Red Hat Data Grid 8.1

Data Grid コマンドラインインターフェース

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: 2023-02-20

Red Hat Data Grid 8.1 Data Grid コマンドラインインターフェース

CLIを使用してデータにアクセスし、Data Gridを管理します

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

コマンドラインインターフェース(CLI)を使用して Data Grid サーバーに接続し、データにアクセスし、管理操作を実行します。

目次

第1章 RED HAT DATA GRID

Data Grid は、高性能の分散型インメモリデータストアです。

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。

1.1. 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 8.1 ドキュメント
- Data Grid 8.1 コンポーネントの詳細
- Data Grid 8.1 でサポートされる構成
- Data Grid 8 機能のサポート
- Data Grid で非推奨の機能

1.2. 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。

1.3. 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。これは大規模な取り組みであるため、これらの変更は今後の複数のリリースで段階的に実施されます。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2章 DATA GRID CLI の使用

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

2.1. ユーザーの作成と変更

Data Grid Server では、ユーザーはデフォルトのプロパティーレルムに対して認証する必要があります。Data Grid Server にアクセスする前に、少なくとも 1 ユーザーとパスワードを作成して認証情報を追加する必要があります。ユーザーが属するセキュリティー承認グループを追加および変更することもできます。

手順

1. **$RHDG_HOME** でターミナルを開きます。
2. **user** コマンドで Data Grid ユーザーを作成し、変更します。

ヒント

このコマンドの使用に関する詳細は、**help user** を実行します。

ユーザーとパスワードの作成

- Linux

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

- Microsoft Windows

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

グループメンバーシップを使用したユーザーの作成

- Linux

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

- Microsoft Windows

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

2.2. 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
     ```

2.3. 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
  } ]
}
```

2.3.1. 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。

2.4. 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、またはクラスターを正常に停止します。

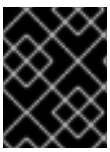
手順

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
```

2.4.1. 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3章 PERFORMING CACHE OPERATIONS WITH THE DATA GRID CLI

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

3.1. DATA GRID コマンドラインインターフェース (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

3.1.1. XML の設定

XML 形式の Data Grid 設定は、スキーマに準拠する必要があり、以下が含まれます。

- & **lt;Infinispan&** gt; ルート要素。
- & **lt;cache-container&gt**; 定義。

XML 設定の例

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

3.1.2. JSON 設定

JSON 形式の Data Grid の設定：

- キャッシュ定義のみが必要です。
- XML 設定の構造に従う必要があります。
  - XML 要素は 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。

JSON 設定の例

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

3.2. キャッシュエントリの追加

データコンテナに**key:value**ペアのエントリを作成します。

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、エントリを確認します。

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

3.3. キャッシュのクリアとエントリの削除

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を削除します。

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

3.4. 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4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。

4.1. 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つのエントリを追加して、CLIから切断します。

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

2. CLIを実行し、ファイルを入力として指定します。

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

3. 新しい Data Grid CLI 接続を作成し、**mybatch** を確認します。

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

注記

CLIバッチファイルは、システムプロパティ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の値に置き換えられます。

4.2. 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 -
```

注記

-c フラグを使用しない場合は、**connect** コマンドを実行する必要があります。

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

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5章 DATA GRID CLIの設定

Data Grid CLIの設定プロパティを定義します。

5.1. DATA GRID CLIプロパティ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、設定プロパティの値を設定します。
3. **config get**コマンドで設定プロパティを確認します。

   ヒント

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

5.2. 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
```

5.3. 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
```

参照

Setting Up SSL Identities for Data Grid Server

5.4. DATA GRID CLIストレージディレクトリ

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

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6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。

6.1. 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
   }
}
```

6.2. 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. カウンタ値を**2**増やします。

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

4. カウンタ値を**-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7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、クエリーを可能にするために Protocol Buffers(Protobuf)を使用してキャッシュ内のデータを構造化します。

