Red Hat JBoss Data Virtualization 6.3
管理および設定ガイド

管理者向け

Red Hat Customer Content Services

Red Hat JBoss Data Virtualization 6.3 管理および設定ガイド

管理者向け

Red Hat Customer Content Services

法律上の通知

Copyright © 2015 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では Red Hat JBoss Data Virtualization の管理方法について説明しています。

目次

パート I. はじめに

第1章 注意事項

1.1. データのバックアップ

警告

Red Hat は、システム設定をバックアップしてから本書に掲載されている設定タスクを実行することを推奨します。

1.2. 変数名: EAP_HOME

EAP_HOME は、JBoss Data Virtualization がデプロイされた Red Hat JBoss Enterprise Application Platform インストールのルートディレクトリーを示しています。

1.3. 変数名: MODE

MODE は、JBoss Data Virtualization が実行されているモードがスタンドアロンまたはドメインモードであるかによって **standalone** または **domain** のいずれかになります。本書に記載されているファイルパスに **MODE** がある場合は、これらのどちらかに置き換えてください (この変数は、ディレクトリー構造のどこに製品がインストールされているかに応じて、ご自身で設定する必要があります)。

1.4. Red Hat のドキュメントサイト

Red Hat の公式ドキュメントサイトは https://access.redhat.com/site/documentation/ になります。本書を含む最新バージョンのドキュメントをご覧になれます。

パート II. JBoss Data Virtualization 管理の概要

第2章 管理ツール

2.1. 管理ツールの概要

Red Hat JBoss Data Virtualization の管理に使用できるツールは次のとおりです。

管理コンソール

管理コマンドラインインターフェース (CLI)

AdminShell

管理 API

JBoss Operations Network

2.2. 管理コンソール

2.2.1. 管理コンソールおよび JBoss Data Virtualization

Red Hat JBoss Data Virtualization は、Web ベースの管理コンソールにプラグインを提供します。JBoss EAP インスタンスの実行時にデプロイされるデータ仮想化サービスの設定および監視を可能にする、Web インターフェースを提供します。

注記

JBoss Data Virtualization に固有でない管理コンソールの一般的な使用の詳細については、『Red Hat JBoss Enterprise Application Platform 管理および設定ガイド』を参照してください。

2.2.2. 管理コンソールにログインします。

前提条件

- JBoss EAP 6 が稼働している必要があります。
- コンソールへのアクセス権限を持つユーザーが作成済みである必要があります。

1. Web ブラウザーを起動し、http://localhost:9990/console/App.html へ移動します。

注記

ポート 9990 は、管理コンソールソケットバインディングとして事前定義されています。

2. ユーザー名とパスワードを入力し、管理コンソールへログインします。

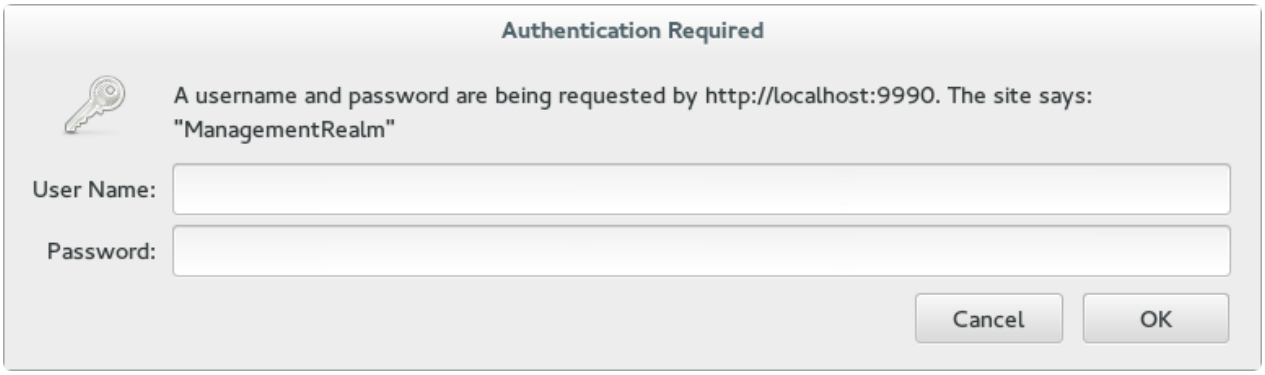


図2.1 管理コンソールのログイン画面

結果:

ログイン後、 http://localhost:9990/console/App.html#home へリダイレクトされ、管理コンソールのランディングページが表示されます。

2.2.3. 管理コンソール - Configuration タブ

`Configuration` タブには、一般的な設定プロパティーとデータ仮想化固有の設定プロパティーの両方が含まれています。左側のナビゲーションツリーにある **`Teiid`** をクリックし、Teiid の設定を表示します。サブカテゴリーは次のとおりです。

