Red Hat Data Grid 8.4

Data Grid のアップグレード

Data Grid を 8.4 にアップグレード

Last Updated: 2024-01-27

Red Hat Data Grid 8.4 Data Grid のアップグレード

Data Grid を 8.4 にアップグレード

法律上の通知

概要

Data Grid クラスターをある 8.x バージョンから別のバージョンにアップグレードします。ローリングアップグレードを実行して、Data Grid が互換性のためにデータを変換するダウンタイムまたはオフラインアップグレードを回避できます。

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。

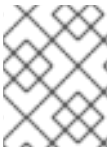
DATA GRID のドキュメント

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 DATA GRID 8 のアップグレードに関する注意点

Data Grid 8 のバージョンから別のバージョンにアップグレードする前に、このセクションの詳細を確認してください。

1.1. DATA GRID 8.4 へのアップグレード

以下の情報を参照して、以前のバージョンの Data Grid 8 から 8.4 へのアップグレードが成功するようにします。

Hot Rod クライアントのデフォルト

Data Grid 8.4.6 では、Hot Rod クライアントのプロパティーに変更が導入されました。

infinispan.client.hotrod.ssl_hostname_validation

新しいプロパティー **infinispan.client.hotrod.ssl_hostname_validation** のデフォルト値は **true** です。このプロパティーにより、RFC 2818 ルールに基づいた TLS ホスト名の検証が有効になります。さらに、ホスト名の検証が有効な場合は、**infinispan.client.hotrod.sni_host_name** の設定が必須になりました。

表1.1 デフォルトのプロパティーの変更

プロパティー	Data Grid 8.4	以前のバージョン
infinispan.client.hotrod.connect_timeout	**2000 ミリ秒/2 秒**	60000 ミリ秒/60 秒
infinispan.client.hotrod.socket_timeout	**2000 ミリ秒/2 秒**	60000 ミリ秒/60 秒
infinispan.client.hotrod.max_retries	**3**	10
infinispan.client.hotrod.min_evictable_idle_time	**180000 ミリ秒/3 分**	1800000 ミリ秒/30 分

JGroups およびクロスサイトメトリクスのメトリクス命名方法の改善

Data Grid 8.4.4 では、JGroups メトリクスおよびクロスサイトメトリクスの **name-as-tags** プロパティーを有効にできます。

name-as-tags を有効にすると、メトリクスが簡素化され、クラスター名とサイト名がメトリクス名に含まれるのではなく、タグとして表示されます。

name-as-tags を **false** に設定すると、メトリクスがチャネルに基づいて命名され、同じ目的のメトリクスが複数作成されます。

```
# TYPE vendor_jgroups_xsite_frag4_get_number_of_sent_fragments gauge
# HELP vendor_jgroups_xsite_frag4_get_number_of_sent_fragments Number of sent fragments
vendor_jgroups_xsite_frag4_get_number_of_sent_fragments{cluster="xsite",node="..."} 0.0
# TYPE vendor_jgroups_cluster_frag4_get_number_of_sent_fragments gauge
# HELP vendor_jgroups_cluster_frag4_get_number_of_sent_fragments Number of sent fragments
vendor_jgroups_cluster_frag4_get_number_of_sent_fragments{cluster="cluster",node="..."} 2.0
```

name-as-tags を **true** に設定すると、メトリクスが簡素化され、クラスター名とサイト名がタグとして表示されます。

```
# TYPE vendor_jgroups_frag4_get_number_of_sent_fragments gauge
# HELP vendor_jgroups_frag4_get_number_of_sent_fragments Number of sent fragments
vendor_jgroups_frag4_get_number_of_sent_fragments{cache_manager="default",cluster="xsite",node="..."} 0.0
vendor_jgroups_frag4_get_number_of_sent_fragments{cache_manager="default",cluster="cluster",node="..."} 2.0
```

メトリクスが簡素化されるだけでなく、クラスター名とサイト名を変更してもメトリクス名の一貫性が保たれるため、Grafana ダッシュボードを更新する必要がなくなります。

