Red Hat Data Grid 8.4

Data Grid クロスサイトレプリケーション

Data Grid クラスター間のデータのバックアップ

Last Updated: 2024-04-30

Red Hat Data Grid 8.4 Data Grid クロスサイトレプリケーション

Data Grid クラスター間のデータのバックアップ

法律上の通知

概要

Data Grid は、グローバルクラスターを形成して、地理的な場所全体にデータを複製できます。本ガイドでは、キャッシュのバックアップの場所を設定し、クロスサイト操作を実行する方法を説明します。

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。

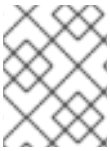
DATA GRID のドキュメント

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 8.4 ドキュメント
- Data Grid 8.4 コンポーネントの詳細
- Data Grid 8.4 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 クロスサイトレプリケーション

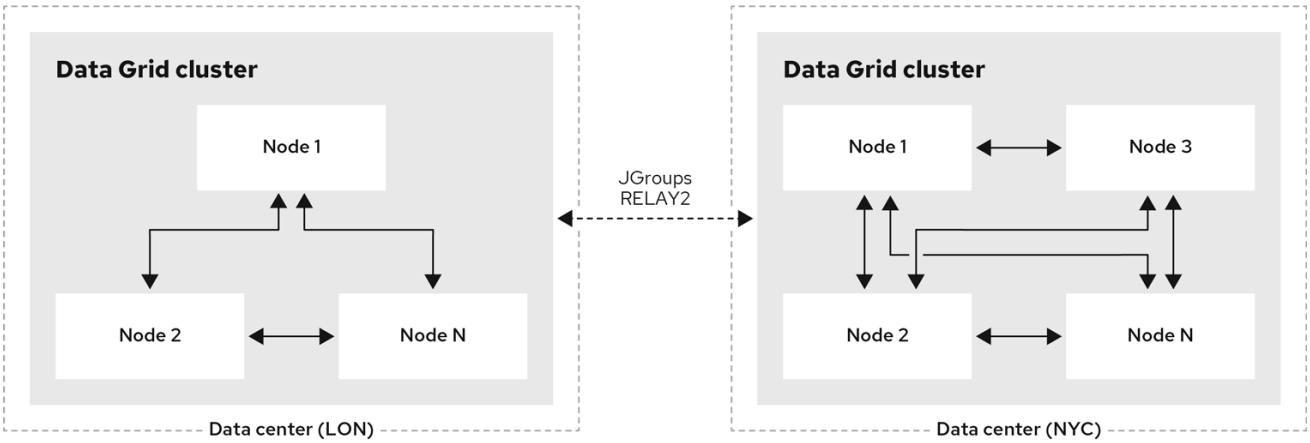
このセクションでは、Data Grid のクロスサイトレプリケーション機能について説明します。これには、リレーノード、状態遷移、およびリモートキャッシュのクライアント接続に関する詳細が含まれます。

1.1. クロスサイトレプリケーション

Data Grid は、地理的に分散したデータセンターで実行されているクラスター間および異なるクラウドプロバイダー間でデータをバックアップできます。クロスサイトレプリケーションは、Data Grid にグローバルクラスタービューを提供し、以下を提供します。

- 停電や災害時のサービス継続を保証します。
- グローバルに分散されたキャッシュ内のデータへの単一アクセスポイントをクライアントアプリケーションに提供します。

図1.1 クロスサイトレプリケーション

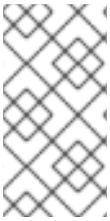


1.2. リレーノード

リレーノードは、Data Grid クラスターのノードであり、バックアップの場所から要求を送受信します。

ノードがリレーノードではない場合、バックアップリクエストをローカルのリレーノードに転送する必要があります。リレーノードのみが、バックアップロケーションにリクエストを送信できます。

最適なパフォーマンスを得るには、すべてのノードをリレーノードとして設定する必要があります。これにより、クラスター内の各ノードがバックアップ要求をローカルリレーノードに転送しなくても、リモートサイトに直接バックアップできるため、バックアップ要求の速度が向上します。



注記

このドキュメントの図では、JGroups RELAY2 プロトコルのデフォルトが 1 つのリレーノードを持つ Data Grid クラスターであることから、これを示しています。同様に、クラスター内の各リレーノードは、リモートクラスター内の各リレーノードと通信するため、1 つのリレーノードの方が説明がしやすくなります。

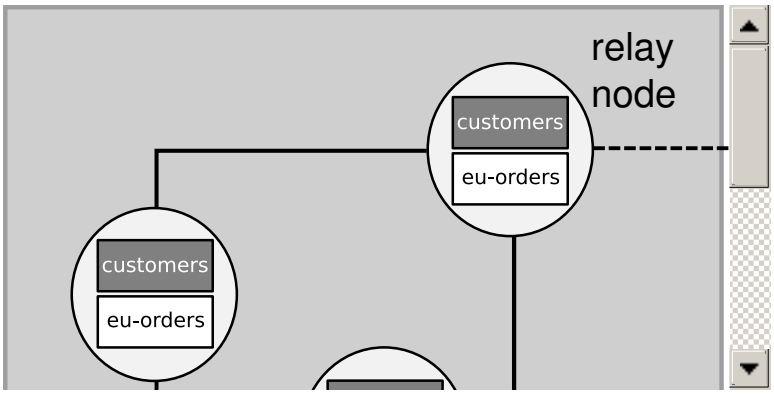
注記

JGroups 設定では、リレーノードを"site master"ノードと呼びます。Data Grid は、よりわかりやすく、ユーザーにとってより直感的な選択肢を提供するため、代わりにリレーノードを使用します。

1.3. DATA GRID キャッシュのバックアップ

Data Grid キャッシュには、リモートサイトをバックアップの場所として命名できるようにする **backups** 設定が含まれます。

たとえば、以下の図では、3 つのキャッシュ "customers"、"eu-orders"、および "us-orders" を示しています。



- **LON** では、"customers" はバックアップの場所として **NYC** を指定します。
- **NYC** では、"customers" はバックアップの場所として **LON** を指定します。
- "eu-orders" および "us-orders" にはバックアップはなく、それぞれのクラスターのローカルになります。

1.4. バックアップストラテジー

Data Grid は、キャッシュへの書き込みが発生すると同時に、クラスター間でデータを複製します。たとえば、クライアントが **LON** に "k1" と書き込む場合、Data Grid は同時に "k1" を **NYC** バックアップします。