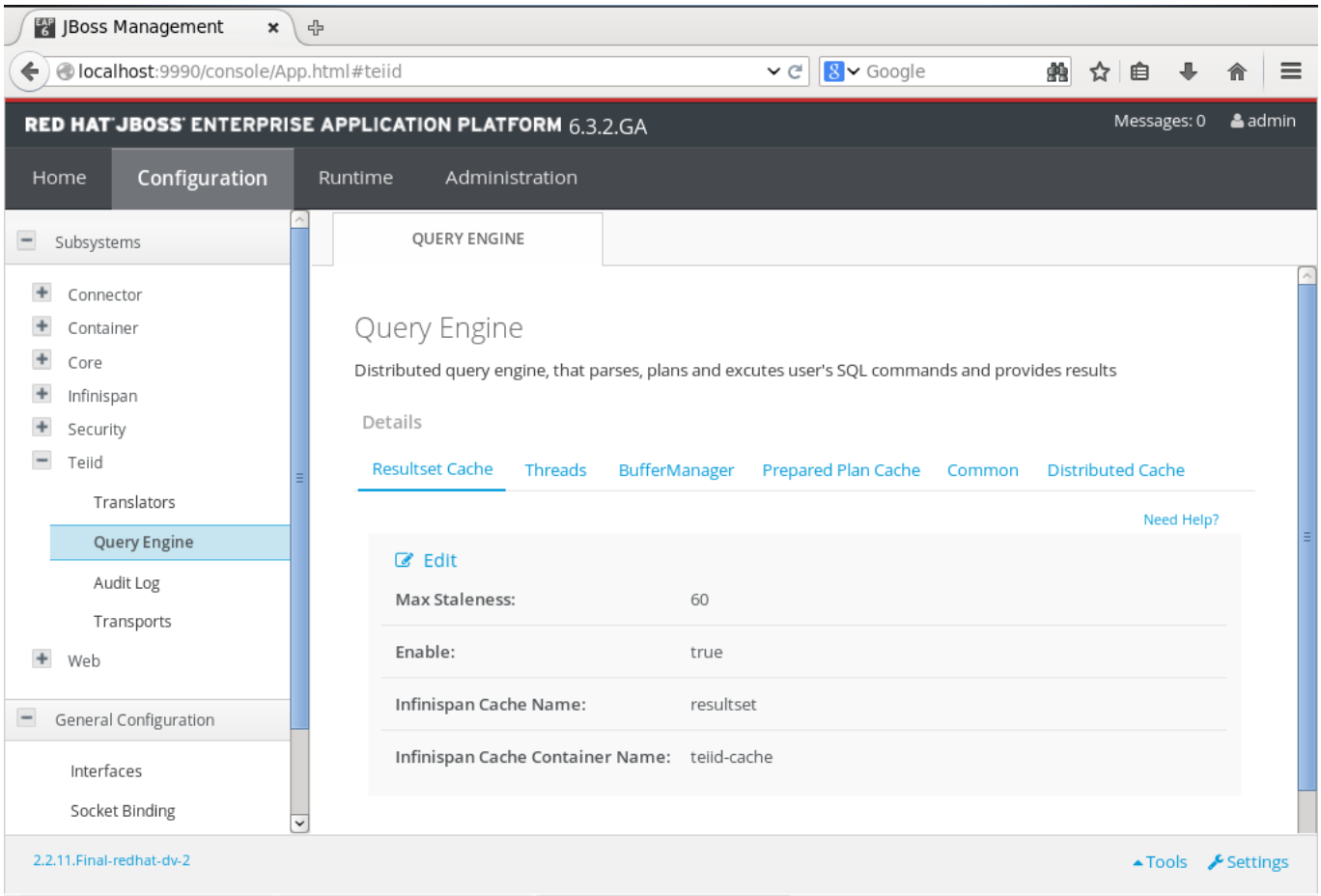


図2.2 Configuration タブ

- Query Engine - コアの Teiid エンジンプロパティーを表示および設定します。

- Translators - Teiid で設定されたトランスレーターを表示および削除します。
- Transports - トランスポートを表示し、Teiid エンジンに追加したり、削除したりします。
- Audit Logs - Teiid の監査ロギングを有効または無効にします。

Configuration タブを使用して、Teiid の設定内容を変更します。設定プロパティーはタブごとに分類されています。各タブの **Need Help** リンクをクリックすると、各フィールドの詳細が表示されます。

注記

一部のプロパティーは、サーバーを再起動して変更を反映させる必要があります。

2.2.4. 管理コンソール - Runtime タブ

Runtime タブには、JBoss EAP サーバーと Teiid サブシステムに関する実行時情報が表示されます。左側のナビゲーションツリーにある **Virtual Databases** をクリックすると、Teiid に関する実行時情報が表示されます。

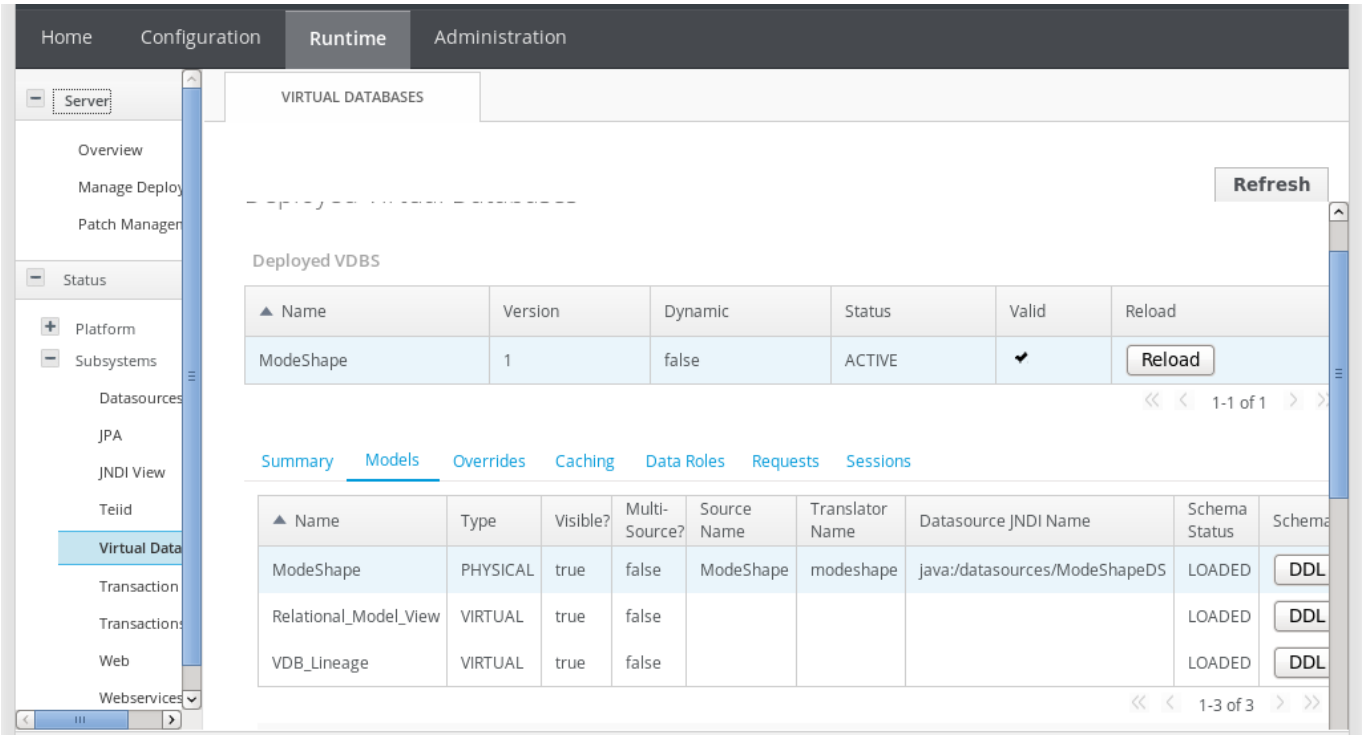


図2.3 Runtime タブ

サーバーにデプロイされたすべての VDB は最上部の表に表示されます。VDB を選択し、ハイライトすると、その VDB に関する詳細が下の表に表示されます。これらの各タブは、機能のグループに分割されます。

