Red Hat Data Grid 8.2

Data Grid コマンドラインインターフェイス

CLI を使用してデータにアクセスし、Data Grid を管理します

Last Updated: 2023-08-25

Red Hat Data Grid 8.2 Data Grid コマンドラインインターフェイス

CLI を使用してデータにアクセスし、Data Grid を管理します

法律上の通知

概要

コマンドラインインターフェイス (CLI) から Data Grid Server に接続して、データにアクセスし、管理操作を実行します。

目次

RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリーデータストアです。

スキーマレスデータ構造

さまざまなオブジェクトをキーと値のペアとして格納する柔軟性があります。

グリッドベースのデータストレージ

クラスター間でデータを分散および複製するように設計されています。

エラスティックスケーリング

サービスを中断することなく、ノードの数を動的に調整して要件を満たします。

データの相互運用性

さまざまなエンドポイントからグリッド内のデータを保存、取得、およびクエリーします。

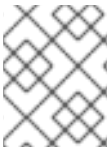
DATA GRID のドキュメント

Data Grid のドキュメントは、Red Hat カスタマーポータルで入手できます。

- Data Grid 8.2 ドキュメント
- Data Grid 8.2 コンポーネントの詳細
- Data Grid 8.2 でサポートされる設定
- Data Grid 8 機能のサポート
- Data Grid で非推奨の機能

DATA GRID のダウンロード

Red Hat カスタマーポータルで Data Grid Software Downloads にアクセスします。



注記

Data Grid ソフトウェアにアクセスしてダウンロードするには、Red Hat アカウントが必要です。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 DATA GRID CLI のスタートガイド

コマンドラインインターフェイス (CLI) を使用すると、Data Grid サーバーにリモートで接続して、データにアクセスし、管理機能を実行できます。

1.1. ユーザーの作成と変更

Data Grid ユーザーの認証情報を追加し、データへのアクセスを制御するパーミッションを割り当てます。

Data Grid サーバーのインストールは、プロパティーレルムを使用して、Hot Rod エンドポイントおよび REST エンドポイントのユーザーを認証します。これは、Data Grid にアクセスする前に 1 人以上のユーザーを作成する必要があることを意味します。

デフォルトでは、ユーザーはキャッシュにアクセスして Data Grid リソースと対話するためのパーミッションを持つロールも必要です。ユーザーにロールを個別に割り当てたり、ロールパーミッションを持つグループにユーザーを追加したりすることができます。

Data Grid コマンドラインインターフェイス (CLI) の **user** コマンドを使用して、ユーザーを作成し、ロールを割り当てます。

ヒント

CLI セッションから **help user** を実行し、コマンドの詳細を取得します。

1.1.1. 認証情報の追加

Data Grid Console の **admin** ユーザーと、Data Grid 環境を完全に制御する必要があります。このため、初めて認証情報を追加する時に **admin** パーミッションを持つユーザーを作成する必要があります。

手順

1. **$RHDG_HOME** でターミナルを開きます。
2. CLI で **user create** コマンドを使用して **admin** ユーザーを作成します。

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

   または、ユーザー名 admin は自動的に **admin** パーミッションを取得します。

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

3. 任意のテキストエディターで、**user.properties** および **groups.properties** を開き、ユーザーおよびグループを確認します。

```
$ 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. ユーザーへのロールの割り当て

ユーザーにロールを割り当て、ユーザーがデータにアクセスし、Data Grid リソースを変更するための適切なパーミッションを持つようにします。

手順

1. **admin** ユーザーで CLI セッションを開始します。

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

2. **deployer** ロールを katie に割り当てます。

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

3. katie のロールを一覧表示します。

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

1.1.3. グループへのユーザーの追加

グループを使用すると、複数のユーザーのパーミッションを変更できます。グループにロールを割り当ててから、そのグループにユーザーを追加します。ユーザーは、グループロールからパーミッションを継承します。

手順

1. **admin** ユーザーで CLI セッションを開始します。
2. **user create** コマンドを使用してグループを作成します。
   a. **--groups** 引数を使用して、グループ名として developers を指定します。
   b. グループのユーザー名とパスワードを設定します。
   プロパティーレルムでは、グループは特別なタイプのユーザーで、ユーザー名とパスワードも必要です。

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

3. グループを一覧表示します。

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

4. **application** ロールを developers グループに割り当てます。

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

5. developers グループのロールを一覧表示します。

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

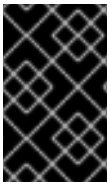
6. 必要に応じて、既存のユーザーを一度に 1 人ずつグループに追加します。

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

1.1.4. ユーザーのロールとパーミッション

Data Grid には、データにアクセスして Data Grid リソースと対話するためのパーミッションをユーザーに付与するデフォルトのロールのセットが含まれています。

ClusterRoleMapper は、Data Grid がセキュリティープリンシパルを承認ロールに関連付けるために使用するデフォルトのメカニズムです。



重要

ClusterRoleMapper は、プリンシパル名をロール名に一致させます。**admin** という名前のユーザーは **admin** パーミッションを自動的に取得し、**deployer** という名前のユーザーは **deployer** パーミッションを取得する、というようになります。

ロール	パーミッション	説明
admin	ALL	Cache Manager ライフサイクルの制御など、すべてのパーミッションを持つスーパーユーザー。
deployer	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR、CREATE	**application** パーミッションに加えて、Data Grid リソースを作成および削除できます。
application	ALL_READ、ALL_WRITE、LISTEN、EXEC、MONITOR	**observer** パーミッションに加え、Data Grid リソースへの読み取りおよび書き込みアクセスがあります。また、イベントをリッスンし、サーバータスクおよびスクリプトを実行することもできます。
observer	ALL_READ、MONITOR	**monitor** パーミッションに加え、Data Grid リソースへの読み取りアクセスがあります。
monitor	MONITOR	JMX および **metrics** エンドポイント経由で統計を表示できます。

参照資料

- org.infinispan.security.AuthorizationPermission Enumeration

- Data Grid Configuration Schema Reference

1.2. DATA GRID SERVER への接続

Data Grid への CLI 接続を確立します。

前提条件

ユーザーの認証情報を追加し、稼働中の Data Grid Server インスタンスが 1 つ以上ある。

手順

1. **$RHDG_HOME** でターミナルを開きます。
2. CLI を起動します。
   - **Linux:**

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

   - **Microsoft Windows:**

     ```
     $ bin\cli.bat
     ```

3. **connect** コマンドを実行し、プロンプトが表示されたらユーザー名とパスワードを入力します。
   - **11222** のデフォルトポート上の Data Grid Server:

     ```
     [disconnected]> connect
     ```

   - ポートオフセットが **100** の Data Grid Server:

     ```
     [disconnected]> connect 127.0.0.1:11322
     ```

1.3. CLI リソースのナビゲート

Data Grid CLI は、Data Grid クラスターリソースの一覧表示、説明、および操作を可能にするナビゲート可能なツリーを公開します。

ヒント

Tab キーを押して、使用可能なコマンドとオプションを表示します。**-h** オプションを使用して、ヘルプテキストを表示します。

Data Grid クラスターに接続すると、デフォルトのキャッシュコンテナーのコンテキストで開きます。

```
[//containers/default]>
```

- **ls** を使用してリソースをリストします。

```
[//containers/default]> ls
```

```
caches
counters
configurations
schemas
tasks
```

- **cd** を使用してリソースツリーをナビゲートします。

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

- **describe** を使用して、リソースに関する情報を表示します。