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。

7.1. メディアタイプの設定

異なるメディアタイプでキャッシュエントリーをエンコードし、要件に合わせて最も適した形式でデータを保存します。

たとえば、以下の手順では、**application/x-protostream** メディアタイプを設定する方法を説明します。

手順

1. 以下のように、**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。

```
<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. **--file=** オプションを指定して **pcache.xml** から **qcache** 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** へのエントリーを追加し、エンコーディングを確認します。

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

7.2. PROTOBUF スキーマの登録

Protobuf スキーマには、**.proto** 定義ファイルのメッセージとして知られるデータ構造が含まれます。

手順

1. 以下のメッセージで **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
```

7.3. PROTOBUF スキーマを使用したキャッシュのクエリー

Data Grid は自動的に JSON を Protobuf に変換し、JSON 形式でキャッシュエントリーを読み書きし、Protobuf スキーマを使用してクエリーできるようにします。

たとえば、以下の JSON ドキュメントについて考えてみましょう。

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 JSON ドキュメントにはそれぞれ以下が含まれます。

- 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** スキーマのデータタイプに対応する複数のフィールド。

手順

1. **pcache** キャッシュに移動します。

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

2. 以下のように、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。

```
[//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、gender datatype 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8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。

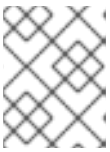
8.1. バックアップ場所のオフラインおよびオンライン化

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。

8.2. 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. CACHE(1)

9.3.1. 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。

9.3.2. 概要

cache ['CACHE_NAME']

9.3.3. 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。

9.3.4. 関連項目

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

9.4. CAS(1)

9.4.1. 名前

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

9.4.2. 概要

cas ['OPTIONS'] ['COUNTER_NAME']

9.4.3. 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。

9.4.4. 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。

9.4.5. 関連項目

add(1)、cas(1)、reset(1)

9.5. CD(1)

9.5.1. 名前

cd - サーバーリソースツリーをナビゲートします。

9.5.2. 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。

9.5.3. 概要

cd ['PATH']

9.5.4. 例

cd caches
リソースツリーの**caches**パスに変更します。

9.5.5. 関連項目

cache(1)、ls(1)、container(1)

9.6. CLEARCACHE(1)

9.6.1. 名前

clearcache - キャッシュからすべてのエントリを削除します。

9.6.2. 概要

clearcache ['CACHE_NAME']

9.6.3. 例

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

9.6.4. 関連項目

cache(1)、drop(1)、remove(1)

9.7. CONFIG(1)

9.7.1. 名前

config - CLI設定プロパティを管理します。

9.7.2. 概要

config

config set 'name' 'value'

config get 'name'

9.7.3. 説明

CLI設定プロパティを管理（リスト、設定、取得）します。

9.7.4. 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を一覧表示します。

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

config get 'name'
　特定のプロパティの値を取得します。

9.7.5. 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。

9.7.6. プロパティ

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。

9.7.7. 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。

9.7.8. 関連項目

alias(1)、unalias(1)

9.8. CONNECT(1)

9.8.1. 名前

connect - 稼働中の ${infinispan.brand.name} サーバーに接続します。

9.8.2. 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。

9.8.3. 概要

connect ['OPTIONS'] ['SERVER_LOCATION']

9.8.4. オプション

-u, --username='USERNAME'

${infinispan.brand.name} 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。

9.8.5. 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。

9.8.6. 関連項目

disconnect(1)

9.9. CONTAINER(1)

9.9.1. 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。

9.9.2. 概要

container ['CONTAINER_NAME']

9.9.3. 例

container default
デフォルトのコンテナを選択します。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。

9.9.4. 関連項目

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

9.10. COUNTER(1)

9.10.1. 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。

9.10.2. 概要

counter ['COUNTER_NAME']

9.10.3. 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。

9.10.4. 関連項目

add(1)、cas(1)

9.11. CREATE(1)

9.11.1. 名前

create: ${infinispan.brand.name} サーバーでキャッシュおよびカウンターを作成します。

9.11.2. 概要

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

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

9.11.3. 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。

9.11.4. 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。

9.11.5. 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。

9.11.6. 関連項目

drop(1)

9.12. DESCRIBE(1)

9.12.1. 名前

describe - リソースに関する情報を表示します。

9.12.2. 概要

describe ['PATH']

9.12.3. 例

describe //containers/default
デフォルトのコンテナ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。

9.12.4. 関連項目

cd(1)、ls(1)

9.13. DISCONNECT(1)

9.13.1. 名前

disconnect: ${infinispan.brand.name} サーバーで、CLI セッションを終了します。

9.13.2. 概要

disconnect

9.13.3. 例

disconnect
現在のCLIセッションを終了します。

9.13.4. 関連項目

connect(1)

9.14. DROP(1)

9.14.1. 名前

drop - キャッシュとカウンターを削除します。

9.14.2. 概要

drop cache CACHE_NAME

drop counter COUNTER_NAME

9.14.3. 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。

9.14.4. 関連項目

create(1)、clearcache(1)

9.15. ENCODING(1)

9.15.1. 名前