- Summary: このパネルには、選択された VDB、その VDB に関連するプロパティー、およびこの VDB がインポートする他の VDB の詳細が表示されます。このタブは、VDB ステータスの概要を提供するよう設計されています。
- Models: このパネルには、該当する VDB に定義されたすべてのモデルと、各モデルのトランスレーター名とソース接続 JNDI 名が表示されます。また、各モデルのタイプとそれがマルチソースであるかどうかも示されます。特定のモデルが選択された場合は、定義されたそのモデルのすべてのプロパティーと、モデルに関連付けられたすべてのエラーも表示されます。VDB がアクティブなステータスでデプロイさ

れていない場合は、これらのエラーと修正を検証し、デプロイメントの問題を解決する必要があります。

Models パネルの **DDL** ボタンをクリックすると、該当するモデルのスキーマが表示されます。これは、Teiid Designer を使用して設計された XML ベースのモデルには適用されません。このツールでは、JNDI 名をダブルクリックして編集することが可能です。これは、該当する環境で JNDI 名を変更する場合に役に立ちます。

- Overrides: このパネルには、Teiid Designer でオーバーライドされたすべてのトランスレーターとそのプロパティーが表示されます。
- Teiid: このスクリーンでは実行時の Teiid メトリックスが表示されます。
- Caching: Caching パネルには、キャシュがどれぐらい効果的に使用されるかなど結果セットキャッシュのキャッシュ統計が表示されます。また、VDB のすべての内部マテリアライズビューと、これらがいつロードされたかなどのロードステータスも表示されます。さらに、ソースからコンテンツを更新またはリロードできるように特定のビューまたは VDB 内のすべてのビューを無効にするオプションが提供されます。このパネルは、結果セットキャッシュ内容全体または選択された VDB の準備済み計画キャッシュ内容を破棄する UI も提供します。
- Data Roles: Data Roles パネルには、Teiid Designer を使用して VDB で定義されたすべてのポリシーまたは vdb.xml ファイルでハンドコーディングされたすべてのポリシーが表示されます。選択した各ポリシーに対して、ユーザーが所有する権限の種類などのそのポリシーのパーミッションがリストされ、そのポリシーに対してマップされたエンタープライズロール割り当てが表示されます。エンタープライズロールはこの UI を使用してポリシーに対して追加または削除できます。
- Requests: このパネルには、VDB の選択時に選択した VDB に対する現在のすべての要求が表示されます。"refresh" をクリックすると、最新の日付要求が取得されます。パネルの上部テーブルには、teiid エンジンにユーザーが送信した要求が表示されます。これらのいずれかの要求が選択されると、下部テーブルに Teiid エンジンにより物理ソースに送信されたすべてのソースレベルクエリーが表示されます。この UI を使用して、ユーザーはキャンセル要求をユーザーレベルクエリーに送信することもできます。キャンセルは非同期操作であるため、この操作は保証されません (キャンセルが送信されるまでにクエリーが完了しないことがあります)。
- Sessions: このパネルには、選択した VDB に接続されたすべてのアクティブセッションが表示されます。また、接続プロパティーと選択したセッションまたはすべてのセッションを修了するオプションも表示されます。

2.2.5. 管理コンソール - Administration タブ

Administration タブを使用してロールとグループを設定できます。ユーザーを含めたり、除外したりするロールは管理コンソールで設定できます。また、ロールに含めたり、ロールから除外したりするグループを設定することもできます。この設定は、SuperUser または Administrator ロールのユーザーのみが実行できます。