```
[//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 リソース

Data Grid CLI は、以下の目的でさまざまなリソースを公開します。

- ローカルキャッシュまたはクラスター化キャッシュを作成、変更、および管理します。
- Data Grid クラスターの管理操作を実行します。

キャッシュリソース

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

```
configurations
schemas
```

caches

Data Grid キャッシュインスタンス。デフォルトのキャッシュコンテナーは空です。CLI を使用して、テンプレートまたは **infinispan.xml** ファイルからキャッシュを作成します。

counters

オブジェクトの数を記録する **Strong** カウンターまたは **Weak** カウンター。

configurations

Data Grid 設定。

schemas

キャッシュ内のデータを構造化する Protocol Buffers (Protobuf) スキーマ。

tasks

Data Grid キャッシュ定義を作成および管理するリモートタスク。

クラスターリソース

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

containers

Data Grid クラスター上のキャッシュコンテナー。

cluster

クラスターに参加している Data Grid サーバーを一覧表示します。

server

Data Grid サーバーを管理および監視するためのリソース。

1.4. DATA GRID SERVER のシャットダウン

個別に実行中のサーバーを停止するか、クラスターを正常に停止します。

手順

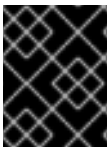
1. Data Grid への CLI 接続を作成します。
2. 次のいずれかの方法で Data Grid Server をシャットダウンします。
   - **shutdown cluster** コマンドを使用して、クラスターのすべてのノードを停止します。以下に例を示します。

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

     このコマンドは、クラスターの各ノードの **data** フォルダーにクラスターの状態を保存します。キャッシュストアを使用する場合、**shutdown cluster** コマンドはキャッシュのすべてのデータも永続化します。

   - **shutdown server** コマンドおよびサーバーのホスト名を使用して、個々のサーバーインスタンスを停止します。以下に例を示します。

```
[//containers/default]> shutdown server <my_server01>
```



重要

shutdown server コマンドは、リバランス操作が完了するまで待機しません。これにより、同時に複数のホスト名を指定すると、データが失われる可能性があります。

ヒント

このコマンドの使用方法の詳細については、**help shutdown** を実行してください。

検証

Data Grid は、サーバーをシャットダウンしたときに以下のメッセージをログに記録します。

```
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 クラスターの再起動

シャットダウン後に Data Grid クラスターをオンラインに戻す場合、クラスターが利用できるのを待ってから、ノードの追加または削除、またはクラスター状態の変更を行う必要があります。

shutdown server コマンドでクラスター化ノードをシャットダウンする場合は、各サーバーを逆の順序で再起動する必要があります。
たとえば、**server1** をシャットダウンしてから、**server2** をシャットダウンする場合は、最初に **server2** を起動してから **server1** を起動する必要があります。

shutdown cluster コマンドでクラスターをシャットダウンすると、すべてのノードが再度参加した後にのみ、クラスターは完全に機能するようになります。
ノードは任意の順序で再起動できますが、シャットダウン前に参加していたすべてのノードが実行されるまで、クラスターは DEGRADED 状態のままになります。

第2章 PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

コマンドラインインターフェイス (CLI) を使用すると、Data Grid サーバーにリモートで接続して、データにアクセスし、管理機能を実行できます。

2.1. DATA GRID コマンドラインインターフェイス (CLI) を使用したキャッシュの作成

Data Grid CLI を使用して、テンプレートから、または XML もしくは JSON 形式の設定ファイルでキャッシュを追加します。

前提条件

ユーザーを作成し、少なくとも 1 つの Data Grid サーバーインスタンスを開始します。

手順

1. Data Grid への CLI 接続を作成します。
2. **create cache** コマンドを使用して、キャッシュ定義を追加します。
   - **--file** オプションを使用して、XML または JSON ファイルからキャッシュ定義を追加します。

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

   - **--template** オプションを使用して、テンプレートからキャッシュ定義を追加します。

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

     ヒント

     --template= 引数の後に Tab キーを押して、利用可能なキャッシュテンプレートを一覧表示します。

3. **ls** コマンドを使用して、キャッシュが存在することを確認します。

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

4. **describe** コマンドを使用して、キャッシュ設定を取得します。

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

参照資料

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

2.1.1. キャッシュ設定

キャッシュ設定は、XML または JSON 形式で提供できます。

XML

```
<distributed-cache name="myCache" mode="SYNC">
  <encoding media-type="application/x-protostream"/>
  <memory max-count="1000000" when-full="REMOVE"/>
</distributed-cache>
```

JSON

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

JSON 形式
JSON 形式のキャッシュ設定は、XML 設定の構造に従う必要があります。* XML 要素が JSON オブジェクトになります。* XML 属性は JSON フィールドになります。

2.2. キャッシュエントリーの追加

データコンテナーに **key:value** ペアのエントリーを作成します。

前提条件

データを保存できる Data Grid キャッシュを作成している。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のように、エントリーをキャッシュに追加します。
   - キャッシュのコンテキストで **put** コマンドを使用します。

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

   - **put** コマンドで **--cache =** を使用します。

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

3. **get** コマンドを使用して、エントリーを確認します。

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

2.3. キャッシュのクリアとエントリーの削除

Data Grid CLI を使用してキャッシュからデータを削除します。

手順

1. Data Grid への CLI 接続を作成します。
2. 次のいずれかを行います。
   - **clearcache** コマンドを使用してすべてのエントリーを削除します。

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

   - **remove** コマンドを使用して特定のエントリーを削除します。

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

2.4. キャッシュの削除

キャッシュをドロップしてキャッシュを削除し、キャッシュに含まれるすべてのデータを削除します。

手順

1. Data Grid への CLI 接続を作成します。
2. **drop** コマンドでキャッシュを削除します。

```
[//containers/default]> drop cache mycache
```

第3章 バッチ操作の実行

インタラクティブに、またはバッチファイルを使用して、グループで操作を処理します。

前提条件

- Data Grid クラスターを実行中である。

3.1. ファイルを使用したバッチ操作の実行

一連の操作を含むファイルを作成し、それらを Data Grid CLI に渡します。

手順

1. 一連の操作を含むファイルを作成している。
   たとえば、**mybatch** という名前のキャッシュを作成する **batch** という名前のファイルを作成し、キャッシュに 2 つのエントリーを追加して、CLI から切断します。

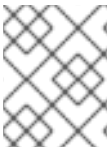
```
connect --username=<username> --password=<password> <hostname>:11222
create cache --template=org.infinispan.DIST_SYNC mybatch
put --cache=mybatch hello world
put --cache=mybatch hola mundo
ls caches/mybatch
disconnect
```

   ヒント

   バッチファイルで直接 **connect** コマンドを使用する代わりに、**autoconnect-url** プロパティーを使用して CLI を設定します。

2. CLI を実行し、ファイルを入力として指定します。

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



注記

CLI バッチファイルは、システムプロパティーの拡張をサポートします。**${property}** 形式を使用する文字列は、**property** システムプロパティーの値に置き換えられます。

3.2. インタラクティブなバッチ操作の実行

標準の入力ストリーム**stdin**を使用して、バッチ操作をインタラクティブに実行します。

手順

1. インタラクティブモードで Data Grid CLI を起動します。

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

ヒント

-c 引数を使用する代わりに、**autoconnect-url** プロパティーを使用して CLI 接続を設定できます。

2. バッチ操作を実行します。以下に例を示します。

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

ヒント

インタラクティブモードでコマンドを追加するには、 **echo** を使用します。

以下の例は、**echo describe** を使用してクラスター情報を取得する方法を示しています。

```
$ 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
  } ]
}
```

第4章 DATA GRID CLI の設定

Data Grid CLI の設定プロパティーを定義します。

4.1. DATA GRID CLI プロパティーと永続ストレージの設定

Data Grid CLI の起動操作を設定し、永続ストレージの場所をカスタマイズします。

前提条件

少なくとも 1 人の Data Grid ユーザーを作成している。

手順

1. オプションで、次のいずれかの方法で Data Grid CLI ストレージディレクトリーへのカスタムパスを設定します。
   - **cli.dir** システムプロパティーの使用:

     ```
     $ bin/cli.sh -Dcli.dir=/path/to/cli/storage ...
     ```

   - **ISPN_CLI_DIR** 環境変数の使用:

     ```
     export ISPN_CLI_DIR=/path/to/cli/storage
     $ bin/cli.sh ...
     ```

2. **config set** コマンドを使用して、設定プロパティーの値を設定します。
   たとえば、CLI が自動的にその URL に接続するように、**autoconnect-url** プロパティーを設定します。

   注記

   リモート接続の場合は、URL を指定し、資格情報を入力します。

   - **http[s]://<username>:<password>@<hostname>:<port>**(Basic 認証用)
   - **http[s]://<token>@<hostname>:<port>**(OAuth 認証用)

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

3. **config get** コマンドで設定プロパティーを確認します。

   ヒント

   help config を実行して、使用可能な設定プロパティーを確認し、使用例を取得します。

4.2. コマンドエイリアスの作成

Data Grid CLI コマンドのエイリアスを作成して、カスタムショートカットを定義します。

手順

1. **alias <alias>=<command>** コマンドを使用してエイリアスを作成します。
たとえば、**quit** コマンドのエイリアスとして **q** を設定します。

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

2. **alias** コマンドを実行して、定義されたエイリアスを確認します。

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

3. **unalias** コマンドを使用してエイリアスを削除します。以下に例を示します。