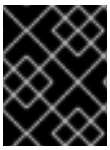
別のクラスターにデータのバックアップを作成するには、Data Grid は同期または非同期ストラテジーを使用します。

同期ストラテジー
Data Grid がデータをバックアップの場所に複製するとき、ローカルクラスターのキャッシュとリモートクラスターのキャッシュに同時に書き込みます。同期ストラテジーでは、Data Grid は両方の書き込み操作が完了するのを待ってから戻ります。

バックアップ操作が失敗した場合に、Data Grid がローカルクラスターのキャッシュへの書き込みを処理する方法を制御できます。Data Grid は以下を行うことができます。

- 失敗したバックアップを無視し、ローカルクラスターへの書き込みを警告せずに続行します。
- 警告メッセージをログに記録するか、例外を出力し、ローカルクラスターへの書き込みを続行します。
- 失敗したバックアップ操作をカスタムロジックで処理します。

同期バックアップは、楽観的なトランザクションに参加するキャッシュを持つ 2 フェーズコミットもサポートします。バックアップの最初のフェーズはロックを取得します。2 番目のフェーズは変更をコミットします。



重要

クロスサイトレプリケーションのある 2 フェーズコミットは、ネットワーク全体で 2 つのラウンドトリップが必要なため、パフォーマンスに大きく影響します。

非同期ストラテジー
Data Grid がデータをバックアップ場所に複製する場合、操作が完了するのを待たずにローカルキャッシュに書き込みます。

非同期バックアップ操作およびローカルキャッシュへの書き込みは、互いに独立しています。バックアップ操作が失敗した場合は、ローカルキャッシュへの書き込み操作は続行され、例外は発生しません。これが発生すると、Data Grid は、リモートクラスターがクロスサイトビューから切断されるまで、書き込み操作も再試行します。

同期バックアップと非同期バックアップの比較
同期バックアップは、サイト全体でのデータの一貫性を最も強力に保証します。**strategy=sync** の場合、**cache.put()** 呼び出しが返されると、ローカルキャッシュとバックアップの場所で値が最新の状態であることがわかります。

この一貫性のために犠牲となるのがパフォーマンスです。同期バックアップは、非同期バックアップと比較すると、レイテンシーが大幅に高くなります。

一方、非同期バックアップは、クライアントリクエストにレイテンシーを追加しないので、パフォーマンスに影響を及ぼすことはありません。ただし、**strategy=async** の場合、**cache.put()** 呼び出しが返されると、バックアップの場所の値がローカルキャッシュの値と同じであるかどうかを確認できません。

1.5. バックアップの場所の自動オフラインパラメーター

クラスター全体でデータをレプリケートする操作は、RAM および CPU を過剰に使用して、リソースを大量に消費します。特定の期間後にリクエストの受け入れを停止する際に、Data Grid がバックアップの場所をオフラインにすることで、リソースの浪費を避けることができます。

Data Grid は、失敗した順次要求の数と最初の失敗からの経過時間に基づいて、リモートサイトをオフラインにします。ターゲットクラスターにクロスサイトビュー (JGroups ブリッジ) のノードがない場合や、ターゲットクラスターが要求を確認する前にタイムアウトが期限切れになる場合は、要求は失敗します。

バックアップのタイムアウト
バックアップ設定には、クラスター間でデータを複製する操作のタイムアウト値が含まれます。タイムアウトが終了する前に操作が完了しない場合、Data Grid は失敗としてそれらを記録します。

以下の例では、データを NYC に複製する操作は、10 秒以内に完了しない場合、失敗として記録されます。

XML

```
<distributed-cache>
  <backups>
    <backup site="NYC"
            strategy="ASYNC"
```

```
            timeout="10000" />
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "backups": {
      "NYC" : {
        "backup" : {
          "strategy" : "ASYNC",
          "timeout" : "10000"
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
  backups:
    NYC:
      backup:
        strategy: "ASYNC"
        timeout: "10000"
```

失敗の数
バックアップの場所がオフラインになる前に発生する可能性のある **連続する** 失敗の数を指定できます。

以下の例では、クラスターが NYC にデータを複製しようとし、5 回連続して操作が失敗すると、NYC は自動的にオフラインになります。

XML

```
<distributed-cache>
  <backups>
    <backup site="NYC"
            strategy="ASYNC"
            timeout="10000">
      <take-offline after-failures="5"/>
    </backup>
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "backups": {
```

```
      "NYC" : {
        "backup" : {
          "strategy" : "ASYNC",
          "timeout" : "10000",
          "take-offline" : {
            "after-failures" : "5"
          }
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
  backups:
    NYC:
      backup:
        strategy: "ASYNC"
        timeout: "10000"
        takeOffline:
          afterFailures: "5"
```

待機時間
バックアップ操作の失敗時に、サイトをオフラインにするまで待機する時間を指定することもできます。待機時間がなくなる前にバックアップ要求が成功した場合、Data Grid はサイトをオフラインにしません。

バックアップの場所を自動的にオフラインにするまでの待機時間は、通常、1、2 分が適切とされています。待機時間が短すぎると、バックアップの場所はすぐにオフラインになります。次に、クラスターをオンラインに戻し、状態遷移操作を実行して、クラスター間でデータが同期されていることを確認します。

失敗数の負の値またはゼロの値は **1** の値と同等です。Data Grid は、障害が発生した後、バックアップの場所がオフラインになるのを待つために最小限の時間のみを使用します。以下に例を示します。

```
<take-offline after-failures="-1"
              min-wait="10000"/>
```

以下の例では、クラスターが NYC にデータを複製しようとし、6 つ以上の連続した失敗があり、最初の失敗した操作から 15 秒が経過すると、NYC は自動的にオフラインになります。

XML

```
<distributed-cache>
  <backups>
    <backup site="NYC"
            strategy="ASYNC"
            timeout="10000">
      <take-offline after-failures="5" min-wait="15000"/>
    </backup>
  </backups>
</distributed-cache>
```

-

JSON

```
{
  "distributed-cache": {
    "backups": {
      "NYC" : {
        "backup" : {
          "strategy" : "ASYNC",
          "timeout" : "10000",
          "take-offline" : {
            "after-failures" : "5",
            "min-wait" : "15000"
          }
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
  backups:
    NYC:
      backup:
        strategy: "ASYNC"
        timeout: "10000"
        takeOffline:
          afterFailures: "5"
          minWait: "15000"
```