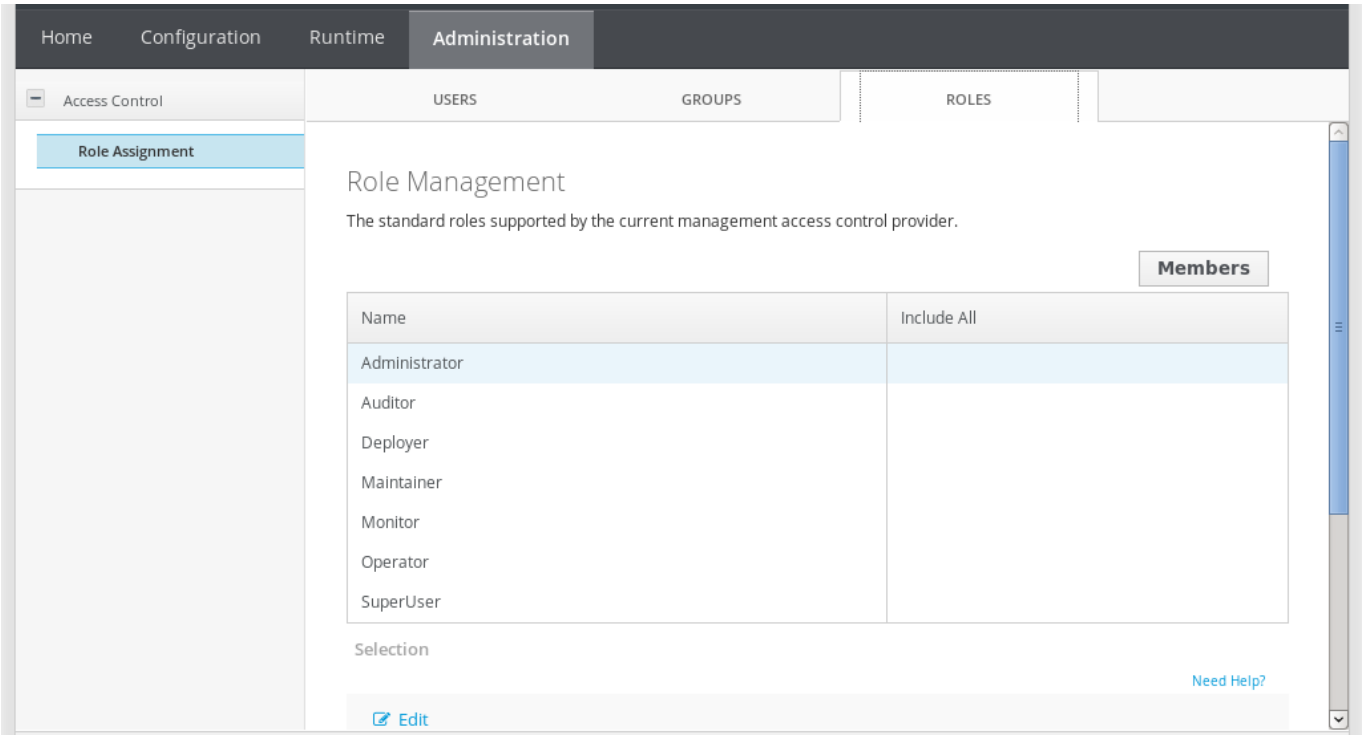


図2.4 Administration タブ

ユーザー管理の詳細については、『Red JBoss EAP 管理および設定ガイド』を参照してください。

2.3. 管理 CLI

2.3.1. 管理 CLI の起動

手順2.1 Linux または Microsoft Windows Server での CLI の起動

A. **Linux での CLI の起動**

コマンドラインで以下のコマンドを入力して、***EAP_HOME*/bin/jboss-cli.sh** ファイルを実行します。

```
$ EAP_HOME/bin/jboss-cli.sh
```

B. **Microsoft Windows Server での CLI の起動**

ダブルクリックするか、コマンドラインで以下のコマンドを入力して、***EAP_HOME*\bin\jboss-cli.bat** ファイルを実行します。

```
C:\>EAP_HOME\bin\jboss-cli.bat
```

2.3.2. 管理 CLI の終了

管理 CLI で、**quit** コマンドを入力します。

```
[domain@localhost:9999 /] quit
```

2.3.3. 管理 CLI を使用した管理対象サーバーインスタンスへの接続

前提条件

- 「管理 CLI の起動」

手順2.2 管理対象サーバーインスタンスへの接続

- **connect コマンドの実行**

  管理 CLI で、**connect** コマンドを入力します。

```
[disconnected /] connect
Connected to domain controller at localhost:9999
```

A. Linux システムで管理 CLI を起動するときに管理対象サーバーへ接続するには、***--connect*** パラメーターを使用します。

```
$ EAP_HOME/bin/jboss-cli.sh --connect
```

B. ***--connect*** パラメーターは、ホストとサーバーのポートを指定するために使用できます。ポートの値が **9999** であるアドレス **192.168.0.1** に接続するには、次のコマンドを使用します。

```
$ EAP_HOME/bin/jboss-cli.sh --connect --controller=192.168.0.1:9999
```

2.3.4. 管理 CLI でのヘルプの取得

サマリー

CLI コマンドを学ぶ必要がある場合や何を行ったらいいかわからない場合に、ガイダンスが必要になることがあります。管理 CLI には、一般的なオプションと状況依存オプションから構成されるヘルプダイアログが組み込まれてます (処理状況に依存するヘルプコマンドでは、スタンドアロンまたはドメインコントローラーへの接続を確立する必要があります。接続が確立されない限り、これらのコマンドはリストに表示されません)。

前提条件

- 「管理 CLI の起動」

1. **一般的なヘルプの場合**

   管理 CLI で、**help** コマンドを入力します。

```
[standalone@localhost:9999 /] help
```

2. **状況依存ヘルプの取得**

   管理 CLI で、**help --commands** 拡張コマンドを入力します。

```
[standalone@localhost:9999 /] help --commands
```

3. 特定のコマンドの詳細な説明については、そのコマンドとその後に ***--help*** を入力してください。

```
[standalone@localhost:9999 /] deploy --help
```

結果:

CLI ヘルプ情報が表示されます。

2.4. AdminShell

2.4.1. AdminShell 機能

AdminShell は以下の機能を提供します。

管理

AdminShell を使用すると、さまざまな管理タスクを実行するために JBoss Data Virtualization インスタンスに接続できます。

データアクセス

AdminShell を使用すると、仮想データベース (VDB) に接続し、SQL コマンドを実行して VDB データを問い合わせ、結果を参照できます。

移行

AdminShell を使用すると、VDB および関連するコンポーネントをある開発環境から別の開発環境に移行するスクリプトを開発できます (移行スクリプトは、本番稼働デプロイメントで実行する前にテストおよび自動化できます)。

テスト版

組み込みの JUnit テストフレームワークを使用すると、ユーザーはシステムの状態とデータの整合性を確認する回帰テストを作成できます。作成されたテストは、システム機能をアトミックに検証します。QA 担当者が手動で検証する必要はありません。

2.4.2. AdminShell スクリプト

Groovy 言語を使用して、タスクを自動化する AdminShell スクリプトを記述します。AdminShell は、JBoss Data Virtualization の管理を支援する Groovy 機能のカスタムライブラリーを提供します。これらのカスタム機能は、他の Groovy コードとは無関係に、または他の Groovy コードとともに使用できます。

基本的なルールは以下のとおりです。これらのルールを覚えて構文に慣れてください。

- すべてのコマンドと関数では大文字と小文字を区別します。
- Groovy コマンドは、セミコロンで終了する必要はありません。これはオプションです。
- 関数は入力としてゼロ個以上のパラメーターをとります。
  - 関数パラメーターはかっこ内で提供されます。
  - 文字列パラメーターは、一重引用符または二重引用符内に提供する必要があります。

```
connectAsAdmin("localhost", "9999", "user", "password", "conn1")
```