```
[//containers/default]> unalias q
```

4.3. DATA GRID SERVER 接続の信頼

SSL/TLS 証明書を使用して Data Grid Server への Data Grid CLI 接続を保護します。Data Grid Server の SSL ID としてキーストアを作成する場合、CLI はサーバー証明書を検証して ID を検証できます。

前提条件

- Data Grid Server の SSL ID を設定している。
- 少なくとも 1 人の Data Grid ユーザーを作成している。

手順

1. 次の例のように、サーバーキーストアの場所を指定します。

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

2. 必要に応じて、キーストアのパスワードを次のように定義します。

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

3. CLI 設定を確認します。

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

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

関連情報

- Setting Up SSL Identities for Data Grid Server

4.4. DATA GRID CLI ストレージディレクトリー

Data Grid CLI は、設定を次のデフォルトディレクトリーに保存します。

オペレーティングシステム	デフォルトパス
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**

このディレクトリーには以下のファイルが格納されています。

cli.properties
: CLI 設定プロパティーの値を格納します。

aliases
: コマンドエイリアスを格納します。

history
: CLI 履歴を保存します。

第5章 カウンターの操作

カウンターは、オブジェクトの数を記録するアトミック増減分操作を提供します。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

5.1. カウンターの作成

Data Grid CLI を使用して強力なカウンターと弱いカウンターを作成します。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切な引数を指定して **create counter** コマンドを実行します。
   a. **my-weak-counter** を作成します。

```
[//containers/default]> create counter --concurrency-level=1 --initial-value=5 --
storage=PERSISTENT --type=weak my-weak-counter
```

   b. **my-strong-counter** を作成します。

```
[//containers/default]> create counter --initial-value=3 --storage=PERSISTENT --
type=strong my-strong-counter
```

3. 使用可能なカウンターを一覧表示します。

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

4. カウンター設定を確認します。
   a. **my-weak-counter** について説明します。

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

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

   b. **my-strong-counter** について説明します。

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

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

5.2. カウンターへのデルタの追加

任意の値でカウンターに増分または減分を適用します。

手順

1. カウンターを選択します。

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

2. 現在のカウントを一覧表示します。

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

3. カウンター値を **2** 増やします。

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

4. カウンター値を **-4** 減らします。

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

注記

強力なカウンターは、演算が適用された後に値を返します。**--quiet = true** を使用して、戻り値を非表示にします。

たとえば、**my-strong-counter]> add --delta = 3 --quiet = true**。

弱いカウンターは空の応答を返します。

第6章 PROTOBUF メタデータを使用したキャッシュのクエリー

Data Grid は、Protocol Buffer (Protobuf) を使用してキャッシュ内のデータを構造化し、クエリーを実行できるようにすることをサポートしています。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

6.1. メディアタイプの設定

さまざまなメディアタイプのキャッシュエントリーをエンコードして、要件に最適な形式でデータを保存します。

たとえば、次の手順は、**application/x-protostream** メディアタイプを設定する方法を示しています。

手順

1. **qcache** という名前の分散キャッシュを追加し、メディアタイプを設定する Data Grid 設定ファイルを作成します。次に例を示します。

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

2. **--file=** オプションを使用して、**pcache.xml** から **pcache** を作成します。

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

3. **pcache** を確認します。

```
[//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. **pcache** にエントリーを追加し、エンコードを確認します。

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

6.2. PROTOBUF スキーマの登録

Protobuf スキーマには、メッセージと呼ばれるデータ構造が **.proto** 定義ファイルに含まれています。

手順

1. 次のメッセージを含む **person.proto** という名前のスキーマファイルを作成します。

```
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. **person.proto** を登録します。

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

3. **person.proto** を確認します。

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

6.3. PROTOBUF スキーマを使用したキャッシュのクエリー

Data Grid は、JSON を Protobuf に自動的に変換するため、キャッシュエントリーを JSON 形式で読み書きし、Protobuf スキーマを使用してクエリーを実行できます。

たとえば、次の JSON ドキュメントについて考えてみます。

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":[]
}
```

前述の各 JSON ドキュメントには次が含まれます。

- JSON ドキュメントが対応する Protobuf メッセージを識別する **_type** フィールド。
- **person.proto** スキーマのデータ型に対応するいくつかのフィールド。

手順

1. **pcache** キャッシュに移動します。

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

2. 各 JSON ドキュメントをエントリーとしてキャッシュに追加します。次に例を示します。

```
[//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. エントリーが存在することを確認します。

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

4. キャッシュをクエリーして、性別データ型が **MALE** である Protobuf **Person** エンティティーからエントリーを返します。

```
[//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"
      }
    }
  } ]
}
```

第7章 クロスサイトレプリケーション操作の実行

異なる場所で実行されている Data Grid クラスターは、データをバックアップするために相互に検出および通信できます。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

7.1. バックアップ場所のオフラインおよびオンライン化

バックアップ場所を手動でオフラインにし、オンラインに戻します。

手順

1. Data Grid への CLI 接続を作成します。
2. **site status** コマンドを使用して、バックアップの場所がオンラインかオフラインかを確認します。

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

注記

--site はオプションの引数です。設定されていない場合、CLI はすべてのバックアップ場所を返します。

3. 次のようにバックアップ場所を管理します。
   - **bring-online** コマンドを使用して、バックアップの場所をオンラインにします。

```
[//containers/default]> site bring-online --cache=customers --site=NYC
```

   - **take-offline** コマンドを使用して、バックアップの場所をオフラインにします。

```
[//containers/default]> site take-offline --cache=customers --site=NYC
```

詳細と例については、**help site** コマンドを実行してください。

7.2. サイト間の状態転送モードの設定

バックアップの場所がオンラインになったことを DataGrid が検出したときに自動的に発生するように、サイト間の状態転送操作を設定できます。または、デフォルトモードを使用することもできます。これは、CLI を介して、または JMX または REST を介して手動で状態転送を実行するものです。

手順

1. Data Grid への CLI 接続を作成します。
2. 次の例のように、**site** コマンドを使用して状態転送モードを設定します。

- 現在の状態転送モードを取得します。

```
[//containers/default]> site state-transfer-mode get --cache=cacheName --site=NYC
"MANUAL"
```

- キャッシュとバックアップの場所の自動状態転送操作を設定します。

```
[//containers/default]> site state-transfer-mode set --cache=cacheName --site=NYC --mode=AUTO
```

ヒント

詳細と例については、**help site** コマンドを実行してください。

7.3. バックアップ場所への状態のプッシュ

キャッシュの状態をリモートのバックアップ場所に転送します。

手順

1. Data Grid への CLI 接続を作成します。
2. 次の例のように、**site** コマンドを使用して状態の転送をプッシュします。