Java 8 からの移行

Red Hat は Data Grid 8.4 の時点で、Data Grid Server のインストール、Hot Rod Java クライアント、およびカスタムアプリケーションの組み込みキャッシュに Data Grid を使用する場合に、Java 11 および Java 17 をサポートします。Data Grid ユーザーは、アプリケーションを少なくとも Java 11 にアップグレードする必要があります。Java 8 のサポートは Data Grid 8.2 で非推奨となり、Data Grid 8.4 で削除されました。

埋め込みキャッシュ

Red Hat は、カスタムアプリケーションでの組み込みキャッシュに、Data Grid を使用する場合に Java 11 および Java 17 をサポートします。Data Grid ユーザーは、アプリケーションを少なくとも Java 11 にアップグレードする必要があります。

リモートキャッシュ

Red Hat は、Data Grid Server および Hot Rod Java クライアントで Java 11 および Java 17 をサポートします。

Java 8 を必要とするアプリケーションで実行している Hot Rod Java クライアントは、古いバージョンのクライアントライブラリーを引き続き使用できます。Red Hat は、最新の Data Grid Server バージョンと組み合わせて、以前の Hot Rod Java クライアントバージョンの使用をサポートしています。ただし、古いバージョンのクライアントを使用し続けると、修正や機能強化が失われることになります。

重要

OpenJDK 17 では、Nashorn JavaScript Engine、その API、および **jjs** ツールへの対応が解除されました。Data Grid Server が自動化タスクに JavaScript を使用する場合は、Nashorn JavaScript エンジンをインストールする必要があります。

Jakarta EE 依存関係の追加

バージョン 8.4 以降、Data Grid は Jakarta EE 9+ ベースの jar を配布します。アプリケーションで Jakarta 固有の依存関係が必要な場合は、アーティファクトに **-jakarta** を追加します。次に例を示します。

pom.xml

```
<dependency>
  <groupId>org.infinispan</groupId>
  <artifactId>infinispan-client-hotrod-jakarta</artifactId>
</dependency>
```

ストアプロパティー設定の更新

Data Grid 8.4 では、ストアが提供するプロパティーが、ストアの明示的な設定をオーバーライドしなくなりました。以下の例では、設定を更新する方法を紹介しています。

この例では、ストアの読み取り専用設定を **true** に設定します。

example.xml

```
<persistence>
  <file-store>
    <index path="testCache/index" />
    <data path="testCache/data" />
    <property name="readOnly">true</property>
  </file-store>
</persistence>
```

以下の更新された例では、**read-only** プロパティーがストア自体に提供されており、ストアの明示的な設定には影響しません。

example.xml

```
<persistence>
  <file-store read-only="true">
    <index path="testCache/index" />
    <data path="testCache/data" />
  </file-store>
</persistence>
```

関連情報

- Data Grid 8 への移行

非推奨の機能

Data Grid 8.4.3 では、次の機能は非推奨となり、将来の Data Grid リリースでは削除される予定です。

Red Hat は、現在のリリースのライフサイクル中にこれらの機能のサポートを提供しますが、拡張機能は提供されなくなり、最終的には削除されます。将来の互換性を確保するために、代替ソリューションに移行することを推奨します。

Java 11 のサポート

Java 11 のサポートは非推奨となり、Data Grid バージョン 8.5 で削除される予定です。Data Grid 8.5 のユーザーは、アプリケーションを少なくとも Java 17 にアップグレードする必要があります。

古い Hot Rod Java クライアントバージョンを最新の Data Grid Server バージョンと組み合わせて引き続き使用できます。ただし、古いバージョンのクライアントを使用し続けると、修正や機能強化が失われることになります。

Java EE 依存関係のサポート

Java EE 依存関係のサポートは非推奨となり、Data Grid バージョン 8.5 で削除される予定です。進化する Java エンタープライズエコシステムに合わせて、Jakarta EE の依存関係と API に移行します。

Spring 5.x および Spring Boot 2.x のサポート

Spring Boot 2.x および Spring 5.x のサポートは非推奨となり、バージョン Data Grid 8.5 で削除される予定です。今後の Data Grid リリースとの互換性を確保するために、Spring Boot および Spring Framework の新しいバージョンに移行します。

JCache のサポート

JCache (JSR 107) のサポートは非推奨となり、Data Grid バージョン 8.5 で削除される予定です。代わりに、Jakarta EE エコシステムの他のキャッシュ API 開発を使用してください。

