org.infinispan.REPL_SYNC",
"org.infinispan.SCATTERED_SYNC", "org.infinispan.INVALIDATION_ASYNC",
"org.infinispan.DIST_ASYNC" ],
  "physical_addresses" : "[192.0.2.0:7800]",
  "coordinator_address" : "<hostname>",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "1",
  "running_cache_count" : "1",
  "node_address" : "<hostname>",
  "cluster_members" : [ "<hostname1>", "<hostname2>" ],
  "cluster_members_physical_addresses" : [ "192.0.2.0:7800", "192.0.2.0:7801" ],
  "cluster_size" : 2,
  "defined_caches" : [ {
    "name" : "mycache",
    "started" : true
  }, {
    "name" : "___protobuf_metadata",
    "started" : true
  } ]
}
```

1.3.1. CLI Resources

The Infinispan CLI exposes different resources to:

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

Cache Resources

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

`caches`
: Infinispan 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`
: Configuration files.

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

Cluster Resources

```
[hostname@cluster/]> ls
containers
cluster
server
```

`containers`
: Cache containers on the Infinispan cluster.

`cluster`
: Lists Infinispan servers joined to the cluster.

`server`
: Resources for managing and monitoring Infinispan servers.

1.4. Shutting Down Infinispan Servers

Use the CLI to gracefully shutdown running servers. This ensures that Infinispan passivates all entries to disk and persists state.

- Use the `shutdown server` command to stop individual servers, for example:

```
[//containers/default]> shutdown server server_hostname
```

- Use the `shutdown cluster` command to stop all servers joined to the cluster, for example:

```
[//containers/default]> shutdown cluster
```

Infinispan servers log the following shutdown messages:

```
INFO  [org.infinispan.SERVER] (pool-3-thread-1) ISPN080002: Infinispan Server stopping
INFO  [org.infinispan.CONTAINER] (pool-3-thread-1) ISPN000029: Passivating all entries
to disk
INFO  [org.infinispan.CONTAINER] (pool-3-thread-1) ISPN000030: Passivated 28 entries
in 46 milliseconds
INFO  [org.infinispan.CLUSTER] (pool-3-thread-1) ISPN000080: Disconnecting JGroups
channel cluster
INFO  [org.infinispan.CONTAINER] (pool-3-thread-1) ISPN000390: Persisted state,
version=<Infinispan version> timestamp=YYYY-MM-DDTHH:MM:SS
INFO  [org.infinispan.SERVER] (pool-3-thread-1) ISPN080003: Infinispan Server stopped
INFO  [org.infinispan.SERVER] (Thread-0) ISPN080002: Infinispan Server stopping
INFO  [org.infinispan.SERVER] (Thread-0) ISPN080003: Infinispan Server stopped
```

When you shutdown Infinispan clusters, the shutdown messages include:

```
INFO  [org.infinispan.SERVER] (pool-3-thread-1) ISPN080029: Cluster shutdown
INFO  [org.infinispan.CLUSTER] (pool-3-thread-1) ISPN000080: Disconnecting JGroups
channel cluster
```

Chapter 2. Performing Cache Operations with the Infinispan CLI

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

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

2.1. Creating Caches from Templates

Use Infinispan cache templates to add caches with recommended default settings.

Procedure

1. Create a distributed, synchronous cache from a template and name it "mycache".

```
[//containers/default]> create cache --template=org.infinispan.DIST_SYNC mycache
```

Press the tab key after the `--template=` argument to list available cache templates.

2. Retrieve the cache configuration.

```
[//containers/default]> describe caches/mycache
{
  "distributed-cache" : {
    "mode" : "SYNC",
    "remote-timeout" : 17500,
    "state-transfer" : {
      "timeout" : 60000
    },
    "transaction" : {
      "mode" : "NONE"
    },
    "locking" : {
      "concurrency-level" : 1000,
      "acquire-timeout" : 15000,
      "striping" : false
    }
  }
}
```

2.2. Adding Cache Entries

Add data to caches with the Infinispan CLI.

Prerequisites

- Create a cache named "mycache".

Procedure

1. Add a key/value pair to `mycache`.

```
[//containers/default]> put --cache=mycache hello world
```

If the CLI is in the context of a cache, do `put k1 v1` for example:

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

2. List keys in the cache.

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

3. Get the value for the `hello` key.
   a. Navigate to the cache.

```
[//containers/default]> cd caches/mycache
```

   b. Use the `get` command to retrieve the key value.

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

2.3. Deleting Cache Entries

Remove data from caches with the Infinispan CLI.

Prerequisites

- Create a cache named "mycache" and add entries.

Procedure

Either delete all entries or remove specific entries as follows:

- Delete all entries from a cache.