```
[//containers/default]> site push-site-state --cache=cacheName --site=NYC
```

詳細と例については、**help site** コマンドを実行してください。

第8章 DATA GRID クラスターのバックアップおよび復元

キャッシュされたエントリー、キャッシュ設定、Protobuf スキーマ、およびサーバースクリプトを含む Data Grid リソースのアーカイブを作成します。その後、バックアップアーカイブを使用して、再起動または移行後に Data Grid Server クラスターを復元できます。

前提条件

- Data Grid CLI を起動している。
- 実行中の Data Grid クラスターに接続している。

8.1. DATA GRID クラスターのバックアップ

ダウンロードまたは Data Grid Server に保存できる **.zip** 形式のバックアップアーカイブを作成します。

前提条件

バックアップアーカイブは、最新のクラスター状態を反映している必要があります。このため、バックアップアーカイブを作成する前に、クラスターが書き込み要求を受け付けていないことを確認する必要があります。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切なオプションを指定して **backup create** コマンドを実行します。以下に例を示します。
   - 自動生成された名前ですべてのリソースをバックアップします。

     ```
     [//containers/default]> backup create
     ```

   - **example-backup** という名前のバックアップアーカイブにすべてのリソースをバックアップします。

     ```
     [//containers/default]> backup create -n example-backup
     ```

   - サーバー上の **/some/server/dir** パスにすべてのリソースをバックアップします。

     ```
     [//containers/default]> backup create -d /some/server/dir
     ```

   - キャッシュとキャッシュテンプレートのみをバックアップします。

     ```
     [//containers/default]> backup create --caches=* --templates=*
     ```

   - 指定した Protobuf スキーマのみをバックアップします。

     ```
     [//containers/default]> backup create --proto-schemas=schema1,schema2
     ```

3. サーバー上で利用可能なバックアップアーカイブを一覧表示します。

   ```
   [//containers/default]> backup ls
   ```

4. サーバーから **example-backup** アーカイブをダウンロードします。
バックアップ操作がまだ進行中の場合、コマンドはバックアップ操作が完了するのを待ちます。

```
[//containers/default]> backup get example-backup
```

5. オプションで、 **example-backup** アーカイブをサーバーから削除します。

```
[//containers/default]> backup delete example-backup
```

8.2. バックアップアーカイブからの DATA GRID クラスターの復元

バックアップアーカイブのコンテンツを Data Grid クラスターに適用して、バックアップされた状態に復元します。

前提条件

- Data Grid CLI のローカルか、Data Grid Server に保存されているバックアップアーカイブを作成している。
- ターゲットコンテナーがバックアップアーカイブのコンテナー名と一致していることを確認してください。コンテナー名が一致しない場合、バックアップを復元することはできません。

手順

1. Data Grid への CLI 接続を作成します。
2. 適切なオプションを指定して **backup restore** コマンドを実行します。
   - サーバーでアクセス可能なバックアップアーカイブからすべてのコンテンツを復元します。

```
[//containers/default]> backup restore /some/path/on/the/server
```

   - ローカルバックアップアーカイブからすべてのコンテンツを復元します。

```
[//containers/default]> backup restore -u /some/local/path
```

   - サーバー上のバックアップアーカイブからキャッシュコンテンツのみを復元します。

```
[//containers/default]> backup restore /some/path/on/the/server --caches=*
```

第9章 コマンドリファレンス

Data Grid CLI コマンドのマニュアルページを確認してください。

ヒント

help コマンドを使用して、CLI セッションから直接マニュアルページにアクセスします。

たとえば、**get** コマンドのマニュアルページを表示するには、次の手順を実行します。