Red Hat JBoss EAP の Data Grid モジュールの非推奨
Data Grid リリースの一部として配布されている Red Hat JBoss EAP アプリケーションの Data Grid モジュールは非推奨となり、Data Grid バージョン 8.5 で削除される予定です。

JBoss EAP ユーザーは、Data Grid モジュールを個別にインストールすることなく、JBoss EAP 製品リリースに統合されている **infinispan** サブシステムを使用できます。

散在 (scattered) キャッシュモード
散在キャッシュモードは非推奨となり、Data Grid バージョン 8.5 で削除される予定です。散在キャッシュの代わりに、代わりに分散キャッシュを使用できます。

Data Grid Console または REST API を使用して ignore リストにキャッシュを追加する
Data Grid Console または REST API を使用してキャッシュを ignore リストに追加し、クライアントリクエストから特定のキャッシュを一時的に除外できる機能は、非推奨になりました。この機能は、今後のリリースでは削除される予定です。

Cache サービスタイプ
Cache サービスタイプは非推奨で、Data Grid 8.5 で削除される予定です。**Cache** サービスタイプは、最小設定でレイテンシーを低く抑えたデータストアを便利に作成できるように設計されています。**DataGrid** サービスタイプを使用して、クラスターのアップグレードやデータ移行などの複雑な操作を自動化します。

Windows での Data Grid Server のテスト
Windows Server 2019 での Data Grid Server のサポートは非推奨となり、Data Grid 8.5 で削除される予定です。ただし、Data Grid チームは、Windows Server 2019 を使用した C++ Hot Rod クライアントのテストを継続します。

PrincipalRoleMapperContext インターフェイスが非推奨に
org.infinispan.security.PrincipalRoleMapperContext は Data Grid 8.4 で非推奨となり、**org.infinispan.security.AuthorizationMapperContext** に置き換えられました。

fetch-state ストアプロパティーの削除
fetch-state 属性は非推奨となり、Data Grid 8.4 で削除され、後継となる属性はありません。使用中の xml 設定からこの属性を削除できます。

今回の変更は、同じデータにアクセスできる共有ストアには影響はありません。ローカルキャッシュストアは、「purge on startup」を使用して、永続ストレージから古いエントリーをロードしないようにすることができます。

最低でも 8.1 からのアップグレード
8.0 からアップグレードする場合は、最初に 8.1 にアップグレードする必要があります。Data Grid 8.0 の永続データは、新しいバージョンのバイナリーではありません。この非互換性の問題に対処するために、Data Grid 8.2 以降では、クラスター起動時に Data Grid 8.1 から既存の永続キャッシュストアを自動的に変換します。ただし、Data Grid は、Data Grid 8.0 からキャッシュストアを変換しません。

第2章 DATA GRID SERVER クラスターのローリングアップグレードの実行

Data Grid クラスターのローリングアップグレードを実行して、ダウンタイムやデータの損失なしにバージョン間で変更し、Hot Rod プロトコルを介してデータを移行します。

2.1. ターゲット DATA GRID クラスターの設定

アップグレードする予定の Data Grid バージョンを使用するクラスターを作成してから、リモートキャッシュストアを使用してソースクラスターをターゲットクラスターに接続します。

前提条件

- ターゲットクラスターに必要なバージョンの Data Grid Server ノードをインストールします。

重要

ターゲットクラスターのネットワークプロパティーはソースクラスターのネットワークプロパティーが重複していないことを確認します。JGroups トランスポート設定でターゲットおよびソースクラスターの一意の名前を指定する必要があります。環境に応じて、異なるネットワークインターフェイスとポートオフセットを使用して、ターゲットクラスターとソースクラスターを分離することもできます。

手順

1. ターゲットクラスターがソースクラスターに接続できるリモートキャッシュストア設定を JSON 形式で作成します。
   ターゲットクラスターのリモートキャッシュストアは、Hot Rod プロトコルを使用して、ソースクラスターからデータを取得します。

```
{
  "remote-store": {
    "cache": "myCache",
    "shared": true,
    "raw-values": true,
    "security": {
      "authentication": {
        "digest": {
          "username": "username",
          "password": "changeme",
          "realm": "default"
        }
      }
    },
    "remote-server": [
      {
        "host": "127.0.0.1",
        "port": 12222
      }
    ]
  }
}
```