- 他の Java クラスおよびメソッドは、必要なライブラリーがすでにクラスパスで指定されている限り、スクリプト内からアクセスしたり、呼び出したりできます (**lib** ディレクトリーに追加された JAR ファイルは自動的にクラスパスに含まれます)。

```
import my.package.*;
myObject = new MyClass();
myObject.doSomething();
```

注記

disconnect() コマンドを入力して、AdminShell を終了する前に JBoss Data Virtualization から切断する必要があります。

参照

- Groovy スクリプトの詳細については、http://groovy.codehaus.org/ を参照してください。
- Groovy スクリプトと SQL の詳細については、http://groovy.codehaus.org/Database+features を参照してください。
- Groovy スクリプトを使用したテストの詳細については、http://groovy.codehaus.org/Unit+Testing と http://junit.org/ を参照してください。

2.4.3. AdminShell ヘルプ

AdminShell で利用可能なすべてのカスタム管理メソッドは、**adminHelp()** メソッドを使用してリストできます (カスタム AdminShell メソッドのみが表示されることに注意)。

```
adminHelp()
```

特定のメソッドの定義と入力パラメーターが必要な場合は、**adminHelp("*METHOD*")** を使用します。

```
adminHelp("deploy")

/*
*Deploy a VDB from file
*/
void deploy(
String /* file name */)
throws AdminException
throws FileNotFoundException
```

すべての SQL 拡張メソッドをリストする場合は、**sqlHelp()** メソッドを使用します。

```
sqlHelp()
```

特定のメソッドの定義と入力パラメーターを取得する場合は、**sqlHelp("*METHOD*")** を使用します。

2.4.4. AdminShell 基本コマンド

表2.1 AdminShell 基本コマンド

コマンド	説明
adminHelp();	利用可能なすべてのカスタム AdminShell メソッドを表示します。
adminHelp("*METHOD*");	提供された AdminShell メソッドの定義および入力パラメーターを表示します。
sqlHelp();	利用可能なすべてのカスタム SQL AdminShell メソッドを表示します。
sqlHelp("*METHOD*");	提供された SQL AdminShell メソッドの定義および入力パラメーターを表示します。
println "*STRING*";	値をコンソールに出力します。
sql = connect();	**connection.properties** ファイルで指定されたデフォルトの接続を使用して、拡張された Groovy SQL 接続を取得します。
sql.execute("*SQL*");	SQL コマンドを実行します。
connectAsAdmin();	**connection.properties** ファイルで指定されたデフォルトの接続を使用して管理ユーザーで接続詞ます。このコマンドは VDB 名を必要としません。これは管理接続であり JDBC 接続ではないため、この接続を使用して SQL コマンドを発行できません。SSL が使用される場合は、接続 URL とクライアント SSL 設定を必要に応じて調整する必要があります (通常は、2 方向 SSL の場合のみ)。
println getConnectionName();	アクティブな接続の名前を返します。
useConnection("*cNAME*");	該当する接続を使用します。
disconnect();	アクティブな接続を切断します。
disconnectAll();	すべての接続 (アクティブな接続および中断された接続) を切断します。

2.4.5. 非インタラクティブ AdminShell でのスクリプトの実行

手順2.3 非インタラクティブ AdminShell でのスクリプトの実行

1. **コマンドラインターミナルを開く**
2. **スクリプトを実行する**

   /adminshell.sh load [-D*param*=*value*] *PATH/FILENAME* コマンドを実行します。

注記

-D を使用して渡すオプションプロパティーには、**System.getProperty()** を使用してスクリプトファイル内からアクセスできます。

```
value = System.getProperty("param")
```

重要

スクリプト内で確立されたすべての接続は、スクリプトが修了する前に切断する必要があります。

2.4.6. インタラクティブ AdminShell の起動

手順2.4 インタラクティブ AdminShell の起動

1. **adminshell ディレクトリーに移動する**

   a. コマンドラインターミナルを開きます。

   b. AdminShell ディレクトリーへ移動します: ***/EAP_HOME*/dataVirtualization/teiid-adminshell/**.

2. AdminShell を展開します: **unzip teiid-VERSION-adminshell-dist.zip**

3. **adminshell スクリプトを起動する**

   ./adminshell.sh コマンドを実行します。

   ログの出力は以下のようになります。

```
================================================================================

  Teiid AdminShell Bootstrap Environment

  TEIID_HOME  = /EAP_HOME/dataVirtualization/teiid-adminshell-VERSION
  CLASSPATH   = /EAP_HOME/dataVirtualization/teiid-
adminshell-VERSION/lib/patches/*:/EAP_HOME/dataVirtualization/teiid-
adminshell-VERSION/lib/teiid-
adminshell-VERSION.jar:/EAP_HOME/dataVirtualization/teiid-
adminshell-VERSION/lib/*
  JAVA        = /usr/lib/jvm/java-1.7.0-openjdk.x86_64/bin/java

================================================================================

===> [import static org.teiid.adminshell.AdminShell.*; import static
org.teiid.adminshell.GroovySqlExtensions.*; import
org.teiid.adminapi.*;]
Groovy Shell (1.7.2, JVM: 1.7.0_25)
Type 'help' or '\h' for help.
-------------------------------------------------------------------------
-------------
groovy:000>
```

結果:

インタラクティブ AdminShell が実行されています。この時点で、個々のコマンドを行ごとに実行できます。

注記

exit コマンドを入力することにより、いつでもインタラクティブ AdminShell を終了できます。

2.4.7. インタラクティブ AdminShell に役に立つヒント

表2.2 インタラクティブ AdminShell コマンド

コマンド	説明
↑ ('up arrow')	実行した過去のコマンドを最新のものから表示します。
help	インタラクティブシェア内で使用するために特別に提供された追加コマンドを表示します。