```
$ help get
```

9.1. ADD(1)

9.1.1. 名前

add - 任意の値でカウンターに増分または減分を適用します。

9.1.2. 概要

add ['OPTIONS'] ['COUNTER_NAME']

9.1.3. オプション

--delta='nnn'
　カウンター値を増加または減少させるデルタを設定します。デフォルトは **1** です。

-q, --quiet='[true|false]'
　強力なカウンターの戻り値を非表示にします。デフォルトは **false** です。

9.1.4. 例

add --delta=10 cnt_a
cnt_a の値を **10** 増やします。

add --delta=-5 cnt_a
cnt_a の値を **5** 減らします。

9.1.5. 関連項目

cas(1)、reset(1)

9.2. ALIAS(1)

9.2.1. 名前

alias - エイリアスを作成または表示します。

9.2.2. 概要

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

9.2.3. 例

alias q=quit
quit コマンドのエイリアスとして **q** を作成します。

alias
定義されているすべてのエイリアスを一覧表示します。

9.2.4. 関連項目

config(1)、unalias(1)

9.3. BACKUP(1)

9.3.1. 名前

backup - コンテナーのバックアップ作成と復元を管理します。

9.3.2. 概要

backup create ['OPTIONS']

backup delete ['OPTIONS'] **BACKUP_NAME**

backup get ['OPTIONS'] **BACKUP_NAME**

backup ls

backup restore ['OPTIONS'] **BACKUP_PATH**

9.3.3. バックアップ作成のオプション

-d, --dir='PATH'
バックアップアーカイブを作成および保存するサーバー上のディレクトリーを指定します。

-n, --name='NAME'
バックアップアーカイブの名前を定義します。

--caches='cache1,cache2,...'
バックアップするキャッシュを一覧表示します。'*'を使用して、すべてのキャッシュをバックアップします。

--templates='template1,template2,...'
バックアップするキャッシュテンプレートを一覧表示します。'*'を使用して、すべてのテンプレートをバックアップします。

--counters='counter1,counter2,...'
バックアップするカウンターを一覧表示します。'*'を使用して、すべてのカウンターをバックアップします。

--proto-schemas='schema1,schema2,...'
バックアップする Protobuf スキーマを一覧表示します。'*'を使用して、すべてのスキーマをバックアップします。

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

バックアップするサーバータスクを一覧表示します。'*'を使用して、すべてのタスクをバックアップします。

9.3.4. バックアップ取得のオプション

--no-content

コンテンツをダウンロードしません。このコマンドは、バックアップ操作が完了したときにのみ返します。

9.3.5. バックアップ復元のオプション

-u, --upload

サーバーにアップロードされるローカルバックアップアーカイブへのパスを定義します。

-n, --name='NAME'

復元要求の名前を定義します。

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

復元するキャッシュを一覧表示します。'*'を使用して、バックアップアーカイブからすべてのキャッシュを復元します。

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

復元するキャッシュテンプレートを一覧表示します。'*'を使用して、バックアップアーカイブからすべてのテンプレートを復元します。

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

復元するカウンターを一覧表示します。'*'を使用して、バックアップアーカイブからすべてのカウンターを復元します。

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

復元する Protobuf スキーマを一覧表示します。'*'を使用して、バックアップアーカイブからすべてのスキーマを復元します。

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

復元するサーバータスクを一覧表示します。'*'を使用して、バックアップアーカイブからすべてのタスクを復元します。

9.3.6. 例

backup create -n example-backup
example-backup という名前ですべてのコンテナーコンテンツのバックアップを開始します。

backup create -d /some/server/dir
すべてのコンテナーコンテンツのバックアップを開始し、サーバーのパス **/some/server/dir** に保存します。

backup create --caches=* --templates=*
キャッシュとキャッシュ設定リソースのみを含むバックアップを開始します。

backup create --proto-schemas=schema1,schema2
指定されたスキーマリソースのみを含むバックアップを開始します。

backup ls
サーバーで使用可能なすべてのバックアップを一覧表示します。

backup get example-backup
example-backup アーカイブをサーバーからダウンロードします。バックアップ操作が進行中の場合、コマンドはバックアップ操作が完了するのを待ちます。

backup restore /some/path/on/the/server
サーバー上のバックアップアーカイブからすべてのコンテンツを復元します。

backup restore -u /some/local/path
サーバーにアップロードされたローカルバックアップアーカイブからすべてのコンテンツを復元します。

backup restore /some/path/on/the/server --caches=*
サーバー上のバックアップアーカイブからキャッシュコンテンツのみを復元します。

backup restore /some/path/on/the/server --proto-schemas=schema1,schema2
サーバー上のバックアップアーカイブから、指定されたスキーマリソースのみを復元します。

backup delete example-backup
example-backup アーカイブをサーバーから削除します。

9.3.7. 関連項目

drop(1)

9.4. BENCHMARK(1)

9.4.1. 名前

benchmark - キャッシュに対してパフォーマンスベンチマークを実行します。

HTTP および HotRod プロトコル **http**、**https**、**hotrod**、および **hotrods** のパフォーマンスベンチマークを実行できます。URI を使用してベンチマークのプロトコルを指定します。プロトコルを指定しない場合、ベンチマークは現在の CLI 接続の URI を使用します。

Hot Rod URI のベンチマークは、クラスター全体に接続します。HTTP URI の場合、ベンチマークは単一のノードにのみ接続します。

ベンチマークは、既存のキャッシュに対してパフォーマンスをテストします。ベンチマークを実行する前に、測定する機能を備えたキャッシュを作成する必要があります。たとえば、クロスサイトレプリケーションのパフォーマンスを評価する場合は、バックアップの場所を持つキャッシュを作成する必要があります。永続性のパフォーマンスをテストする場合は、適切なキャッシュストアを使用するキャッシュを作成します。

9.4.2. 概要

benchmark ['OPTIONS'] [**uri**]

9.4.3. ベンチマークのオプション

-t, --threads='num'
作成するスレッドの数を指定します。デフォルトは **10** です。

--cache='cache'

ベンチマークが実行されるキャッシュの名前を指定します。デフォルトは **benchmark** です。キャッシュがまだ存在しない場合は、ベンチマークを実行する前にキャッシュを作成する必要があります。

*--key-size='num'

キーのサイズをバイト単位で設定します。デフォルトは 16 バイトです。

*--value-size='num'

値のサイズをバイト単位で設定します。デフォルトは 1000 バイトです。

*--keyset-size='num'

テストキーセットのサイズをバイト単位で定義します。デフォルトは **1000** です。

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

出力の詳細レベルを指定します。可能な値は、最も簡素なものから最も詳細なものまで、**SILENT**、**NORMAL**、および **EXTRA** です。デフォルトは **NORMAL** です。

-c, --count='num'

実行する測定の反復回数を指定します。デフォルトは **5** です。

--time='time'

各反復にかかる時間を秒単位で設定します。デフォルトは **10** です。

--warmup-count='num'

実行するウォームアップの反復回数を指定します。デフォルトは **5** です。

--warmup-time='time'

各ウォームアップの反復にかかる時間を秒単位で設定します。デフォルトは **1** です。

--mode='mode'

ベンチマークモードを指定します。可能な値は、**Throughput**、**AverageTime**、**SampleTime**、**SingleShotTime**、および **All** です。デフォルトは **Throughput** です。

--time-unit='unit'

ベンチマークレポートの結果の時間単位を指定します。可能な値は、**NANOSECONDS**、**MICROSECONDS**、**MILLISECONDS**、および **SECONDS** です。デフォルトは **MICROSECONDS** です。

9.4.4. 例

benchmark hotrod://localhost:11222
HotRod プロトコルを使用してベンチマークテストを実行します。

benchmark --value-size=10000 --cache=largecache hotrod://localhost:11222
サイズが 10000 バイトのテスト値を使用して、**largecache** キャッシュに対して HotRod プロトコルでベンチマークテストを実行します。

benchmark --mode=All --threads=20 https://user:password@server:11222
20 スレッドを使用して HTTPS プロトコルでベンチマークテストを実行し、レポートにすべてのモードを含めます。

9.5. CACHE(1)

9.5.1. 名前

cache - 後続のコマンドのデフォルトキャッシュを選択します。

9.5.2. 概要

cache ['CACHE_NAME']

9.5.3. 例

cache mycache
mycache を選択ます。**cd caches/mycache** を使用してリソースツリーをナビゲートするのと同じです。

9.5.4. 関連項目

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

9.6. CAS(1)

9.6.1. 名前

cas - 強力なカウンターで'compare-and-swap'操作を実行します。

9.6.2. 概要

cas ['OPTIONS'] ['COUNTER_NAME']

9.6.3. オプション

--expect='nnn'
　カウンターの期待値を指定します。
--value='nnn'
　カウンターに新しい値を設定します。
-q, --quiet='[true|false]'
　戻り値を非表示にします。デフォルトは false です。

9.6.4. 例

cas --expect=10 --value=20 cnt_a
現在の値が **10** の場合にのみ、**cnt_a** の値を **20** に設定します。

9.6.5. 関連項目

add(1)、cas(1)、reset(1)

9.7. CD(1)

9.7.1. 名前

cd - サーバーリソースツリーをナビゲートします。

9.7.2. 説明

PATH は、絶対パスまたは現在のリソースに対する相対パスです。**../** は親リソースを指定します。

9.7.3. 概要

cd ['PATH']

9.7.4. 例

cd caches
リソースツリーの **caches** パスに変更します。

9.7.5. 関連項目