```
[//containers/default]> clearcache mycache
```

- Remove specific entries from a cache.

```
[//containers/default]> remove --cache=mycache hello
```

2.4. Creating Custom Caches

Add caches with custom Infinispan configuration files in XML or JSON format.

Procedure

- Add the path to your configuration file with the `--file=` option as follows:

```
[//containers/default]> create cache --file=prod_dist_cache.xml dist_cache_01
```

XML Configuration

A configuration in `XML` format must conform to the schema and include:

- `<infinispan>` root element.
- `<cache-container>` definition.

The following example shows a valid `XML` configuration:

```
<infinispan>
    <cache-container>
        <distributed-cache name="cacheName" mode="SYNC">
            <memory>
                <object size="20"/>
            </memory>
        </distributed-cache>
    </cache-container>
</infinispan>
```

JSON Configuration

A configuration in `JSON` format payload:

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

The following example shows the previous XML configuration in JSON format:

```
{
  "distributed-cache": {
    "mode": "SYNC",
    "memory": {
      "object": {
        "size": 20
      }
    }
  }
}
```

Chapter 3. Performing Batch Operations

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

Prerequisites

- A running Infinispan cluster.

3.1. Performing Batch Operations with Files

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

Procedure

1. Create a file that contains a set of operations.

   For example, create a file named `batch` that creates a cache named `mybatch`, adds two entries to the cache, and disconnects from the CLI.

   ```
   $ cat > batch<<EOF
   create cache --template=org.infinispan.DIST_SYNC mybatch
   put --cache=mybatch hello world
   put --cache=mybatch hola mundo
   disconnect
   EOF
   ```

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

   ```
   $ bin/cli.sh -c localhost:11222 -f batch
   ```

3. Open a new CLI connection to Infinispan and verify `mybatch`.

   ```
   [//containers/default]> ls caches
   ___protobuf_metadata
   mybatch
   [//containers/default]> ls caches/mybatch
   hola
   hello
   [//containers/default]> disconnect
   [disconnected]>
   ```

3.2. Performing Batch Operations Interactively

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

Procedure

1. Start the Infinispan CLI in interactive mode.

```
$ bin/cli.sh -c localhost:11222 -f -
```

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

```
$ bin/cli.sh -f -
connect
```

2. Run batch operations, for example:

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

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. Querying Caches with Protobuf Metadata

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

Prerequisites

- Start the Infinispan CLI.
- Connect to a running Infinispan cluster.

4.1. Configuring Media Types

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

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

Procedure

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

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

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

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

3. Verify `pcache`.

```
[//containers/default]> ls caches
pcache
___protobuf_metadata
[//containers/default]> describe caches/pcache
{
  "distributed-cache" : {
    "mode" : "SYNC",
    "encoding" : {
      "key" : {
        "media-type" : "application/x-protostream"
      },
      "value" : {
        "media-type" : "application/x-protostream"
      }
    },
    "transaction" : {
      "mode" : "NONE"
    }
  }
}
```

4. Add an entry to `pcache` and check the encoding.

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

4.2. Registering Protobuf Schemas

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

Procedure

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

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

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

message PhoneNumber {
    required string number = 1;
}

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

    optional Gender gender = 7;
}
```

2. Register `person.proto`.

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

3. Verify `person.proto`.

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

4.3. Querying Caches with Protobuf Schemas

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

For example, consider the following JSON documents:

lukecage.json

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

jessicajones.json

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

matthewmurdock.json

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

Each of the preceding JSON documents contains:

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

Procedure

1. Navigate to the `pcache` cache.

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

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

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

3. Verify that the entries exist.

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

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

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

Chapter 5. Command Reference

View help pages for Infinispan CLI commands.

Use the `-h` flag to access help pages directly from your CLI session.

5.1. ADD(1)

5.1.1. NAME

add - adds/subtracts a delta to/from a counter

5.1.2. SYNOPSIS

add [OPTIONS] ['NAME']

5.1.3. DESCRIPTION

The **add** adds or subtracts a value ('delta') to/from a counter. If no delta is specified, the counter is incremented by 1.

5.1.4. OPTIONS

--delta='nnn'

The delta to add/subtract from the counter's value. Defaults

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

Whether the value of the counter should be printed.