シェルの動作

インタラクティブシェルは特別なインタープリターを使用し、Groovy スクリプトを実行して期待される動作とは異なる動作を表示します。

- def ステートメントは、シェルのコンテキストで変数を定義しません。たとえば、**`def x = 1`**, use **`x = 1`** は使用しないでください。
- シェルは、アノテーションを使用する Groovy クラスを解析できません。

2.4.8. インタラクティブ AdminShell でのスクリプトの保存

手順2.5 インタラクティブ AdminShell でのスクリプトの保存

1. **インタラクティブ AdminShell を開く**
    a. コマンドラインターミナルを開きます。
    b. ***`/EAP_HOME`*`/dataVirtualization/teiid-adminshell-`*`VERSION/`** に移動します。
    c. **`./adminshell.sh`** コマンドを実行します。
2. **ファイルへの記録を開始する**
    `record start` ***`PATH/FILENAME`*** コマンドを入力します。
3. **記録する必要なコマンドを入力する**
    入力/出力を記録する一連のコマンドを入力します。
4. **ファイルへの記録を停止する**
    `record stop` コマンドを入力します。

結果:

`record start` コマンドを発行し、**`record stop`** コマンドを発行するまでのすべての入力と出力が *PATH/FILENAME* にキャプチャされます。

注記

入力と出力の両方が記録されるため、ファイルはスクリプトファイルとして実行する前に編集する必要があります。

2.4.9. インタラクティブ AdminShell でのスクリプトの実行

手順2.6 インタラクティブ AdminShell でのスクリプトの実行

1. **インタラクティブ AdminShell を開く**
    a. コマンドラインターミナルを開きます。
    b. **`/EAP_HOME/dataVirtualization/teiid-adminshell-VERSION/`** に移動します。
    c. **`./adminshell.sh`** コマンドを実行します。
2. **スクリプトを実行する**
    A. **インタラクティブシェル `load` コマンドを使用してスクリプトを実行する**
    `load PATH/FILENAME` コマンドを実行します。
    B. **Groovy `evaluate` コマンドを使用してスクリプトを実行する**
    `evaluate("PATH/FILENAME" as File)` コマンドを実行します。

注記

また、スクリプトは、インタラクティブシェルを開いた後に行ごとに実行できます。

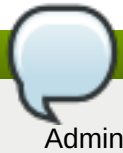
2.4.10. AdminShell GUI の起動

手順2.7 AdminShell GUI の起動

1. **`adminshell` ディレクトリーに移動する**
    a. コマンドラインターミナルを開きます。
    b. **`/EAP_HOME/dataVirtualization/teiid-adminshell-VERSION/`** に移動します。
2. **`adminshell-console` スクリプトを起動する**
    `./adminshell-console.sh` コマンドを実行します。

結果:

AdminShell GUI が表示されます。



注記

AdminShell GUI は、**File** → **Exit** をクリックしていつでも終了できます。

2.4.11. AdminShell GUI でのスクリプトの保存

手順2.8 AdminShell GUI でのスクリプトの保存

1. **`adminshell` ディレクトリーに移動する**

    a. コマンドラインターミナルを開きます。

    b. **`/EAP_HOME/dataVirtualization/teiid-adminshell-VERSION/`** に移動します。

2. **スクリプトを入力する**

    AdminShell GUI の上部の*スクリプトウィンドウ*にスクリプトを入力します。スクリプトは、作業中に **Script** → **Run** で実行してテストできます。

3. **スクリプトを保存する**

    a. **File** → **Save As** を選択します。

    b. スクリプトの場所とファイル名を選択し、**Save** をクリックします。

2.4.12. AdminShell GUI でのスクリプトの実行

手順2.9 AdminShell GUI でのスクリプトの実行

1. **AdminShell GUI を開く**

    a. コマンドラインターミナルを開きます。

    b. **`/EAP_HOME/dataVirtualization/teiid-adminshell-VERSION/`** に移動します。

    c. **`./adminshell-console.sh`** コマンドを実行します。

2. **スクリプトを準備する**

    A. **新しいスクリプトを入力する**

    AdminShell GUI の上部の*スクリプトウィンドウ*にスクリプトを入力します。

    B. **保存されたスクリプトをロードする**

        a. **File** → **Open** を選択します。

        b. 必要なスクリプトファイルを見つけ、**Open** をクリックします。

        スクリプトが、AdminShell GUI の上部の*スクリプトウィンドウ*に表示されます。

3. **スクリプトを実行する**

    Script → **Run** を選択します。

2.4.13. AdminShell 接続プロパティー

`EAP_HOME/dataVirtualization/teiid-adminshell-VERSION/connection.properties` ファイルは、Boss Data Virtualization インスタンスに接続するために AdminShell が使用するデフォルトの接続プロパティーを提供します。

```
jdbc.user=user
jdbc.password=user
jdbc.url=jdbc:teiid:admin@mm://localhost:31000;
```

```
admin.host=localhost
admin.port=9999
admin.user=admin
admin.password=admin
```

入力パラメーターなしで **connect()** または **connectionAsAdmin()** を呼び出すと、このプロパティーファイルで定義された設定を使用して JBoss Data Virtualization に接続します。**connect()** は、**jdbc** の接頭辞を持つプロパティーを使用し、**connectAsAdmin()** は **admin** の接頭辞を持つプロパティーを使用します。また、好きなプロパティーを使用して接続するために **connect()** メソッドまたは **connectAsAdmin()** メソッドにパラメーターを含めることもできます。

```
connect("URL", "USER", "PASSWORD")
```

警告

パスワードはクリアテキストで保存しないでください。上記の例はデモを目的としています。

ファイルにパスワードを保存する場合は、パスワードを安全にするのに必要な対策をとってください。必要な対策がとれない場合は、この機能を使用せずに、パスワードを対話的 (または他の安全な方法) に入力してください。

2.4.14. AdminShell での複数の接続

AdminShell を使用して、ユーザーは 1 つまたは複数の JBoss Data Virtualization インスタンスに対する複数の接続を管理できます。たとえば、ユーザーは開発サーバーに対する接続と統合サーバーに対する別の接続を同時に行えます。