cache(1)、ls(1)、container(1)

9.8. CLEARCACHE(1)

9.8.1. 名前

clearcache - キャッシュからすべてのエントリーを削除します。

9.8.2. 概要

clearcache ['CACHE_NAME']

9.8.3. 例

clearcache mycache
mycache からすべてのエントリーを削除します。

9.8.4. 関連項目

cache(1)、drop(1)、remove(1)

9.9. CONFIG(1)

9.9.1. 名前

config - CLI 設定プロパティーを管理します。

9.9.2. 概要

config

config set 'name' 'value'

config get 'name'

9.9.3. 説明

CLI 設定プロパティーを管理 (リスト、設定、取得) します。

9.9.4. コマンドの概要

config
　設定されているすべての設定プロパティーを一覧表示します。

config set 'name' ['value']
　特定のプロパティーの値を設定します。値を指定しない場合、プロパティーは設定されません。

config get 'name'
　特定のプロパティーの値を取得します。

9.9.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help
　コマンドまたはサブコマンドのヘルプページを表示します。

9.9.6. プロパティー

autoconnect-url
　起動時に CLI が自動的に接続する URL を指定します。

autoexec
　起動時に実行する CLI バッチファイルのパスを指定します。

trustall
　すべてのサーバー証明書を信頼するかどうかを指定します。値は **false**(デフォルト) および **true** です。

truststore
　サーバー ID を検証する証明書チェーンを含むキーストアへのパスを定義します。

truststore-password
　キーストアにアクセスするためのパスワードを指定します。

9.9.7. 例

config set autoconnect-url http://192.0.2.0:11222
CLI の起動時に、カスタム IP アドレスでサーバーに接続します。

config get autoconnect-url
autoconnect-url 設定プロパティーの値を返します。

config set autoexec /path/to/mybatchfile
CLI の起動時に、"mybatchfile"という名前のバッチファイルを実行します。

config set trustall true
すべてのサーバー証明書を信頼します。

config set truststore /home/user/my-trust-store.jks
"my-trust-store.jks"という名前のキーストアのパスを指定します。

config set truststore-password secret
必要に応じて、キーストアのパスワードを設定します。

9.9.8. 関連項目

alias(1)、unalias(1)

9.10. CONNECT(1)

9.10.1. 名前

connect - 実行中の Data Grid サーバーに接続します。

9.10.2. 説明

デフォルトは **http://localhost:11222** で、認証が必要な場合は資格情報の入力を求められます。

9.10.3. 概要

connect ['OPTIONS'] ['SERVER_LOCATION']

9.10.4. オプション

-u, --username='USERNAME'

Data Grid サーバーで認証するユーザー名を指定します。

-p, --password='PASSWORD'

パスワードを指定します。

9.10.5. 例

connect 127.0.0.1:11322 -u test -p changeme
100 のポートオフセットとサンプルの資格情報を使用して、ローカルで実行されているサーバーに接続します。

9.10.6. 関連項目

disconnect(1)

9.11. CONTAINER(1)

9.11.1. 名前

container - 後続のコマンドを実行するためのコンテナーを選択します。

9.11.2. 概要

container ['CONTAINER_NAME']

9.11.3. 例

container default
デフォルトのコンテナーを選択します。これは、**cd containers/default** を使用してリソースツリーをナビゲートするのと同じです。

9.11.4. 関連項目

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

9.12. COUNTER(1)

9.12.1. 名前

counter - 後続のコマンドのデフォルトカウンターを選択します。

9.12.2. 概要

counter ['COUNTER_NAME']

9.12.3. 例

counter cnt_a
cnt_a を選択します。**cd counters/cnt_a** を使用してリソースツリーをナビゲートするのと同じです。

9.12.4. 関連項目

add(1)、cas(1)

9.13. CREATE(1)

9.13.1. 名前

create - Data Grid サーバーにキャッシュとカウンターを作成します。

9.13.2. 概要

create cache ['OPTIONS'] **CACHE_NAME**

create counter ['OPTIONS'] **COUNTER_NAME**

9.13.3. キャッシュ作成のオプション

-f, --file='FILE'
JSON または XML 形式の設定ファイルを指定します。

-t, --template='TEMPLATE'
設定テンプレートを指定します。タブのオートコンプリートを使用して、使用可能なテンプレートを表示します。

-v, --volatile='[true|false]'
キャッシュが永続的であるか揮発性であるかを指定します。デフォルトは false です。

9.13.4. カウンター作成のオプション

-t, --type='[weak|strong]'
カウンターが弱いか強いかを指定します。

-s, --storage='[PERSISTENT|VOLATILE]'
　カウンターが永続的であるか揮発性であるかを指定します。

-c, --concurrency-level='nnn'
　カウンターの同時並行性レベルを設定します。

-i, --initial-value='nnn'
　カウンターの初期値を設定します。

-l, --lower-bound='nnn'
　強力なカウンターの下限を設定します。

-u, --upper-bound='nnn'
　強力なカウンターの上限を設定します。

9.13.5. 例

create cache --template=org.infinispan.DIST_SYNC mycache
DIST_SYNC テンプレートから **MyCache** という名前のキャッシュを作成します。

create counter --initial-value=3 --storage=PERSISTENT --type=strong cnt_a
cnt_a という名前の強力なカウンターを作成します。

9.13.6. 関連項目

drop(1)

9.14. CREDENTIALS(1)

9.14.1. 名前

credentials - Data Grid Server のクレデンシャルを含むキーストアを管理します

9.14.2. 概要

credentials ls

credentials add 'alias'

credentials remove 'alias'

9.14.3. 説明

キーストア内の資格情報を一覧表示、作成、および削除します。デフォルトでは、コマンドはサーバー設定ディレクトリーの **credentials.pfx** キーストアを管理します。

9.14.4. 概要

credentials ls
　キーストアに保存されている資格情報のエイリアスを一覧表示します。

クレデンシャルの追加

credentials add 'alias'

エイリアスと対応するクレデンシャルをキーストアに追加します。

クレデンシャルの削除

credentials remove 'alias'
エイリアスと対応するクレデンシャルをキーストアから削除します。

9.14.5. オプション

-h, --help
コマンドのヘルプを出力します。

-s, --server-root='path-to-server-root'
サーバーのルートディレクトリーへのパスを指定します。デフォルトは **server** です。

--path='credentials.pfx'
クレデンシャルキーストアへのパスを指定します。デフォルトはサーバー設定ディレクトリー **server/conf** です。

-p, --password='password'
クレデンシャルキーストアのパスワードを指定します。

-t, --type='PKCS12'
資格情報を含むキーストアのタイプを指定します。サポートされているタイプは **PKCS12** または **JCEKS** です。デフォルトは **PKCS12** です。

9.14.6. クレデンシャル追加のオプション

-c, --credential='credential'
保存する資格情報を指定します。

9.14.7. 例

credentials add dbpassword -c changeme -p "secret1234!"
資格情報キーストアがまだ存在しない場合は、新しいデフォルトの資格情報キーストアを作成し、"changeme"のパスワードに"dbpassword"のエイリアスを追加します。このコマンドはクレデンシャルキーストアのパスワードとして"secret1234!"も設定します。このパスワードは、サーバー設定のパスワード **<clear-text-credential clear-text="secret1234!"/>** と一致する必要があります。

credentials ls -p "secret1234!"
デフォルトのクレデンシャルキーストア内のすべてのエイリアスを一覧表示します。

credentials add ldappassword -t JCEKS -p "secret1234!"
JCEKS 形式で資格情報キーストアを作成し、エイリアス"ldappassword"を追加します。このコマンドは、エイリアスに対応するパスワードを指定するように要求します。

9.15. DESCRIBE(1)

9.15.1. 名前

describe - リソースに関する情報を表示します。

9.15.2. 概要

describe ['PATH']

9.15.3. 例

describe //containers/default
デフォルトのコンテナーに関する情報を表示します。

describe //containers/default/caches/mycache
mycache キャッシュに関する情報を表示します。

describe //containers/default/caches/mycache/k1
k1 キーに関する情報を表示します。

describe //containers/default/counters/cnt1
cnt1 カウンターに関する情報を表示します。

9.15.4. 関連項目

cd(1)、ls(1)

9.16. DISCONNECT(1)

9.16.1. 名前

disconnect - Data Grid サーバーとの CLI セッションを終了します。

9.16.2. 概要

disconnect

9.16.3. 例

disconnect
現在の CLI セッションを終了します。

9.16.4. 関連項目

connect(1)

9.17. DROP(1)

9.17.1. 名前

drop - キャッシュとカウンターを削除します。

9.17.2. 概要

drop cache CACHE_NAME

drop counter COUNTER_NAME

9.17.3. 例

drop cache mycache
mycache キャッシュを削除します。

drop counter cnt_a
cnt_a カウンターを削除します。

9.17.4. 関連項目

create(1)、clearcache(1)

9.18. ENCODING(1)

9.18.1. 名前

encoding - キャッシュエントリーのエンコーディングを表示および設定します。

9.18.2. 説明

キャッシュに対する**put**および**get**操作のデフォルトのエンコーディングを設定します。引数が指定されていない場合、**encoding**コマンドは現在のエンコーディングを表示します。

有効なエンコーディングでは、次のような標準の MIME タイプ (IANA メディアタイプ) の命名規則が使用されます。

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

9.18.3. 概要

encoding ['ENCODING']

9.18.4. 例

encoding application/json
エントリーを **application/json** としてエンコードするように、現在選択されているキャッシュを設定します。