5.1.5. EXAMPLES

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

Adds `10` to the current value of the counter 'cnt_a'

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

Subtracts `5` from the current value of the counter 'cnt_a'

5.1.6. SEE ALSO

cas(1), reset(1)

5.2. CACHE(1)

5.2.1. NAME

cache - selects a cache making it the default for subsequent operations

5.2.2. SYNOPSIS

cache ['NAME']

5.2.3. DESCRIPTION

The **cache** command selects an existing cache, making it the default for subsequent cache manipulation operations such as **get** and **put**. If the command is invoked without specifying an argument, the name of the currently selected cache will be shown.

5.2.4. SEE ALSO

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

5.3. CAS(1)

5.3.1. NAME

cas - compares and swaps the value of a counter

5.3.2. SYNOPSIS

cas [OPTIONS] ['NAME']

5.3.3. DESCRIPTION

The **cas** command performs a 'compare-and-swap' operation on a counter.

5.3.4. OPTIONS

--expect='nnn'
: The expected value

--value='nnn'
: The new value

-q, --quiet='[true|false]'
: Whether the value of the counter should be printed.

5.3.5. EXAMPLES

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

Sets the value of the counter 'cnt_a' to `20` only if its current value is `10`

5.3.6. SEE ALSO

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

5.4. CD(1)

5.4.1. NAME

cd - navigates the resource tree

5.4.2. SYNOPSIS

cd 'PATH'

5.4.3. DESCRIPTION

The **cd** command navigates the server resource tree. `PATH` can be either absolute or relative to the current resource. The special `..` resource name indicates the parent of a resource.

5.4.4. SEE ALSO

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

5.5. CLEARCACHE(1)

5.5.1. NAME

clearcache - clears a cache from all content

5.5.2. SYNOPSIS

clearcache ['NAME']

5.5.3. DESCRIPTION

The **clearcache** command removes all entries from a cache. Unless a cache name has been specified as an argument, the currently selected cache will be used.

5.5.4. SEE ALSO

cache(1)

5.6. CONNECT(1)

5.6.1. NAME

connect - connects to a ${infinispan.brand.name} server

5.6.2. SYNOPSIS

connect ['OPTIONS'] ['CONNECTION']

5.6.3. DESCRIPTION

The **connect** command connects to a running ${infinispan.brand.name} server. If the command is invoked without specifying an argument, the CLI will default to http://localhost:11222. If the connection requires authentication, the CLI will prompt for credentials.

5.6.4. OPTIONS

-u, --username='USERNAME'

The username to use when connecting if the server requires credentials.

-p, --password='PASSWORD'

The password to use when connecting if the server requires credentials.

5.6.5. SEE ALSO

disconnect(1)

5.7. CONTAINER(1)

5.7.1. NAME

container - selects a container making it the default for subsequent operations

5.7.2. SYNOPSIS

container ['NAME']

5.7.3. DESCRIPTION

The **container** command selects an existing container, making it the default. This is equivalent to navigating the resource tree using `cd /containers/NAME`. If the command is invoked without specifying an argument, the name of the currently selected container will be shown.

5.7.4. SEE ALSO

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

5.8. COUNTER(1)

5.8.1. NAME

counter - selects a counter making it the default for subsequent operations

5.8.2. SYNOPSIS

counter ['NAME']

5.8.3. DESCRIPTION

The **counter** command selects an existing counter, making it the default for subsequent counter manipulation operations such as **inc** and **dec**. If the command is invoked without specifying an argument, the name of the currently selected counter will be shown.

5.8.4. SEE ALSO

add(1), cas(1), inc(1), dec(1)

5.9. CREATE(1)

5.9.1. NAME

create - creates a cache or counter

5.9.2. SYNOPSIS

create cache [OPTIONS] NAME

create counter [OPTIONS] NAME

5.9.3. DESCRIPTION

Creates a cache or a counter on the server.

5.9.4. CREATE CACHE OPTIONS

-f, --file='FILE'

A file containing a JSON or XML configuration.

-t, --template='TEMPLATE'

The configuration template to use.

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

Whether the cache should be made volatile in the server so that it won't survive restarts.

5.9.5. CREATE COUNTER OPTIONS

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

The type of counter.

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

Sets whether the counter should be **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 the counter. Only valid for **strong** counters

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

Sets the upper bound of the counter. Only valid for **strong** counters

5.9.6. SEE ALSO

drop(1)

5.10. DESCRIBE(1)

5.10.1. NAME

describe - describes a resource

5.10.2. SYNOPSIS

describe ['PATH']

5.10.3. DESCRIPTION

The **describe** command retrieves and shows information about the specified resource or the currently selected one. The information displayed depends on the type of the resource.