新しい接続が行われるたびに、接続に一意な名前が割り当てられ、アクティブな接続になります。すでに接続がある場合は、一時停止されます (閉じられません)。

getConnectionName() メソッドは、アクティブな接続の名前を返します。この接続名は、次のコマンドを使用して変数 *cName* に割り当てることができます。

```
cName = getConnectionName();
```

現在の接続の名前は、接続を変更するために必要です。アクティブな接続を変更するには、使用する接続の名前 (または名前が割り当てられた変数) を指定して **useConnection()** コマンドを使用します。

```
useConnection(cName);
```

例

以下の例は、2 つの接続を使用して切り替える方法を示しています。

```
// 接続を作成する
connectAsAdmin();

//接続名を conn1 に割り当てる
conn1 = getConnectionName();
```

```
deploy("file.vdb")

// 2 番目の接続 (この時点ではアクティブな接続) を作成する
connectAsAdmin();

//新しい接続名を conn2 に割り当てる
conn2 = getConnectionName();

deploy("file.vdb")

// アクティブな接続を conn1 に切り替える
useConnection(conn1);

// アクティブな接続 (conn1) を閉じる
disconnect();
```

2.4.15. サンプルスクリプト

例2.1 VDB のデプロイ

```
connectAsAdmin();
deploy("/path/to/<name>.vdb");

// デプロイメントを検証する
VDB vdb = getVDB("<name>", 1);
if (vdb != null){
     print (vdb.getName()+"."+vdb.getVersion()+" is deployed";
}
else {
     print ("<name>.vdb failed to deploy";
}
```

例2.2 データソースの作成 (Oracle)

```
connectAsAdmin();

// 最初に Oracle 用 JDBC jar ファイルをデプロイする
deploy("ojdbc6.jar");

props = new Properties();
props.setProperty("connection-url","jdbc:oracle:thin:@<host>:1521:<sid>");
props.setProperty("user-name", "scott");
props.setProperty("password", "tiger");

createDataSource("oracleDS", "ojdbc6.jar", props);
```

例2.3 Teiid に対する SQL クエリの実行

```
sql = connect("jdbc:teiid:<vdb>@mm://<host>:31000", "user", "user");

// 選択する
```

```
sql.eachRow("select * from sys.tables") { println "${it}" }

// 更新、挿入、削除する
sql.execute(<sql command>);
```

2.4.16. AdminShell FAQ

問： なぜ adminhelp コマンドは GUI ツールで動作しないのですか?

答： AdminShell GUI 環境では、Groovy によって直接サポートされない **load**、**help**、**adminhelp** などのシェルコマンドが認識されません。GUI では、同等の機能形式を使用する必要があります (たとえば、**adminhelp** の代わりに **adminHelp()** を使用します)。

問： 事前に作成されたスクリプトを利用できますか?

答： 現時点では提供されていません。

問： connectAsAdmin() と connect() の違いは何ですか?

答： **connectAsAdmin()** メソッドは、JBoss Data Virtualization の AdminAPI に対するコンテキスト接続を作成します。**connect()** メソッドは、JBoss Data Virtualization に対する SQL 呼び出しに使用する Groovy SQL オブジェクトの拡張を返します。

問： getAdmin() は何を行うのですか? なぜこれが必要なのですか?

答： **getAdmin()** メソッドは、**connectAsAdmin()** で接続したときに作成された現在のコンテキスト接続オブジェクトを返します。このオブジェクトは、インターフェース org.teiid.adminapi.Admin を実装します。提供された AdminShell コマンドは、この API のラッパーです。上級ユーザーは、提供されたラッパーコマンドがニーズを満たさない場合にこの API を直接使用できます。

問： スクリプトを記述するために IDE サポートを利用できますか?

答： AdminShell GUI ツールは軽量な IDE です。Groovy には完全な IDE サポートが利用可能ですが、クラスパスとスクリプトインポートを手動で処理する必要があります。

問： 他の環境で AdminShell メソッドを使用できますか?

答： AdminShell メソッド (名前付きコンテキスト接続、AdminAPI ラッパー、およびヘルプシステムを含む) は、Groovy に対する直接的な依存関係を持たないため、他のスクリプト言語で使用できます。

別の言語で AdminShell メソッドを使用するには、静的なメソッドと Admin クラスをスクリプトにインポートします。また、***EAP_HOME*/dataVirtualization/teiid-adminshell-*VERSION*/lib/teiid-*VERSION*.jar** と ***EAP_HOME*/dataVirtualization/teiid-adminshell-*VERSION*/lib/teiid-adminshell-*VERSION*.jar** がクラスパスで指定されていることを確認する必要があります。

以下のコード例は、Java、BeanShell、Groovy などで動作するインポートステートメントを示しています。

```
import static org.teiid.adminshell.AdminShell.*;
import org.teiid.adminapi.*;
```

問： **デバッグサポートは利用できますか?**

答： インタラクティブ AdminShell と GUI には現在の状態を検査するサポートが組み込まれています。ただし、ラインベースデバッグは本書の範囲外です。

2.5. JBoss Operations Network

2.5.1. JBoss エージェントプラグインパックのインストール

JBoss Operations Network には、他の JBoss 製品を管理する追加のエージェントプラグインがあります。これらは JBoss ON リソースプラグインですが、別のパッケージに含まれ、ダウンロードする別のサブスクリプションを必要とします。

1. カスタマーサポートポータルからプラグイン JAR ファイルをダウンロードします。
    a. https://access.redhat.com/jbossnetwork/ にナビゲートします。
    b. *PRODUCT* に対して JBoss ON を選択します。
    c. **Download** を選択します。
2. プラグイン JAR ファイルを ***JON_HOME*/plugins** ディレクトリーに抽出します。
3. JBoss ON サーバーでプラグインを更新します。これは、JBoss ON GUI を使用するか、サーバーを再起動することにより実行できます。
4. GUI からプラグインをロードするには、以下の手順に従ってください。
    a. **Administration** タブを開きます。
    b. 左側の **Configuration** 領域で、**Agent Plug-ins** リンクを選択します。
    c. ロードされたエージェントプラグインのリストの下部で、**SCAN FOR UPDATES** ボタンをクリックします。
5. エージェントでプラグインをリロードして新しいプラグインをロードします。これは、エージェントのコマンドプロンプトで以下のプラグインコマンドを使用して実行できます。

```
> plugins update
```