1.6. 状態遷移

状態遷移は、サイト間でデータを同期する管理操作です。

たとえば、**LON** がオフラインになると、**NYC** がクライアント要求の処理を開始します。**LON** をオンラインに戻すと、**LON** の Data Grid クラスターには **NYC** のクラスターと同じデータはありません。

LON と **NYC** の間でデータの一貫性を保つには、**NYC** から **LON** に状態をプッシュできます。

- 状態遷移は双方向です。たとえば、**NYC** から **LON** へ、または **LON** から **NYC** へ状態のプッシュを実行できます。
- 状態をオフラインサイトにプッシュすると、オンラインに戻ります。
- 状態遷移は、発信元サイトと受信サイトの両方のサイトに存在するデータのみを上書きします。Data Grid はデータを削除しません。
  たとえば、"k2" は **LON** および **NYC** に存在します。"k2" は、**LON** がオフライン時に **NYC** から削除されます。**LON** をオンラインに戻すと、"k2" は引き続きその場所に存在します。**NYC** から **LON** に状態をプッシュすると、転送は **LON** の "k2" には影響しません。

ヒント

状態遷移後にキャッシュの内容が同じになるようにするには、状態をプッシュする前に受信サイトのキャッシュからすべてのデータを削除します。

CLI から **clear()** メソッドまたは **clearcache** コマンドを使用します。

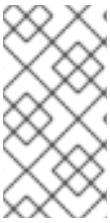
- 状態遷移は、プッシュの開始後に発生するデータへの更新を上書きしません。
  たとえば、"k1,v1" は **LON** および **NYC** に存在します。**LON** がオフラインになったため、**NYC** から**LON** に状態遷移をプッシュします。これにより、**LON** が再びオンラインになります。状態遷移が完了する前に、クライアントは **LON** に "k1,v2" を配置します。

  この場合、プッシュを開始した後に変更が発生したため、**NYC** からの状態遷移は "k1,v2" を上書きしません。

自動状態遷移
デフォルトでは、CLI を使用して、または JMX や REST 経由で、クロスサイトの状態遷移操作を手動で実行する必要があります。

ただし、非同期バックアップストラテジーを使用すると、Data Grid はクロスサイト状態遷移操作を自動的に実行できます。

バックアップロケーションがオンラインに戻り、ネットワーク接続が安定すると、Data Grid はバックアップロケーション間の双方向の状態遷移を開始します。たとえば、Data Grid は、状態を **LON** から **NYC** に、**NYC** から **LON**に同時に遷移します。



注記

一時的なネットワークの切断による状態遷移操作のトリガーを回避するために、バックアップの場所がオフラインになるために満たす必要のある条件が 2 つあります。バックアップの場所のステータスはオフラインでなければならず、JGroups RELAY2 のクロスサイトビューに含めることはできません。

自動状態遷移は、キャッシュの開始時にもトリガーされます。

LON が起動するシナリオでは、キャッシュの開始後に、通知が **NYC** に送信されます。これに続いて、**NYC** は **LON** への一方向の状態遷移を開始します。

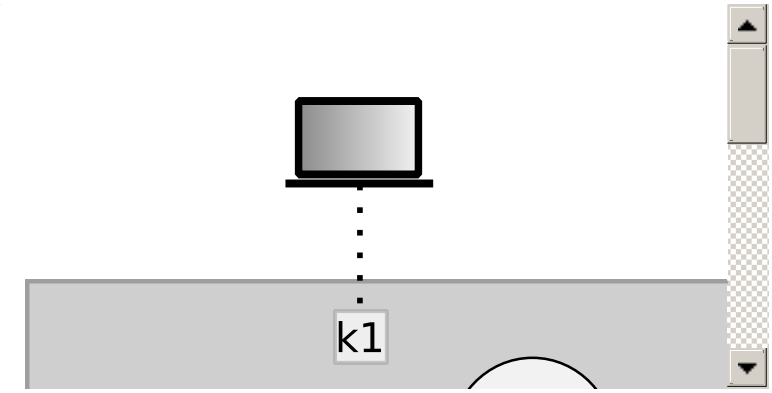
関連情報

- org.infinispan.Cache.clear()
- Data Grid コマンドラインインターフェイスの使用
- Data Grid REST API

1.7. サイト全体でのクライアント接続

クライアントは、Active/Passive または Active/Active 設定のいずれかで Data Grid クラスターに書き込みできます。

Active/Passive
以下の図は、Data Grid が 1 つのサイトのみからのクライアント要求を処理する Active/Passive を示しています。



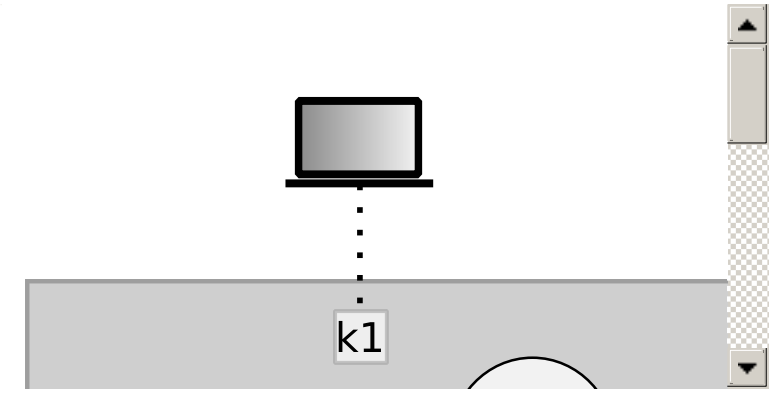
上記のイメージでは、以下のようになります。

1. クライアントは **LON** で Data Grid クラスターに接続します。
2. クライアントは "k1" をキャッシュに書き込みます。
3. **LON** のリレーノード "n1" は、"k1" を複製する要求を **NYC** のリレーノード "nA" に送信します。

Active/Passive では、**NYC** はデータの冗長性を提供します。**LON** の Data Grid クラスターが何らかの理由でオフラインになると、クライアントは **NYC** への要求の送信を開始できます。**LON** をオンラインに戻すと、**NYC** とデータを同期し、クライアントを **LON** に戻すことができます。