5.10.4. EXAMPLES

```
describe //containers/default
```

Shows information about a container:

```
{
  "name" : "default",
  "version" : "10.0.0-SNAPSHOT",
  "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", "mycache", "org.infinispan.INVALIDATION_ASYNC",
"org.infinispan.DIST_ASYNC" ],
  "cluster_name" : "cluster",
  "physical_addresses" : "[172.17.0.1:7800]",
  "cache_manager_status" : "RUNNING",
  "created_cache_count" : "1",
  "running_cache_count" : "1",
  "node_address" : "host-29389",
  "cluster_members" : [ "host-29389" ],
  "cluster_members_physical_addresses" : [ "172.17.0.1:7800" ],
  "cluster_size" : 1,
  "defined_caches" : [ {
    "name" : "mycache",
    "started" : true
  }, {
    "name" : "___protobuf_metadata",
    "started" : true
  } ],
  "coordinator_address" : "host-29389"
}
```


`describe //containers/default`

Shows information about a cache:


```
{
  "distributed-cache" : {
    "mode" : "SYNC",
    "remote-timeout" : 17500,
    "state-transfer" : {
      "timeout" : 60000
    },
    "transaction" : {
      "mode" : "NONE"
    },
    "locking" : {
      "concurrency-level" : 1000,
      "acquire-timeout" : 15000,
      "striping" : false
    }
  }
}
```

`describe //containers/default/caches/mycache/k1`

Shows information about a cache key:

```
{
"cluster-backup-owners" : [ "" ],
"cluster-node-name" : [ "host-29389" ],
"cluster-primary-owner" : [ "host-29389" ],
"cluster-server-address" : [ "[172.17.0.1:7800]" ],
"connection" : [ "keep-alive" ],
"content-length" : [ "0" ],
"content-type" : [ "application/octet-stream" ],
"etag" : [ "1461965704" ],
"last-modified" : [ "Thu, 1 Jan 1970 01:00:00 +0100" ]
}
```

`describe //containers/default/counters/cnt1`

Shows information about a counter:

```
{
  "strong-counter" : {
    "name" : "cnt1",
    "initial-value" : 0,
    "storage" : "PERSISTENT",
    "lower-bound" : -9223372036854775808,
    "upper-bound" : 100
  }
}
```

5.10.5. SEE ALSO

cd(1), ls(1)

5.11. DISCONNECT(1)

5.11.1. NAME

disconnect - disconnects from a ${infinispan.brand.name} server

5.11.2. SYNOPSIS

disconnect

5.11.3. DESCRIPTION

The **disconnect** command disconnects from a ${infinispan.brand.name} server. Use the **connect**

5.11.4. SEE ALSO

connect(1)

5.12. DROP(1)

5.12.1. NAME

drop - drops (deletes) a cache or counter

5.12.2. SYNOPSIS

drop cache NAME

drop counter NAME

5.12.3. DESCRIPTION

Drops (deletes) a cache or a counter from the server.

5.12.4. SEE ALSO

create(1)

5.13. ENCODING(1)

5.13.1. NAME

encoding - sets/shows the encoding used to put/get entries to/from the server.

5.13.2. SYNOPSIS

encoding ['ENCODING']

5.13.3. DESCRIPTION

The **encoding** command is used to set a default encoding to be used for subsequent **put**/**get** operations on a cache. When invoked without arguments it shows the currently selected encoding. Encodings must use the standard MIME types (IANA media types) naming convention. Examples of valid encodings are:

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

5.13.4. SEE ALSO

get(1), put(1)

5.14. GET(1)

5.14.1. NAME

get - retrieves a cache entry

5.14.2. SYNOPSIS

get [OPTIONS] KEY

5.14.3. DESCRIPTION

The **get** command retrieves an entry from a cache.

5.14.4. OPTIONS

-c, --cache='NAME'

The name of the cache. If not specified, the currently selected cache will be used.

5.14.5. SEE ALSO

query(1), put(1)

SH DESCRIPTION

Prints out a description of how to invoke a specific command or a list of available commands .SH ARGUMENTS .IP command an optional command name. If omitted a list of all available commands is printed

5.15. LS(1)

5.15.1. NAME

ls - shows child resources of a resource.

5.15.2. SYNOPSIS

ls ['PATH']

5.15.3. DESCRIPTION

The **ls** shows a list of child resources of the currently selected resource or the named resource specified as an argument.