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

9.15.2. 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**

9.15.3. 概要

encoding ['ENCODING']

9.15.4. 例

encoding application/json
エントリを**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。

9.15.5. 関連項目

get(1)、put(1)

9.16. GET(1)

9.16.1. 名前

get - キャッシュからエントリを取得します。

9.16.2. 概要

get ['OPTIONS'] **KEY**

9.16.3. オプション

-c, --cache='NAME'
エントリ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。

9.16.4. 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。

9.16.5. 関連項目

query(1)、put(1)

9.17. HELP(1)

9.17.1. 名前

help - コマンドのマニュアルページを出力します。

9.17.2. 概要

help ['COMMAND']

9.17.3. 例

help get
getコマンドのマニュアルページを出力します。

9.17.4. 関連項目

version(1)

9.18. LOGGING(1)

9.18.1. 名前

logging: ${infinispan.brand.name} サーバーランタイムロギング設定を検査し、操作します。

9.18.2. 概要

logging list-loggers

logging list-appenders

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

logging remove **LOGGER_NAME**

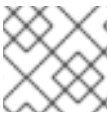
9.18.3. ロギング設定のオプション

-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。

9.18.4. 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。

9.19. LS(1)

9.19.1. 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。

9.19.2. 概要

ls ['PATH']

9.19.3. 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。

9.19.4. 関連項目

cd(1)

9.20. MIGRATE(1)

9.20.1. 名前

migrate: データを ${infinispan.brand.name} から別のバージョンに移行します。

9.20.2. 概要

migrate cluster synchronize

migrate cluster disconnect

9.20.3. 説明

migrate コマンドを使用して、データを ${infinispan.brand.name} から別のバージョンに移行します。

9.20.4. 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。

9.20.5. 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。

9.20.6. 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上のコア数です。

9.20.7. クラスターの接続解除オプション

-c, --cache='name'
ソースから切断するキャッシュの名前。

9.21. PATCH(1)

9.21.1. 名前

patch - サーバーパッチを管理します。

9.21.2. 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。

9.21.3. 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'...]

9.21.4. 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。

9.21.5. 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外のターゲットサーバーへのパスを設定します。

9.21.6. 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。

9.21.7. 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外のターゲットサーバーへのパスを設定します。

9.21.8. 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'）。

9.21.9. 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。

9.22. PUT(1)

9.22.1. 名前

put - キャッシュエントリを追加または更新します。

9.22.2. 説明

新しいキーのエントリを作成します。既存のキーの値を置き換えます。

9.22.3. 概要

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

9.22.4. 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'

エントリの値を含むファイルを指定します。

-l, --ttl='TTL'

エントリ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、エントリが削除されることはありません。

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

エントリ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、エントリが削除されることはありません。

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

エントリが存在しない場合にのみエントリを配置します。

9.22.5. 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。

9.22.6. 関連項目

get(1)、remove(1)

9.23. QUERY(1)

9.23.1. 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。

9.23.2. 概要

query ['OPTIONS'] **QUERY_STRING**

9.23.3. 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** および BROADCAST です。デフォルトは **FETCH** です。

9.23.4. 例

クエリー "from org.infinispan.rest.search.entity.Person p where p.gender = 'MALE'"
Queries the entries from a Protobuf **Person** entity that the gender datatype is **MALE**.

9.23.5. 関連項目

schema(1)

9.24. QUIT(1)

9.24.1. 名前

quit - コマンドラインインターフェイスを終了します。

9.24.2. 概要

quit

9.24.3. 例

を終了すると、CLI を終了します。

9.24.4. 関連項目

disconnect(1)、shutdown(1)

9.25. REMOVE(1)

9.25.1. 名前

remove - キャッシュからエントリを削除します。

9.25.2. 概要

remove KEY ['OPTIONS']

9.25.3. オプション

--cache='NAME'

エントリ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。

9.25.4. 例

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

9.25.5. 関連項目

cache(1)、drop(1)、clearcache(1)

9.26. RESET(1)

9.26.1. 名前

reset - カウンターの初期値を復元します。

9.26.2. 概要

reset ['COUNTER_NAME']

9.26.3. 例

reset cnt_a
cnt_aカウンターをリセットします。

9.26.4. 関連項目

add(1)、cas(1)、drop(1)

9.27. SCHEMA(1)

9.27.1. 名前

schema - protobufスキーマをアップロードして登録します。

9.27.2. 概要

schema ['OPTIONS'] SCHEMA_NAME

9.27.3. 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。

9.27.4. 例

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

9.27.5. 関連項目

query(1)

9.28. SHUTDOWN(1)

9.28.1. 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。

9.28.2. 概要

shutdown server ['SERVERS']

クラスターのシャットダウン

9.28.3. 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を永続化し、すべてのノードを停止します。

9.28.4. 関連項目

connect(1)、disconnect(1)、quit(1)

9.29. SITE(1)

9.29.1. 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。

9.29.2. 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']

9.29.3. 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。

9.29.4. 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。

9.30. STATS(1)

9.30.1. 名前

stats - リソースに関する統計を表示します。

9.30.2. 概要

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

9.30.3. 例