9.18.5. 関連項目

get(1)、put(1)

9.19. GET(1)

9.19.1. 名前

get - キャッシュからエントリーを取得します。

9.19.2. 概要

get ['OPTIONS'] **KEY**

9.19.3. オプション

-c, --cache='NAME'

エントリーを取得するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

9.19.4. 例

get hello -c mycache
mycache から **hello** という名前のキーの値を取得します。

9.19.5. 関連項目

query(1)、put(1)

9.20. HELP(1)

9.20.1. 名前

help - コマンドのマニュアルページを出力します。

9.20.2. 概要

help ['COMMAND']

9.20.3. 例

help get
getコマンドのマニュアルページを出力します。

9.20.4. 関連項目

version(1)

9.21. LOGGING(1)

9.21.1. 名前

logging - Data Grid サーバーのランタイムロギング設定を検査および操作します。

9.21.2. 概要

logging list-loggers

logging list-appenders

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

logging remove **LOGGER_NAME**

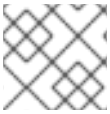
9.21.3. ロギング設定のオプション

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

特定のロガーのログレベルを指定します。

-a, --appender='APPENDER'

特定のロガーに設定するアペンダーを指定します。このオプションは、複数のアペンダーに対して繰り返すことができます。



注記

ロガー名なしで**logging set**を呼び出すと、ルートロガーが変更されます。

9.21.4. 例

logging list-loggers
利用可能なすべてのロガーを一覧表示します。

logging set --level=DEBUG --appenders=FILE org.infinispan
org.infinispan ロガーのログレベルを **DEBUG** に設定し、**FILE** アペンダーを使用するように設定します。

9.22. LS(1)

9.22.1. 名前

ls - 現在のパスまたは特定のパスのリソースを一覧表示します。

9.22.2. 概要

ls ['PATH']

9.22.3. 例

ls caches
使用可能なキャッシュを一覧表示します。

ls ../
親リソースを一覧表示します。

9.22.4. 関連項目

cd(1)

9.23. MIGRATE(1)

9.23.1. 名前

migrate - Data Grid のあるバージョンから別のバージョンにデータを移行します。

9.23.2. 概要

migrate cluster synchronize

migrate cluster disconnect

9.23.3. 説明

あるバージョンの Data Grid から別のバージョンにデータを移行するには、 **migrate** コマンドを使用します。

9.23.4. コマンドの概要

クラスターの移行

migrate cluster synchronize
　ソースクラスターとターゲットクラスターの間でデータを同期します。

migrate cluster disconnect
　ターゲットクラスターをソースクラスターから切断します。

9.23.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help
　コマンドまたはサブコマンドのヘルプページを表示します。

9.23.6. クラスター同期オプション

-c, --cache='name'
　同期するキャッシュの名前。

-b, --read-batch='num'
　バッチで処理するエントリーの量。デフォルトは 10000 です。

-t, --threads='num'
　使用するスレッドの数。デフォルトはサーバーのコア数です。

9.23.7. クラスター切断オプション

-c, --cache='name'
　ソースから切断するキャッシュの名前。

9.24. PATCH(1)

9.24.1. 名前

patch - サーバーパッチを管理します。

9.24.2. 説明

サーバーパッチを一覧表示、説明、インストール、ロールバック、および作成します。

パッチは、サーバーをアップグレードして問題を解決したり、新しい機能を追加したりするためのアーティファクトを含む zip アーカイブファイルです。パッチは、異なるバージョンの複数のサーバーインストールにターゲットバージョンを適用できます。

9.24.3. 概要

patch ls

patch install 'patch-file'

patch describe 'patch-file'

patch rollback

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

9.24.4. パッチリストのオプション

--server='path/to/server'
　現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

-v、--verbose
　個々のファイルに関する情報を含む、インストールされている各パッチの内容を表示します。

9.24.5. パッチインストールのオプション

--dry-run
　パッチが変更を適用せずに実行する操作を示します。

--server='path/to/server'
　現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

9.24.6. パッチ説明のオプション

-v、--verbose
　個々のファイルに関する情報を含む、パッチの内容を表示します。

9.24.7. パッチロールバックのオプション

--dry-run
　パッチが変更を適用せずに実行する操作を示します。

--server='path/to/server'
　現在のサーバーのホームディレクトリー外のターゲットサーバーへのパスを設定します。

9.24.8. パッチ作成のオプション

-q, --qualifier='name'
　パッチの説明的な修飾子文字列を指定します (例:'one-off for issue nnnn')。

9.24.9. 例

patch ls
サーバーに現在インストールされているパッチをインストール順に一覧表示します。

patch install mypatch.zip
サーバーの現在のディレクトリーに"mypatch.zip"をインストールします。

patch install mypatch.zip --server=/path/to/server/home
サーバーの別のディレクトリーに"mypatch.zip"をインストールします。

patch describe mypatch.zip
"mypatch.zip"のターゲットバージョンとソースバージョンのリストを表示します。

patch create mypatch.zip 'target-server' 'source-server-1' ['source-server-2'...]
ターゲットサーバーのバージョンを使用し、ソースサーバーのバージョンに適用する"mypatch.zip"という名前のパッチファイルを作成します。

patch rollback
サーバーに適用された最後のパッチをロールバックし、以前のバージョンを復元します。

9.25. PUT(1)

9.25.1. 名前

put - キャッシュエントリーを追加または更新します。

9.25.2. 説明

新しいキーのエントリーを作成します。既存のキーの値を置き換えます。

9.25.3. 概要

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

9.25.4. オプション

-c, --cache='NAME'
キャッシュの名前を指定します。デフォルトは現在選択されているキャッシュです。

-e, --encoding='ENCODING'
値のメディアタイプを設定します。

-f, --file='FILE'
エントリーの値を含むファイルを指定します。

-l, --ttl='TTL'
エントリーが自動的に削除されるまでの秒数 (存続時間) を設定します。**0** の場合または指定されていない場合、デフォルトはキャッシュ設定の **lifespan** の値になります。負の値を設定すると、エントリーが削除されることはありません。

-i, --max-idle='MAXIDLE'
エントリーをアイドル状態にできる秒数を設定します。最大アイドル時間が経過してもエントリーの読み取りまたは書き込み操作が発生しない場合、エントリーは自動的に削除されます。**0** の場合または指定されていない場合、デフォルトはキャッシュ設定の **maxIdle** の値になります。負の値を設定すると、エントリーが削除されることはありません。

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

エントリーが存在しない場合にのみエントリーを配置します。

9.25.5. 例

put -c mycache hello world
値が **world** の **hello** キーを **mycache** キャッシュに追加します。

put -c mycache -f myfile -i 500 hola
値が **myfile** の内容の **hola** キーを追加します。また、最大アイドル時間を **500** 秒に設定します。

9.25.6. 関連項目

get(1)、remove(1)

9.26. QUERY(1)

9.26.1. 名前

query - Ickle クエリー文字列に一致するエントリーを取得します。

9.26.2. 概要

query ['OPTIONS'] **QUERY_STRING**

9.26.3. オプション

-c, --cache='NAME'
照会するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

--max-results='MAX_RESULTS'
返す結果の数を設定します。デフォルトは **10** です。

-o, --offset='OFFSET'
返される最初の結果のインデックスを指定します。デフォルトは **0** です。

--query-mode='QUERY_MODE'
サーバーがクエリーを実行する方法を指定します。値は **FETCH** と BROADCAST です。デフォルトは **FETCH** です。

9.26.4. 例

query "from org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
現在選択されているキャッシュをクエリーして、性別データ型が **MALE** である Protobuf **Person** エンティティーからエントリーを返します。

9.26.5. 関連項目

schema(1)

9.27. QUIT(1)

9.27.1. 名前

quit - コマンドラインインターフェイスを終了します。

9.27.2. 概要

quit

exitと**bye**はコマンドエイリアスです。

9.27.3. 例

quit
CLI セッションを終了します。

exit
CLI セッションを終了します。

bye
CLI セッションを終了します。

9.27.4. 関連項目

disconnect(1)、shutdown(1)

9.28. REMOVE(1)

9.28.1. 名前

remove - キャッシュからエントリーを削除します。

9.28.2. 概要

remove KEY ['OPTIONS']

9.28.3. オプション

--cache='NAME'
エントリーを削除するキャッシュを指定します。デフォルトは現在選択されているキャッシュです。

9.28.4. 例

remove --cache=mycache hola
mycache キャッシュから **hola** エントリーを削除します。

9.28.5. 関連項目

cache(1)、drop(1)、clearcache(1)

9.29. RESET(1)

9.29.1. 名前

reset - カウンターの初期値を復元します。

9.29.2. 概要

reset ['COUNTER_NAME']

9.29.3. 例

reset cnt_a
cnt_a カウンターをリセットします。

9.29.4. 関連項目

add(1)、cas(1)、drop(1)

9.30. SCHEMA(1)

9.30.1. 名前

schema - protobuf スキーマをアップロードして登録します。

9.30.2. 概要

schema ['OPTIONS'] **SCHEMA_NAME**

9.30.3. オプション

-u, --upload='FILE'
　指定された名前の protobuf スキーマとしてファイルをアップロードします。

9.30.4. 例