5.15.4. SEE ALSO

cd(1)

5.16. PUT(1)

5.16.1. NAME

put - puts an entry in a cache

5.16.2. SYNOPSIS

put [OPTIONS] KEY [VALUE]

5.16.3. DESCRIPTION

The **put** command puts an entry in a cache. The first argument is compulsory and indicates the key of the entry. The second argument is optional when the

5.16.4. OPTIONS

-c, --cache='NAME'

The name of the cache. If not specified, the currently selected cache will be used.

-e, --encoding='ENCODING'

The encoding to use when putting the trname of the cache. If not specified, the currently selected cache will be used.

-f, --file='FILE'

The file to be used to supply the value.

-l, --ttl='TTL'

The time-to-live of the entry in seconds. If 0 or not specified, the default expiration for the cache will be used. If a negative value, the entry never expires.

-i, --max-idle='MAXIDLE'

The maximum idle time of the entry in seconds. If 0 or not specified, the default expiration for the cache will be used. If a negative value, the entry never expires.

-a, --if-absent=[true|false]

Only puts an entry if it doesn't already exist.

5.16.5. SEE ALSO

get(1), remove(1)

5.17. QUERY(1)

5.17.1. NAME

query - retrieves entries from a cache using an Ickle query

5.17.2. SYNOPSIS

query [OPTIONS] QUERY

5.17.3. DESCRIPTION

The **query** command retrieves an entry from a cache.

5.17.4. CREATE CACHE OPTIONS

-f, --file='FILE'
A file containing a JSON or XML configuration.

-t, --template='TEMPLATE'
The configuration template to use.

-p, --permanent='[true|false]'
Whether the cache should be made permanent in the server so that it survives restarts.

5.17.5. CREATE COUNTER OPTIONS

-t, --type='[weak|strong]' The type of counter.

5.17.6. SEE ALSO

get(1)

5.18. QUIT(1)

5.18.1. NAME

quit - exits the CLI

5.18.2. SYNOPSIS

quit

5.18.3. DESCRIPTION

The **quit** command exits the CLI.

5.18.4. SEE ALSO

disconnect(1)

SH DESCRIPTION

Removes the entry associated with the specified key from a cache. .SH ARGUMENTS .IP cache (optional) the name of the cache to use. If not specified, the currently selected cache will be used. See the .B cache command .IP key the key for which to remove the associated entry .SH OUTPUT

5.19. RESET(1)

5.19.1. NAME

reset - resets a counter

5.19.2. SYNOPSIS

reset ['NAME']

5.19.3. DESCRIPTION

The **reset** command resets a counter to its initial value.

5.19.4. SEE ALSO

add(1), cas(1)

5.20. SCHEMA(1)

5.20.1. NAME

schema - manipulates protobuf schemas

5.20.2. SYNOPSIS

schema [OPTIONS] `NAME` [`VALUE`]

5.20.3. DESCRIPTION

The **schema** command allows manipulation (upload, retrieve) of protobuf schemas.

5.20.4. OPTIONS

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

5.21. SHUTDOWN(1)

5.21.1. NAME

shutdown - shuts down individual servers or the whole cluster

5.21.2. SYNOPSIS

shutdown server ['SERVERS']

shutdown cluster

5.21.3. DESCRIPTION

The **shutdown** command performs an orderly shutdown of individual server nodes or of the whole cluster.

5.22. SITE(1)

5.22.1. NAME

site - manages backup sites

5.22.2. 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]

5.22.3. DESCRIPTION

Manages backup sites status and operations.

5.22.4. BRING-ONLINE OPTIONS

Brings a site online for the specified cache

-c, --cache='CACHENAME'
: The cache name

-s, --site='SITENAME'
The name of the site

5.22.5. TAKE-OFFLINE OPTIONS

Takes a site offline for the specified cache

-c, --cache='CACHENAME'
The cache name

-s, --site='SITENAME'
The name of the site

5.22.6. PUSH-SITE-STATE OPTIONS

Pushes state to a remote site

-c, --cache='CACHENAME'
The cache name

-s, --site='SITENAME'
The name of the site

5.22.7. CANCEL-PUSH-STATE OPTIONS

Cancels pushing state to a remote site

-c, --cache='CACHENAME'
The cache name

-s, --site='SITENAME'
The name of the site

5.22.8. CANCEL-RECEIVE-STATE OPTIONS

Cancels receiving state from a remote site

-c, --cache='CACHENAME'
The cache name

-s, --site='SITENAME'
The name of the site

5.22.9. PUSH-SITE-STATUS

Shows information about the current push status

-c, --cache='CACHENAME'

The cache name

-s, --site='SITENAME'

The name of the site

SH DESCRIPTION

Shows the version of both the client and the server