2. Data Grid コマンドラインインターフェイス (CLI) または REST API を使用して、リモートキャッシュストア設定をターゲットクラスターに追加し、ソースクラスターに接続できるようにします。

   - CLI: ターゲットクラスターで **migrate cluster connect** コマンドを使用します。

     ```
     [//containers/default]> migrate cluster connect -c myCache --file=remote-store.json
     ```

   - REST API: **rolling-upgrade/source-connection** メソッドを使用して、ペイロードにリモートストア設定が含まれる POST リクエストを呼び出します。

     ```
     POST /rest/v2/caches/myCache/rolling-upgrade/source-connection
     ```

3. 移行するキャッシュごとに直前の手順を繰り返します。

4. すべての要求の処理を開始するために、クライアントをターゲットクラスターに切り替えます。

   a. クライアント設定をターゲットクラスターの場所で更新します。

   b. クライアントを再起動します。

関連情報

- リモートキャッシュストア設定スキーマ

2.2. ターゲットクラスターへのデータの同期

ターゲット Data Grid クラスターを設定してソースクラスターに接続する場合、ターゲットクラスターはリモートキャッシュストアを使用してクライアント要求を処理し、オンデマンドでデータをロードできます。データをターゲットクラスターに完全に移行して、ソースクラスターの使用を停止できるようにするには、データを同期します。この操作はソースクラスターからデータを読み取り、ターゲットクラスターに書き込みます。データは、ターゲットクラスターのすべてのノードに並行して移行され、各ノードはデータのサブセットを受け取ります。ターゲットクラスターに移行する各キャッシュの同期を実行する必要があります。

前提条件

- 適切な Data Grid バージョンでターゲットクラスターを設定している。

手順

1. ターゲットクラスターに移行する各キャッシュと Data Grid コマンドラインインターフェイス (CLI) または REST API との同期を開始します。

   - CLI: **migrate cluster synchronize** コマンドを使用します。

     ```
     migrate cluster synchronize -c myCache
     ```

   - REST API: POST リクエストと共に **?action=sync-data** パラメーターを使用します。

     ```
     POST /rest/v2/caches/myCache?action=sync-data
     ```

操作が完了すると、Data Grid はターゲットクラスターにコピーされたエントリーの合計数で応答します。

2. ターゲットクラスター内の各ノードをソースクラスターから切断します。

- CLI: **migrate cluster disconnect** コマンドを使用します。

```
migrate cluster disconnect -c myCache
```

- REST API: DELETE リクエストを呼び出します。

```
DELETE /rest/v2/caches/myCache/rolling-upgrade/source-connection
```

次のステップ

ソースクラスターからすべてのデータを同期すると、ローリングアップグレードプロセスが完了します。ソースクラスターの使用を停止できるようになりました。

第3章 キャッシュストア間のデータの移行

Data Grid は、キャッシュストア間で永続化されたデータを移行するための Java ユーティリティーを提供します。

Data Grid をアップグレードする場合、メジャーバージョン間の機能相違点は、キャッシュストア間の後方互換性を許可しません。**StoreMigrator** を使用してデータを変換し、ターゲットバージョンとの互換性を持つことができます。

たとえば、Data Grid 8.0 にアップグレードすると、デフォルトのマーシャラーが Protostream に変更になります。以前の Data Grid バージョンでは、キャッシュストアはバイナリー形式を使用し、マーシャリングする変更との互換性がありません。つまり、Data Grid 8.0 は、以前の Data Grid バージョンでキャッシュストアから読み込むことができません。

他の場合は、Data Grid のバージョンが、JDBC Mixed および Binary ストアなどのキャッシュストア実装を非推奨または削除します。このような場合は、**StoreMigrator** を使用して異なるキャッシュストア実装に変換できます。

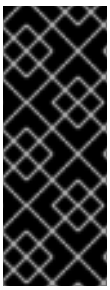
3.1. キャッシュストアマイグレーター

Data Grid は、最新の Data Grid キャッシュストア実装のデータを再作成する **StoreMigrator.java** ユーティリティーを提供します。