Active/Active

以下の図は、Data Grid が 2 つのサイトでクライアント要求を処理する Active/Active を示しています。



上記のイメージでは、以下のようになります。

1. クライアント A は、**LON** で Data Grid クラスターに接続します。
2. クライアント A は "k1" をキャッシュに書き込みます。
3. クライアント B は、**NYC** の Data Grid クラスターに接続します。
4. クライアント B は "k2" をキャッシュに書き込みます。
5. **LON** および **NYC** のリレーノード は、"k1" が **NYC** に複製され、"k2" が **LON** に複製されるように要求を送信します。

Active/Active では、**NYC** と **LON** の両方は、クライアント要求の処理中にデータをリモートキャッシュに複製します。**NYC** または **LON** のいずれかがオフラインになると、クライアントはオンラインサイトへの要求の送信を開始できます。その後、オフラインサイトをオンラインに戻し、状態をプッシュしてデータを同期し、必要に応じてクライアントを切り替えることができます。

バックアップストラテジーおよびクライアント接続

重要

Active/Active 設定では、非同期バックアップストラテジー (**strategy=async**) が推奨されます。

複数のクライアントが同じエントリーに同時に書き込もうとし、バックアップストラテジーが同期 (**strategy=sync**) の場合、デッドロックが発生します。ただし、両方のサイトが異なるデータセットにアクセスする場合、Active/Passive 設定で同期バックアップストラテジーを使用できます。この場合、同時書き込みによるデッドロックのリスクはありません。

1.7.1. 同時書き込みおよび競合エントリー

クライアントが同時に同じエントリーに書き込むが、サイトが異なる場合は、Active/Active サイト設定でエントリーの競合が発生する可能性があります。

たとえば、クライアント B が **NYC** の k1 に書き込むのと同時に、クライアント A は **LON** の "k1" に書き込みます。この場合、"k1" の値は **LON** と **NYC** とで異なります。レプリケーションの発生後、"k1" のどの値がどのサイトに存在するかは保証されません。

データの整合性を確保するために、Data Grid はベクトルクロックアルゴリズムを使用して、次の図のように、バックアップ操作中に競合するエントリーを検出します。

```
          LON       NYC

k1=(n/a)   0,0       0,0

k1=2       1,0  -->  1,0   k1=2

k1=3       1,1  <--  1,1   k1=3

k1=5       2,1       1,2   k1=8

              -->  2,1 (conflict)
(conflict)  1,2  <--
```

ベクトルクロックは、エントリーへの書き込みごとにインクリメントするタイムスタンプのメタデータです。上記の例では、**0,0** は、"k1" 上のベクトルクロックの初期値を表します。

クライアントは **LON** に "k1=2" を配置し、ベクトルクロックは **1,0** で、Data Grid はこれを **NYC** に複製します。その後、クライアントは **NYC** に "k1=3" を配置し、ベクタークロックは **1,1** に更新されます。Data Grid はこれを **LON** に複製します。

ただし、クライアントが **NYC** に "k1=8" を配置するのと同時に、クライアントが **LON** に"k1=5" を配置すると、Data Grid は競合するエントリーを検出します。これは、"k1" のベクター値が **LON** と **NYC** の間で厳密に大きかったり小さかったりしないためです。

競合するエントリーを見つけると、Data Grid は Java **compareTo(String anotherString)** メソッドを使用してサイト名を比較します。どのキーが優先されるかを決定するために、Data Grid は辞書式順序で他のキーよりも小さいサイト名を選択します。**AAA** という名前のサイトからのキーは、**AAB** という名前のサイトからのキーよりも優先されます。

同じ例に従って、"k1" の競合を解決するために、Data Grid は **LON** からの "k1" の値を使用します。これにより、Data Grid が競合を解決し、値を複製した後に、**LON** と **NYC** の両方で "k1=5" となります。

ヒント

競合するエントリーを解決するための優先順位を表す簡単な方法として、サイト名の前に番号を付けます。たとえば、**1LON** や **2NYC** などです。

バックアップストラテジー

Data Grid は、非同期バックアップストラテジ (**strategy=async**) のみを使用して競合解決を実行します。

Active/Active 設定で同期バックアップストラテジーを使用することはできません。この設定では、同時書き込みがデッドロックになり、データが失われます。ただし、両方のサイトが異なるデータセットにアクセスする場合、Active/Active 設定で同期バックアップストラテジーを使用できます。この場合、同時書き込みによるデッドロックのリスクはありません。

クロスサイトマージポリシー

Data Grid は、以下を行うために Data Grid を設定するクロスサイトマージポリシーに加えて、**XSiteEntryMergePolicy** SPI を提供します。

- 競合するエントリーを常に削除します。
- 書き込み/削除の競合が発生したときに、書き込み操作を適用します。
- 書き込み/削除の競合が発生したときにエントリーを削除します。

関連情報

- **XSiteMergePolicy** enum lists all merge polices that Data Grid provides
- **XSiteEntryMergePolicy** SPI
- java.lang.String#compareTo()

1.8. クロスサイトレプリケーションを使用した有効期限

有効期限は、時間に基づいてキャッシュエントリーを削除します。Data Grid では、エントリーの有効期限を設定する方法は 2 つあります。

有効期間

lifespan 属性は、エントリーが存在することのできる最大時間を設定します。クロスサイトレプリケーションで **lifespan** を設定する場合、Data Grid クラスターはリモートサイトとは別にエントリーを期限切れにします。

最大アイドル

max-idle は、特定の期間における読み取り操作または書き込み操作に基づいて、エントリーが存在する期間を指定します。クロスサイトレプリケーションで **max-idle** を設定すると、Data Grid クラスターは touch コマンドを送信して、リモートサイトでアイドルタイムアウト値を調整します。

注記

クロスサイトデプロイメントで最大アイドル有効期限を使用すると、**max-idle** 値の同期を維持するための追加の処理により、一部の操作が完了するまで時間がかかることがあるため、パフォーマンスに影響を及ぼす可能性があります。

第2章 DATA GRID クロスサイトレプリケーションの設定

Data Grid クラスターが相互に検出し、リレーノードがクロスサイトレプリケーションのメッセージを送信できるように、クラスタートランスポートを設定します。その後、バックアップの場所を Data Grid キャッシュに追加できます。