schema --upload=person.proto person.proto
Protobuf スキーマ **person.proto** を登録します。

9.30.5. 関連項目

query(1)

9.31. SERVER(1)

9.31.1. 名前

server - サーバー設定と状態管理。

9.31.2. 説明

serverコマンドは、サーバーエンドポイントコネクターとデータソースを記述および管理し、サーバーとホストの両方に関する集約された診断レポートを取得します。

レポートは、設定ファイルとログファイルに加えて、CPU、メモリー、開いているファイル、ネットワークソケットとルーティング、スレッドに関する詳細を提供します。

9.31.3. 概要

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'

9.31.4. サーバーコネクター IP フィルターのオプション

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

1 つ以上の IP フィルタリングルール。

9.31.5. 例

server report
ネットワーク、スレッド、メモリーなどに関する情報を含むサーバーレポートを取得します。

server connector ls
サーバーで使用可能なすべてのコネクターを一覧表示します。

server connector describe endpoint-default
ホスト、ポート、ローカルおよびグローバル接続、IP フィルタリングルールなど、指定されたコネクターに関する情報を表示します。

server connector stop my-hotrod-connector
クラスター全体で確立されたすべての接続をドロップするコネクターを停止します。要求を処理しているコネクターを停止しようとすると、このコマンドは拒否されます。

server connector start my-hotrod-connector
クラスター全体の接続を受け入れることができるように、コネクターを開始します。

server connector ipfilter ls my-hotrod-connector
クラスター全体のコネクターでアクティブなすべての IP フィルタリングルールを一覧表示します。

server connector ipfilter set my-hotrod-connector --rules=ACCEPT/192.168.0.0/16,REJECT/10.0.0.0/8 クラスター全体のコネクターに IP フィルタリングルールを設定します。既存のすべてのルールを置き換えます。拒否ルールの 1 つが、呼び出された接続のアドレスと一致する場合、このコマンドは拒否されます。

server connector ipfilter clear my-hotrod-connector
クラスター全体のコネクターのすべての IP フィルタリングルールを削除します。

server datasource ls
サーバー上で使用可能なすべてのデータソースを一覧表示します。

server datasource test my-datasource
データソースでテスト接続を実行します。

9.32. SHUTDOWN(1)

9.32.1. 名前

shutdown - 実行中のサーバーを停止するか、クラスターを正常に停止します。

9.32.2. 概要

shutdown server ['SERVERS']

shutdown cluster

9.32.3. 例

shutdown server
CLI が接続されているサーバーを停止します。

shutdown server my_server01
ホスト名が **my_server01** のサーバーを停止します。

shutdown cluster
クラスターの状態を保存し、キャッシュストアを使用する場合はエントリーを永続化し、すべてのノードを停止します。

9.32.4. 関連項目

connect(1)、disconnect(1)、quit(1)

9.33. SITE(1)

9.33.1. 名前

site - バックアップの場所を管理し、サイト間のレプリケーション操作を実行します。

9.33.2. 概要

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

9.33.3. オプション

--cache='CACHE_NAME'
キャッシュを指定します。

--site='SITE_NAME'
バックアップの場所を指定します。

9.33.4. 状態転送モードのオプション

--mode='MODE'
状態転送モードを設定します。値は **MANUAL**(デフォルト) または **AUTO** です。

9.33.5. 例

site status --cache=mycache
mycache のすべてのバックアップ場所のステータスを返します。

site status --cache=mycache --site=NYC
mycache の **NYC** のステータスを返します。

site bring-online --cache=mycache --site=NYC
mycache のサイト **NYC** をオンラインにします。

site take-offline --cache=mycache --site=NYC
mycache のサイト **NYC** をオフラインにします。

site push-site-state --cache=mycache --site=NYC
キャッシュをリモートバックアップの場所にバックアップします。

site push-site-status --cache=mycache
mycache をバックアップする操作のステータスを表示します。

site cancel-push-state --cache=mycache --site=NYC
mycache を **NYC** にバックアップする操作をキャンセルします。

site cancel-receive-state --cache=mycache --site=NYC
NYC から状態を受信する操作をキャンセルします。

site clear-push-state-status --cache=myCache
mycache の状態をプッシュする操作のステータスをクリアします。

site state-transfer-mode get --cache=myCache --site=NYC
mycache の状態転送モードを **NYC** に取得します。

site state-transfer-mode set --cache=myCache --site=NYC --mode=AUTO
mycache の **NYC** への自動状態転送を設定します。

site name
ローカルサイトの名前を返します。クロスサイトレプリケーションが設定されていない場合、ローカルサイトの名前は常に"local"です。

site view
すべてのサイトの名前のリストを返します。クロスサイトレプリケーションが設定されていない場合は、空のリスト ("[]") を返します。

9.34. STATS(1)

9.34.1. 名前

stats - リソースに関する統計を表示します。

9.34.2. 概要

stats ['PATH']

9.34.3. 例

stats //containers/default
デフォルトのコンテナーに関する統計を表示します。

stats //containers/default/caches/mycache
mycache キャッシュに関する統計を表示します。

9.34.4. 関連項目

cd(1)、ls(1)、describe(1)

9.35. TASK(1)

9.35.1. 名前

task - サーバー側のタスクとスクリプトを実行してアップロードします

9.35.2. 概要

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

task exec ['TASK_NAME']

9.35.3. 例

task upload --file=hello.js hello
hello.js ファイルからスクリプトをアップロードし、**hello** という名前を付けます。

task exec @@cache@names
使用可能なキャッシュ名を返すタスクを実行します。

task exec hello -Pgreetee=world
hello という名前のスクリプトを実行し、**world** の値で **greetee** パラメーターを指定します。

9.35.4. オプション

-P, --parameters='PARAMETERS'
パラメーター値をタスクとスクリプトに渡します。

-f, --file='FILE'
指定された名前のスクリプトファイルをアップロードします。

9.35.5. 関連項目

ls(1)

9.36. UNALIAS(1)

9.36.1. 名前

unalias - エイリアスを削除します。

9.36.2. 概要

unalias 'ALIAS-NAME'

9.36.3. 例

unalias q
q エイリアスを削除します。

9.36.4. 関連項目

config(1)、alias(1)

9.37. USER(1)

9.37.1. 名前

user - プロパティーセキュリティーレルムで Data Grid ユーザーを管理します。

9.37.2. 概要

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'

9.37.3. 説明

ls、**create**、**describe**、**remove**、**password**、**groups**、および **encrypt-all** サブコマンドを使用して、プロパティーレルムのユーザーを管理します。承認にクラスターロールマッパーを使用する場合は、**roles** サブコマンドを使用してプリンシパルからロールへのマッピングを一覧表示および変更します。

9.37.4. コマンドの概要

user ls
　プロパティーファイルに存在するユーザーまたはグループを一覧表示します。

user create 'username'
　パスワードの入力を求めた後、ユーザーを作成します。

user describe 'username'
　ユーザー名、レルム、およびユーザーが属するグループを含め、ユーザーについて説明します。

user remove 'username'
　指定されたユーザーをプロパティーファイルから削除します。

user password 'username'
　ユーザーのパスワードを変更します。

user groups 'username'
　ユーザーが属するグループを設定します。

user encrypt-all
　プレーンテキストのユーザープロパティーファイル内のすべてのパスワードを暗号化します。

user roles ls 'principal'
　指定されたプリンシパル (ユーザーまたはグループ) のすべてのロールを一覧表示します。

user roles grant --roles='role1'[,'role2'...] 'principal'
　プリンシパルに 1 つ以上のロールを付与します。

user roles deny --roles='role1'[,'role2'...] 'principal'
　プリンシパルに対する 1 つ以上のロールを拒否します。

9.37.5. 共通のオプション

これらのオプションは、すべてのコマンドに適用されます。

-h, --help
　コマンドまたはサブコマンドのヘルプページを表示します。

-s, --server-root='path-to-server-root'
　サーバールートへのパス。デフォルトは **server** です。

-f, --users-file='users.properties'
ユーザーパスワードを含むプロパティーファイルの名前。デフォルトは **users.properties** です。

-w, --groups-file='groups.properties'
ユーザーからグループへのマッピングを含むプロパティーファイルの名前。デフォルトは **groups.properties** です。

9.37.6. ユーザー作成/変更のオプション

-a, --algorithms
パスワードのハッシュに使用されるアルゴリズムを指定します。

-g, --groups='group1,group2,...'
ユーザーが属するグループを指定します。

-p, --password='password'
ユーザーのパスワードを指定します。

-r, --realm='realm'
レルム名を指定します。

--plain-text
パスワードをプレーンテキストで保存するかどうかを定義します (非推奨)。

9.37.7. ユーザーリストのオプション

--groups
ユーザーの代わりにグループのリストを表示します。

9.37.8. ユーザー暗号化 (すべて) のオプション

-a, --algorithms
パスワードのハッシュに使用されるアルゴリズムを指定します。

9.38. VERSION(1)

9.38.1. 名前

version - サーバーのバージョンと CLI のバージョンを表示します。

9.38.2. 概要

version

9.38.3. 例

version
サーバーと CLI のバージョンを返します。

9.38.4. 関連項目

help(1)