StoreMigrator は以前のバージョンの Data Grid のキャッシュストアを取得し、キャッシュストア実装をターゲットとして使用します。

StoreMigrator を実行すると、**EmbeddedCacheManager** インターフェイスを使用して定義したキャッシュストアタイプでターゲットキャッシュが作成されます。**StoreMigrator** は、ソースストアからメモリーにエントリーを読み込み、それらをターゲットキャッシュに配置します。

StoreMigrator を使用すると、あるタイプのキャッシュストアから別のストアにデータを移行することもできます。たとえば、JDBC String ベースのキャッシュストアから RocksDB キャッシュストアに移行することができます。



重要

StoreMigrator は、セグメント化されたキャッシュストアから以下にデータを移行できません。

- 非セグメント化されたキャッシュストア。
- セグメント数が異なるセグメント化されたキャッシュストア。

3.2. キャッシュストアマイグレーターの設定

migrator.properties ファイルを使用して、ソースおよびターゲットのキャッシュストアのプロパティーを設定します。

手順

1. **migrator.properties** ファイルを作成します。
2. **migrator.properties** ファイルを使用して、ソースおよびターゲットのキャッシュストアのプロパティーを設定します。
   - **source.** 接頭辞をソースキャッシュストアの全設定プロパティーに追加します。

a. **source.** 接頭辞をソースキャッシュストアの全設定プロパティーに追加します。

ソースキャッシュストアの例

```
source.type=SOFT_INDEX_FILE_STORE
source.cache_name=myCache
source.location=/path/to/source/sifs
source.version=<version>
```

重要

セグメント化されたキャッシュストアからデータを移行するには、**source.segment_count** プロパティーを使用してセグメント数も設定する必要があります。セグメント数は、Data Grid 設定の **clustering.hash.numSegments** と一致させる必要があります。キャッシュストアのセグメント数が対応するキャッシュのセグメント数と一致しないと、Data Grid がキャッシュストアからデータを読み取ることができません。

b. **target.** 接頭辞をターゲットキャッシュストアの全設定プロパティーに追加します。

ターゲットキャッシュストアの例

```
target.type=SINGLE_FILE_STORE
target.cache_name=myCache
target.location=/path/to/target/sfs.dat
```

3.2.1. キャッシュストアマイグレーターの設定プロパティー

ソースおよびターゲットのキャッシュストアを **StoreMigrator** プロパティーで設定します。

表3.1 キャッシュストアタイププロパティー

プロパティー	説明	必須/オプション

プロパティー	説明	必須/オプション
type	ソースまたはターゲットのキャッシュストアのタイプを指定します。 **.type=JDBC_STRING** **.type=JDBC_BINARY** **.type=JDBC_MIXED** **.type=LEVELDB** **.type=ROCKSDB** **.type=SINGLE_FILE_STORE** **.type=SOFT_INDEX_FILE_STORE** **.type=JDBC_MIXED**	必須

表3.2 一般的なプロパティー

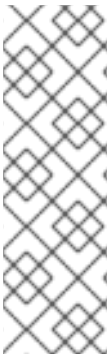
プロパティー	説明	値の例	必須/オプション
cache_name	バックアップするキャッシュの名前。	**.cache_name=myCache**	必須
segment_count	セグメンテーションを使用できるターゲットキャッシュストアのセグメント数。 セグメント数は、Data Grid 設定の **clustering.hash.numSegments** と一致させる必要があります。キャッシュストアのセグメント数が対応するキャッシュのセグメント数と一致しないと、Data Grid がキャッシュストアからデータを読み取ることができません。	**.segment_count=256**	任意

表3.3 JDBC プロパティー

プロパティー	説明	必須/オプション

プロパティー	説明	必須/オプション
dialect	基礎となるデータベースのダイアレクトを指定します。	必須
version	ソースキャッシュストアのマーシャラーバージョンを指定します。 以下のいずれかの値を設定します。 * Data Grid 7.2.x の場合は **8** * Data Grid 7.3.x の場合は **9** * Data Grid 8.0.x の場合は **10** * Data Grid 8.1.x の場合は **11** * Data Grid 8.2.x の場合は **12** * Data Grid 8.3.x の場合は **13**	ソースストアにのみ必要です。
marshaller.class	カスタムマーシャラークラスを指定します。	カスタムマーシャラーを使用する場合に必要です。
marshaller.externalizers	**[id]:<Externalizer class>** 形式で読み込むカスタム **AdvancedExternalizer** 実装のコンマ区切りリストを指定します。	任意
connection_pool.connection_url	JDBC 接続 URL を指定します。	必須
connection_pool.driver_class	JDBC ドライバーのクラスを指定します。	必須
connection_pool.username	データベースユーザー名を指定します。	必須
connection_pool.password	データベースユーザー名のパスワードを指定します。	必須
db.disable_upsert	データベース upsert を無効にします。	任意
db.disable_indexing	テーブルインデックスが作成されるかどうかを指定します。	任意