2.1. クロスサイトレプリケーションのためのクラスタートランスポートの設定

JGroups RELAY2 をトランスポート層に追加して、Data Grid がキャッシュをバックアップの場所に複製できるようにします。

手順

1. Data Grid 設定を開いて編集します。
2. RELAY2 プロトコルを JGroups スタックに追加します。
3. トランスポート設定の **stack** 属性でスタック名を指定して、データグリッドクラスターがそれを使用するようにします。
4. Data Grid 設定を保存して閉じます。

JGroups RELAY2 スタック
以下の設定は、以下の JGroups RELAY2 スタックを示しています。

- クラスター間トランスポートにデフォルトの JGroups UDP スタックを使用します。これは、ローカルサイトのノード間の通信を参照します。
- クロスサイトのレプリケーショントラフィックにデフォルトの JGroups TCP スタックを使用します。
- ローカルサイトに **LON** という名前を付けます。
- クロスサイトのレプリケーション要求を送信できるクラスター内の最大 1000 ノードを指定します。
- クロスサイトのレプリケーションに参加するすべてのバックアップの場所の名前を指定します。

```
<infinispan>
  <jgroups>
    <stack name="xsite" extends="udp">
      <relay.RELAY2 xmlns="urn:org:jgroups"
                    site="LON"
                    max_site_masters="1000"/>
      <remote-sites default-stack="tcp">
        <remote-site name="LON"/>
        <remote-site name="NYC"/>
      </remote-sites>
    </stack>
  </jgroups>
  <cache-container>
```

```
    <transport cluster="${cluster.name}" stack="xsite"/>
  </cache-container>
</infinispan>
```

関連情報

- JGroups RELAY2 スタック
- Data Grid 設定スキーマ参照

2.1.1. カスタム JGroups RELAY2 スタック

カスタムの JGroups RELAY2 スタックを Data Grid クラスターに追加して、クロスサイトのレプリケーションに異なるトランスポートプロパティーを使用することができます。たとえば、以下の設定では、検出に MPING の代わりに TCPPING を使用し、デフォルトの TCP スタックを拡張します。

```
<infinispan>
  <jgroups>
    <stack name="relay-global" extends="tcp">
      <TCPPING initial_hosts="192.0.2.0[7800]"
               stack.combine="REPLACE"
               stack.position="MPING"/>
    </stack>
    <stack name="xsite" extends="udp">
      <relay.RELAY2 site="LON" xmlns="urn:org:jgroups"
                    max_site_masters="10"
                    can_become_site_master="true"/>
      <remote-sites default-stack="relay-global">
        <remote-site name="LON"/>
        <remote-site name="NYC"/>
      </remote-sites>
    </stack>
  </jgroups>
</infinispan>
```

関連情報

- JGroups RELAY2
- 複数のサイト間のリレー (RELAY2)

2.2. キャッシュへのバックアップの場所の追加

Data Grid がこれらのクラスターのキャッシュにデータを複製できるように、リモートサイトの名前を指定します。

手順

1. Data Grid 設定を開いて編集します。
2. **backups** 要素をキャッシュ設定に追加します。
3. リモートサイトの名前をバックアップの場所として指定します。
   たとえば、**LON** 設定で、**NYC** をバックアップとして指定します。

4. 各サイトが他のサイトのバックアップになるように、各クラスターで上記の手順を繰り返します。
   たとえば、**NYC** のバックアップとして **LON** を追加する場合、**LON** のバックアップとして **NYC** も追加する必要があります。

5. Data Grid 設定を保存して閉じます。

バックアップの設定
以下の例は、**LON** クラスターの "customers" キャッシュ設定を示しています。

XML

```
<replicated-cache name="customers">
  <backups>
    <backup site="NYC"
            strategy="ASYNC" />
  </backups>
</replicated-cache>
```

JSON

```
{
  "replicated-cache": {
    "name": "customers",
    "backups": {
      "NYC": {
        "backup" : {
          "strategy" : "ASYNC"
        }
      }
    }
  }
}
```

YAML

```
replicatedCache:
  name: "customers"
  backups:
    NYC:
      backup:
        strategy: "ASYNC"
```

以下の例は、**NYC** クラスターの "customers" キャッシュ設定を示しています。

XML

```
<distributed-cache name="customers">
  <backups>
    <backup site="LON"
```

```
            strategy="ASYNC" />
    </backups>
  </distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "name": "customers",
    "backups": {
      "LON": {
        "backup": {
          "strategy": "ASYNC"
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
  name: "customers"
  backups:
    LON:
      backup:
        strategy: "ASYNC"
```

関連情報

- Data Grid 設定スキーマ参照

2.3. 異なる名前でのキャッシュへのバックアップ

デフォルトでは、Data Grid は同じ名前のキャッシュ間でデータを複製します。Data Grid を別の名前でキャッシュ間で複製する場合は、キャッシュごとにバックアップを明示的に宣言できます。

手順

1. Data Grid 設定を開いて編集します。
2. **backup-for** または **backupFor** を使用して、リモートサイトからのデータをローカルサイトの別の名前でキャッシュに複製します。
3. Data Grid 設定を保存して閉じます。

設定のバックアップ
以下の例では、**LON** クラスターの "customers" キャッシュから更新を受信するように、"eu-customers" キャッシュを設定します。

XML

```
<distributed-cache name="eu-customers">
  <backups>
    <backup site="LON"
            strategy="ASYNC" />
  </backups>
  <backup-for remote-cache="customers"
              remote-site="LON" />
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "name": "eu-customers",
    "backups": {
      "LON": {
        "backup": {
          "strategy": "ASYNC"
        }
      }
    },
    "backup-for" : {
      "remote-cache" : "customers",
      "remote-site" : "LON"
    }
  }
}
```

YAML

```
distributedCache:
  name: "eu-customers"
  backups:
    LON:
      backup:
        strategy: "ASYNC"
  backupFor:
    remoteCache: "customers"
    remoteSite: "LON"
```