```
stats //containers/default
```
デフォルトのコンテナ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。

9.30.4. 関連項目

cd(1)、ls(1)、describe(1)

9.31. TASK(1)

9.31.1. 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

9.31.2. 概要

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

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

9.31.3. 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。

9.31.4. オプション

-P, --parameters='PARAMETERS'

パラメータ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。

9.31.5. 関連項目

ls(1)

9.32. UNALIAS(1)

9.32.1. 名前

unalias - エイリアスを削除します。

9.32.2. 概要

unalias 'ALIAS-NAME'

9.32.3. 例

unalias q
qエイリアスを削除します。

9.32.4. 関連項目

config(1)、alias(1)

9.33. USER(1)

9.33.1. 名前

user - プロパティーセキュリティーレルムで ${infinispan.brand.name} ユーザーを管理します。

9.33.2. 概要

user ls

user create 'username'

user describe 'username'

user remove 'username'

user password 'username'

user groups 'username'

user encrypt-all

9.33.3. 説明

プロパティーセキュリティーレルムに保存されているユーザーを管理（リスト、作成、説明、削除、変更）します。注記： このコマンドは、プロパティーレルムでのみ使用できます。

9.33.4. コマンドの概要

user ls

プロパティ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ファイル内のすべてのパスワードを暗号化します。

9.33.5. 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ファイルの名前。デフォルトはgroups.propertiesです。

9.33.6. 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）。

9.33.7. 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。

9.33.8. 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。

9.34. VERSION(1)

9.34.1. 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。

9.34.2. 概要

version

9.34.3. 例

version
サーバーとCLIのバージョンを返します。

9.34.4. 関連項目

help(1)