プロパティー	説明	必須/オプション
table.string.table_name_prefix	テーブル名の追加接頭辞を指定します。	任意
table.string.<id|data|timestamp>.name	列名を指定します。	必須
table.string.<id|data|timestamp>.type	列タイプを指定します。	必須
key_to_string_mapper	**TwoWayKey2StringMapper** クラスを指定します。	任意



注記

Binary キャッシュストアから古い Data Grid バージョンの移行には、以下のプロパティーで **table.string.*** を **table.binary.*** に変更します。

- **source.table.binary.table_name_prefix**
- **source.table.binary.<id\|data\|timestamp>.name**
- **source.table.binary.<id\|data\|timestamp>.type**

```
# Example configuration for migrating to a JDBC String-Based cache store
target.type=STRING
target.cache_name=myCache
target.dialect=POSTGRES
target.marshaller.class=org.example.CustomMarshaller
target.marshaller.externalizers=25:Externalizer1,org.example.Externalizer2
target.connection_pool.connection_url=jdbc:postgresql:postgres
target.connection_pool.driver_class=org.postrgesql.Driver
target.connection_pool.username=postgres
target.connection_pool.password=redhat
target.db.disable_upsert=false
target.db.disable_indexing=false
target.table.string.table_name_prefix=tablePrefix
target.table.string.id.name=id_column
target.table.string.data.name=datum_column
target.table.string.timestamp.name=timestamp_column
target.table.string.id.type=VARCHAR
target.table.string.data.type=bytea
target.table.string.timestamp.type=BIGINT
target.key_to_string_mapper=org.infinispan.persistence.keymappers.
DefaultTwoWayKey2StringMapper
```

表3.4 RocksDB プロパティー

プロパティー	説明	必須/オプション
location	データベースディレクトリーを設定します。	必須
圧縮	使用する圧縮タイプを指定します。	任意

```
# Example configuration for migrating from a RocksDB cache store.
source.type=ROCKSDB
source.cache_name=myCache
source.location=/path/to/rocksdb/database
source.compression=SNAPPY
```

表3.5 SingleFileStore プロパティー

プロパティー	説明	必須/オプション
location	キャッシュストア **.dat** ファイルが含まれるディレクトリーを設定します。	必須

```
# Example configuration for migrating to a Single File cache store.
target.type=SINGLE_FILE_STORE
target.cache_name=myCache
target.location=/path/to/sfs.dat
```

表3.6 SoftIndexFileStore プロパティー

プロパティー	説明	値
必須/オプション	**location**	データベースディレクトリーを設定します。
必須	**index_location**	データベースインデックスディレクトリーを設定します。

```
# Example configuration for migrating to a Soft-Index File cache store.
target.type=SOFT_INDEX_FILE_STORE
target.cache_name=myCache
target.location=path/to/sifs/database
target.location=path/to/sifs/index
```

3.3. DATA GRID キャッシュストアの移行

StoreMigrator を使用して、異なる Data Grid バージョンのキャッシュストア間でデータを移行したり、別のタイプのキャッシュストアにデータを移行したりできます。

前提条件

- **infinispan-tools.jar** がある。
- ソースとターゲットのキャッシュストアを **migrator.properties** ファイルで設定している。

手順

- ソースコードから **infinispan-tools.jar** をビルドした場合は、次の手順を実行します。
  1. **infinispan-tools.jar** をクラスパスに追加します。
  2. JDBC ドライバーなど、ソースデータベースとターゲットデータベースの依存関係をクラスパスに追加します。
  3. **migrator.properties** ファイルを **StoreMigrator** の引数として指定します。
- Maven リポジトリーから **infinispan-tools.jar** をプルした場合は、以下のコマンドを実行します。

```
mvn exec:java
```