2.4. クロスサイト状態遷移の設定

クロスサイト状態遷移設定を変更し、パフォーマンスを最適化して、操作を手動で行うか自動で行うかを指定します。

手順

1. Data Grid 設定を開いて編集します。
2. 必要に応じて状態遷移操作を設定します。
    a. **chunk-size** または **chunkSize** を使用して、各状態遷移操作に含めるエントリー数を指定します。

b. **timeout** で状態遷移操作が完了するまで待機する時間をミリ秒単位で指定します。

c. Data Grid が **max-retries** または **maxRetries** で失敗した状態遷移を再試行する最大試行回数を設定します。

d. **wait-time** または **waitTime** を使用して、再試行の間に待機する時間をミリ秒単位で指定します。

e. 状態遷移操作が自動または手動で実行されるかどうかを **mode** で指定します。

3. Data Grid 設定を開いて編集します。

状態遷移の設定

XML

```
<distributed-cache name="eu-customers">
  <backups>
    <backup site="LON"
            strategy="ASYNC">
      <state-transfer chunk-size="600"
                      timeout="2400000"
                      max-retries="30"
                      wait-time="2000"
                      mode="AUTO"/>
    </backup>
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "name": "eu-customers",
    "backups": {
      "LON": {
        "backup": {
          "strategy": "ASYNC",
          "state-transfer": {
            "chunk-size": "600",
            "timeout": "2400000",
            "max-retries": "30",
            "wait-time": "2000",
            "mode": "AUTO"
          }
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
```

```
  name: "eu-customers"
  backups:
    LON:
      backup:
        strategy: "ASYNC"
        stateTransfer:
          chunkSize: "600"
          timeout: "2400000"
          maxRetries: "30"
          waitTime: "2000"
          mode: "AUTO"
```

2.5. 競合解決アルゴリズムの設定

別のアルゴリズムを使用してバックアップの場所間の競合するエントリーを解決するように Data Grid を設定します。

手順

1. Data Grid 設定を開いて編集します。
2. 競合するエントリーを解決するマージポリシーとして、Data Grid アルゴリズムまたはカスタム実装のいずれかを指定します。
3. 編集用に Data Grid 設定を保存して閉じます。

Data Grid アルゴリズム

ヒント

org.infinispan.xsite.spi.XSiteMergePolicy 列挙で、すべての Data Grid アルゴリズムと説明を検索します。

以下の設定例では、両方のサイトから競合するエントリーを削除する **ALWAYS_REMOVE** アルゴリズムを使用します。

XML

```
<distributed-cache>
  <backups merge-policy="ALWAYS_REMOVE">
    <backup site="LON" strategy="ASYNC"/>
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "backups": {
      "merge-policy": "ALWAYS_REMOVE",
      "LON": {
        "backup": {
          "strategy": "ASYNC"
```

```
      }
    }
  }
 }
}
```

YAML

```
distributedCache:
  backups:
    mergePolicy: "ALWAYS_REMOVE"
    LON:
      backup:
        strategy: "ASYNC"
```

カスタム競合解決アルゴリズム
カスタムの **XSiteEntryMergePolicy** 実装を作成する場合は、完全修飾クラス名をマージポリシーとして指定することができます。

XML

```
<distributed-cache>
  <backups merge-policy="org.mycompany.MyCustomXSiteEntryMergePolicy">
    <backup site="LON" strategy="ASYNC"/>
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "backups": {
      "merge-policy": "org.mycompany.MyCustomXSiteEntryMergePolicy",
      "LON": {
        "backup": {
          "strategy": "ASYNC"
        }
      }
    }
  }
}
```

YAML

```
distributedCache:
  backups:
    mergePolicy: "org.mycompany.MyCustomXSiteEntryMergePolicy"
    LON:
      backup:
        strategy: "ASYNC"
```

関連情報

関連情報

- org.infinispan.xsite.spi.XSiteEntryMergePolicy
- org.infinispan.xsite.spi.XSiteMergePolicy
- org.infinispan.xsite.spi.SiteEntry
- Data Grid 設定スキーマ参照

2.6. 非同期バックアップの TOMBSTONE のクリーニング

非同期バックアップ戦略では、Data Grid は、キーを削除するときに、廃棄 (Tombstone) と呼ばれるメタデータを格納します。Data Grid は定期的にタスクを実行して、これらの Tombstone を削除し、バックアップの場所でメタデータが不要になったときに過剰なメモリー使用量を減らします。Tombstone マップのターゲットサイズとタスク実行間の最大遅延を定義することで、このタスクの頻度を設定できます。

手順

1. Data Grid 設定を開いて編集します。
2. **tombstone-map-size** 属性で保存する Tombstone の数を指定します。
   Tombstone の数がこの数を超えて増加すると、Data Grid はより頻繁にクリーンアップタスクを実行します。同様に、Tombstone の数がこの数よりも少ない場合、Data Grid はクリーンアップタスクをそれほど頻繁に実行しません。
3. **max-cleanup-delay** 属性を追加し、Tombstone クリーンアップタスク間の最大遅延をミリ秒単位で指定します。
4. 変更を設定に保存します。

Tombstone のクリーンアップタスクの設定

XML

```
<distributed-cache>
  <backups tombstone-map-size="512000" max-cleanup-delay="30000">
    <backup site="LON" strategy="ASYNC"/>
  </backups>
</distributed-cache>
```

JSON

```
{
  "distributed-cache": {
    "backups": {
      "tombstone-map-size": 512000,
      "max-cleanup-delay": 30000,
      "LON": {
        "backup": {
          "strategy": "ASYNC"
        }
      }
```

```
    }
  }
}
```

YAML

```
distributedCache:
  backups:
    tombstoneMapSize: 512000
    maxCleanupDelay: 30000
    LON:
      backup:
        strategy: "ASYNC"
```

関連情報

- Data Grid 設定スキーマ参照

2.7. クロスサイトビューの確認

クロスサイトレプリケーションを実行するように Data Grid を設定する場合、ログファイルを確認して、Data Grid クラスターがクロスサイトビューを正常に形成したことを確認する必要があります。

手順

1. 適切なエディターで Data Grid ログファイルを開きます。
2. **ISPN000439: Received new x-site view** メッセージを確認します。

たとえば、**LON** の Data Grid クラスターが **NYC** の Data Grid クラスターとクロスサイトビューを形成している場合、ログには以下のメッセージが含まれます。

```
INFO  [org.infinispan.XSITE] (jgroups-5,<server-hostname>) ISPN000439: Received new x-site view: [NYC]
INFO  [org.infinispan.XSITE] (jgroups-7,<server-hostname>) ISPN000439: Received new x-site view: [LON, NYC]
```

2.8. クロスサイトレプリケーション用の HOT ROD クライアントの設定

異なるサイトで Data Grid クラスターを使用するように Hot Rod クライアントを設定します。

hotrod-client.properties

```
# Servers at the active site
infinispan.client.hotrod.server_list = LON_host1:11222,LON_host2:11222,LON_host3:11222

# Servers at the backup site
infinispan.client.hotrod.cluster.NYC =
NYC_hostA:11222,NYC_hostB:11222,NYC_hostC:11222,NYC_hostD:11222
```

ConfigurationBuilder

```
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.addServers("LON_host1:11222;LON_host2:11222;LON_host3:11222")
      .addCluster("NYC")

.addClusterNodes("NYC_hostA:11222;NYC_hostB:11222;NYC_hostC:11222;NYC_hostD:11222")
```

ヒント

以下の方法を使用して、Hot Rod クライアントをデフォルトのクラスターまたは別のサイトのクラスターに切り替えます。

- **RemoteCacheManager.switchToDefaultCluster()**
- **RemoteCacheManager.switchToCluster(${site.name})**

関連情報

- org.infinispan.client.hotrod.configuration package description
- org.infinispan.client.hotrod.configuration.ConfigurationBuilder
- org.infinispan.client.hotrod.RemoteCacheManager

第3章 CLI を使用したクロスサイト操作の実行

Data Grid コマンドラインインターフェイス (CLI) を使用して、Data Grid Server クラスターへの接続、サイトの管理、状態遷移のバックアップの場所へのプッシュを行います。

3.1. 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。

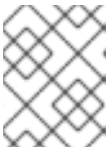
前提条件

- Data Grid への CLI 接続を作成します。

手順

1. **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。

```
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。

ヒント

--all-caches オプションを使用して、すべてのキャッシュのバックアップの場所のステータスを取得します。

2. 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。

```
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。

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

ヒント

--all-caches オプションを使用して、バックアップの場所をオンラインにするか、すべてのキャッシュでバックアップの場所をオフラインにします。

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

3.2. クロスサイト状態遷移モードの設定

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、サイト間の状態遷移操作を設定できます。または、状態遷移を手動で実行するデフォルトのモードを使用できます。

前提条件

- Data Grid への CLI 接続を作成します。

手順

1. 次の例のように、**site** コマンドを使用して状態遷移モードを設定します。
   - 現在の状態遷移モードを取得します。

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

   - キャッシュとバックアップの場所の自動状態遷移操作を設定します。

     ```
     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。

3.3. 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バックアップの場所に転送します。

前提条件

- Data Grid への CLI 接続を作成します。

手順

- 以下の例のように、**site push-site-state** コマンドを使用して、状態遷移をプッシュします。

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

ヒント

--all-caches オプションを使用して、すべてのキャッシュの状態遷移をプッシュします。

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4章 REST API を使用したクロスサイト操作の実行

Data Grid Server は、クロスサイト操作を実行するメソッドを公開する REST エンドポイントを提供します。

4.1. すべてのバックアップロケーションのステータス取得

GET リクエストですべてのバックアップロケーションのステータスを取得します。

```
GET /rest/v2/caches/{cacheName}/x-site/backups/
```

Data Grid は、以下の例のように、各バックアップロケーションのステータスを JSON 形式で応答します。

```
{
  "NYC": {
    "status": "online"
  },
  "LON": {
    "status": "mixed",
    "online": [
      "NodeA"
    ],
    "offline": [
      "NodeB"
    ]
  }
}
```

表4.1 リターンステータス

値	説明
online	ローカルクラスター内のすべてのノードには、バックアップの場所を含むクロスサイトビューがあります。
offline	ローカルクラスター内のノードには、バックアップの場所とのクロスサイトビューがありません。
mixed	ローカルクラスター内の一部のノードにはバックアップの場所を含むクロスサイトビューがあり、ローカルクラスター内の他のノードにはクロスサイトビューがありません。応答は、各ノードのステータスを示します。

4.2. 特定のバックアップの場所のステータスの取得

GET リクエストでバックアップロケーションのステータスを取得する。

```
GET /rest/v2/caches/{cacheName}/x-site/backups/{siteName}
```

Data Grid は、以下の例のように、サイト内の各ノードのステータスを JSON 形式で応答します。

```
{
  "NodeA":"offline",
  "NodeB":"online"
}
```

表4.2 リターンステータス

値	説明
online	ノードはオンラインです。
offline	ノードはオフラインです。
failed	ステータスを取得できません。リモートキャッシュがシャットダウンしているか、リクエスト中にネットワークエラーが発生した可能性があります。

4.3. バックアップの場所をオフラインにする

POST リクエストと **?action=take-offline** パラメーターを使用して、バックアップの場所をオフラインにします。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{siteName}?action=take-offline
```

4.4. バックアップの場所をオンラインにする

?action=bring-online パラメーターを使用してバックアップ場所をオンラインにします。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{siteName}?action=bring-online
```

4.5. バックアップ場所への状態のプッシュ

?action=start-push-state パラメーターを使用して、キャッシュ状態をバックアップ場所にプッシュします。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{siteName}?action=start-push-state
```

4.6. 状態遷移のキャンセル

?action=cancel-push-state パラメーターを使用して状態遷移操作をキャンセルします。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-push-state
```

4.7. 状態遷移ステータスの取得

?action=push-state-status パラメーターを使用して状態遷移操作のステータスを取得します。

```
GET /rest/v2/caches/{cacheName}/x-site/backups?action=push-state-status
```

Data Grid は、以下の例のように、各バックアップ拠点の状態移行の状況を JSON 形式で応答します。

```
{
  "NYC":"CANCELED",
  "LON":"OK"
}
```

表4.3 リターンステータス

値	説明
SENDING	バックアップ場所への状態遷移が進行中です。
OK	状態の転送が正常に完了しました。
ERROR	状態遷移でエラーが発生しました。ログファイルを確認してください。
CANCELLING	状態移行のキャンセルが進行中です。

4.8. 状態遷移ステータスのクリア

?action=clear-push-state-status パラメーターを使用して送信サイトの状態遷移ステータスをクリアします。

```
POST /rest/v2/caches/{cacheName}/x-site/local?action=clear-push-state-status
```

4.9. オフラインにする条件の変更

特定の条件が満たされると、サイトはオフラインになります。オフラインにするパラメーターを変更して、バックアップロケーションが自動的にオフラインになるタイミングを制御します。

手順

1. **GET** リクエストと **take-offline-config** パラメーターで設定されたテイクオフラインパラメーターを確認します。

   ```
   GET /rest/v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
   ```

   Data Grid のレスポンスには、以下のように **after_failures** と **min_wait** フィールドがあります。

   ```
   {
     "after_failures": 2,
     "min_wait": 1000
   }
   ```

2. **PUT** リクエストの本文のオフライン取得パラメーターを変更します。

```
PUT /rest/v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
```

操作が正常に完了すると、サービスは **204 (No Content)** を返します。

4.10. 受信サイトからの状態遷移のキャンセル

2 つのバックアップ場所間の接続が切断された場合は、プッシュを受信しているサイトでの状態遷移をキャンセルできます。

?action=cancel-receive-state パラメーターで、リモートサイトからの状態遷移をキャンセルし、ローカルキャッシュの現在の状態を維持する。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-receive-state
```

4.11. バックアップの場所のステータスの取得

GET 要求により、キャッシュ・マネージャーからすべてのバックアップ・ロケーションのステータスを取得します。

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/
```

Data Grid は、以下の例のように JSON 形式でステータスを応答します。

```
{
  "SFO-3":{
    "status":"online"
  },
  "NYC-2":{
    "status":"mixed",
    "online":[
      "CACHE_1"
    ],
    "offline":[
      "CACHE_2"
    ],
    "mixed": [
      "CACHE_3"
    ]
  }
}
```

表4.4 リターンステータス

値	説明
online	ローカルクラスター内のすべてのノードには、バックアップの場所を含むクロスサイトビューがあります。
offline	ローカルクラスター内のノードには、バックアップの場所とのクロスサイトビューがありません。

値	説明
mixed	ローカルクラスター内の一部のノードにはバックアップの場所を含むクロスサイトビューがあり、ローカルクラスター内の他のノードにはクロスサイトビューがありません。応答は、各ノードのステータスを示します。

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{site}
```

1 つのバックアップの場所のステータスを返します。

4.12. バックアップの場所をオフラインにする

?action=take-offline パラメーターで、バックアップロケーションをオフラインにします。

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=take-offline
```

4.13. バックアップの場所をオンラインにする

?action=bring-online パラメーターを使用してバックアップ場所をオンラインにします。

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=bring-online
```

4.14. 状態遷移モードの取得

GET リクエストで状態遷移モードを確認してください。

```
GET /rest/v2/caches/{cacheName}/x-site/backups/{site}/state-transfer-mode
```

4.15. 状態遷移モードの設定

?action=set パラメーターを使用して状態遷移モードを設定します。

```
POST /rest/v2/caches/{cacheName}/x-site/backups/{site}/state-transfer-mode?action=set&mode={mode}
```

4.16. 状態遷移の開始

?action=start-push-state パラメーターを使用して、すべてのキャッシュの状態をリモートサイトにプッシュします。

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=start-push-state
```

4.17. 状態遷移のキャンセル

?action=cancel-push-state パラメーターを使用して、進行中の状態遷移操作をキャンセルします。

```
POST /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=cancel-push-state
```

第5章 JMX 経由のクロスサイト操作の実行

状態遷移のプッシュや JMX 経由でサイトをオンラインにするなど、クロスサイト操作を実行します。

5.1. JMX MBEAN の登録

Data Grid は、統計の収集と管理操作の実行に使用できる JMX MBean を登録できます。統計を有効にする必要もあります。そうしないと、Data Grid は JMX MBean のすべての統計属性に **0** 値を提供します。

手順

1. Data Grid 設定を開いて編集します。
2. **jmx** 要素またはオブジェクトをキャッシュコンテナーに追加し、**enabled** 属性またはフィールドの値として **true** を指定します。
3. **domain** 属性またはフィールドを追加し、必要に応じて JMX MBean が公開されるドメインを指定します。
4. クライアント設定を保存して閉じます。

JMX の設定

XML

```
<infinispan>
  <cache-container statistics="true">
    <jmx enabled="true"
         domain="example.com"/>
  </cache-container>
</infinispan>
```

JSON

```
{
  "infinispan" : {
    "cache-container" : {
      "statistics" : "true",
      "jmx" : {
        "enabled" : "true",
        "domain" : "example.com"
      }
    }
  }
}
```

YAML

```
infinispan:
  cacheContainer:
    statistics: "true"
```

```
jmx:
  enabled: "true"
  domain: "example.com"
```

5.2. JMX クライアントを使用したクロスサイト操作の実行

JMX クライアントを使用してクロスサイト操作を実行します。

前提条件

- JMX MBean を登録するように Data Grid を設定します。

手順

1. 任意の JMX クライアントで Data Grid に接続します。
2. 以下の MBean から操作を呼び出します。
   - **XSiteAdmin** は、キャッシュのクロスサイト操作を提供します。
   - **GlobalXSiteAdminOperations** は、Cache Manager のクロスサイト操作を提供します。たとえば、サイトをオンラインに戻すには、**bringSiteOnline(siteName)** を呼び出します。

関連情報

- XSiteAdmin MBean
- GlobalXSiteAdminOperations MBean

5.3. クロスサイトレプリケーション用の JMX MBEAN

Data Grid は、統計を収集し、リモート操作を実行できるクロスサイトレプリケーションに JMX MBean を提供します。

org.infinispan:type=Cache コンポーネントは、以下の JMX MBean を提供します。

- **XSiteAdmin** は、特定のキャッシュインスタンスに適用されるクロスサイト操作を公開します。
- **RpcManager** は、クロスサイトレプリケーションのネットワーク要求に関する統計を提供します。
- **AsyncXSiteStatistics** は、キューのサイズや競合数を含む、非同期クロスサイトレプリケーションの統計を提供します。

org.infinispan:type=CacheManager コンポーネントには以下の JMX MBean が含まれます。

- **GlobalXSiteAdminOperations** は、キャッシュコンテナーのすべてのキャッシュに適用されるクロスサイト操作を公開します。

JMX MBean および利用可能な操作および統計の説明に関する詳細は、**Data Grid JMX Components** のドキュメントを参照してください。

関連情報

- Data Grid JMX Components