注記

これは、エージェントまたはエージェントのグループに対してプラグインの更新操作をスケジュールすることによっても実行できます。別の方法として、エージェントを再起動します。

2.6. Dashboard Builder

2.6.1. データソース

Red Hat JBoss Dashboard Builder は、コンテナーの JNDI または JDBC ドライバーを直接使用して外部データベースに接続できます。データベースへの接続は、ページ **External Connections** のワークスペース

Showcase で設定できます。データベースへの接続を確立したら、データベースからデータを収集し、ページのダッシュボード領域でデータをインジケーターとして視覚化することを可能にするデータプロバイダーを作成する必要があります。

CSV ファイルに接続してデータを取得する場合は、データプロバイダーから直接接続が確立されます。

Red Hat JBoss Dashboard Builder は、ローカルデータを格納するために独自のローカル内部データベースを使用します。このデータベースは、Dashboard Builder に対して読み取り専用ですが、外部からアクセスできます。

2.6.2. データソースへの接続

JNDI データソース (アプリケーションコンテナーからセットアップされ、アクセス可能なデータソース) またはカスタムデータソースとしてのデータソースに直接接続できます (アプリケーションコンテナーに正しい JDBC ドライバーがデプロイされている場合)。

外部データソースに接続するには、以下の手順に従ってください。

1. データソースが稼働しており、アプリケーションサーバーがデータソースにアクセスできることを確認してください (ドライバーやログインクレデンシャルなどを確認します。Red Hat JBoss EAP 6 でこれを行うには、管理コンソールで **Subsystems** → **Connector** → **Datasources** を選択します)。
2. Dashboard Builder のツリーメニュー (デフォルトでは、Showcaseパースペクティブにあります) で、**Administration** → **External connections** に移動します。
3. On the displayed External Connection panel, click the **`New DataSource`** 

button.
4. データソースタイプ (JNDI またはカスタムデータソース) を選択し、以下の各データソースパラメーターを提供します。

2.6.3. データプロバイダーの作成

新しいデータプロバイダーを作成するには、以下を実行します。

1. ツリーメニュー (Showcase ワークスペースのラテラルメニューのパネル) で、**Administration** → **Data providers** をクリックします。
2. In the **`Data Providers`** panel, click the **`Create new data provider`** 

button.
3. 更新された **`Data Providers`** パネルの **`Type`** ドロップダウンメニューで、データプロバイダーが処理するソースに応じてデータプロバイダーのタイプを選択します。
4. データプロバイダーパラメーターを定義します。
   CSV ファイルを介したデータプロバイダー
   - 名前: ユーザーフレンドリーな名前とロケール
   - CSV ファイル URL: ファイル url (たとえば、**`file:///home/me/example.csv`**)
   - データセパレーター: CSV ファイルでセパレーターとして使用される記号 (デフォルト値はセミコロンです。カンマをセパレーター記号として使用する場合は、必要に応じて数字形式を変更してください)。

- 引用符: 引用符に使用される記号 (デフォルト値は二重引用符です。記号はロケールに応じて異なることがあります)。
- エスケープ記号: 後続の記号をエスケープしてそのリテラル値を保持するために使用される記号
- 日付形式: 日付と時刻の形式
- 数値形式: 数千および小数に解決される数値の形式

データベースを介したデータプロバイダー (SQL クエリー)

- 名前: ユーザーフレンドリーな名前とロケール
- データソース: クエリーを実行するデータソース (デフォルト値は **`local`** です。Dashboard Builder データソースに対してクエリーを実行できます)
- クエリー: 必要なデータを返すクエリー

5. Click **`Attempt data load`** 

to verify the parameters are correct.
6. **`Save`** をクリックします。
7. 検出されたデータがあるテーブルで、データ型を定義し、必要に応じて、データにユーザーフレンドリーな名前を指定します。**`Save`** をクリックします。

データプロバイダーは、この時点で任意のページのインジケーターで視覚化できます。

2.6.4. Dashboard Builder ワークスペース

2.6.4.1. ワークスペースの作成

新しいワークスペースを作成するには、以下の手順に従ってください。

1. 上部メニューの **`Create workspace`** ボタンをクリックします。

   `Workspace` ノードが展開され、右側にワークスペース詳細が示されたワークスペース管理領域がある管理コンソールが表示されます。
2. 右側の **`Create workspace`** テーブルで、ワークスペースパラメーターを設定します。
   - 名前: ワークスペース名とロケール
   - タイトル: ワークスペースとロケール
   - スキン: ワークスペースリソースに提供するスキン
   - エンベロープ: ワークスペースリソースに適用するエンベロープ
3. **`Create workspace`** をクリックします。
4. オプションで、左側のツリーメニューでワークスペース名をクリックし、右側にワークスペースプロパティーがある領域で、追加のワークスペースパラメーターを定義できます。
   - URL: ワークスペース URL
   - ユーザーホーム検索: ホームページ設定

Role assigned page に設定すると、ページパーミッション設定されたホームページが適用されます。そのため、ロールごとに異なるホームページを設定できます (**Current page** に設定するとすべてのユーザーが現在のページをホームページとして使用します)。

2.6.4.2. ページの作成

新しいページを作成するには、以下の手順に従ってください。

1. 正しいワークスペースを表示します。
2. Next to the **Page** dropdown box 

in the top menu, click the **Create new page**  button .
3. **Pages** ノードが展開され、右側にページ詳細が示されたページ管理領域がある管理コンソールが表示されます。
4. 右側の **Create new page** テーブルで、ページパラメーターを設定します。
   - 名前: ページ名とロケール
   - 親ページ: 新しいページの親ページ
   - スキン: ページに適用するスキン
   - エンベロープ: ページに適用するエンベロープ
   - ページレイアウト: ページのレイアウト
5. **Create new page** をクリックします。
6. オプションで、左側のツリーメニューでページ名をクリックし、右側にワークスペースプロパティーがある領域で、追加のページパラメーターを定義できます。
   - URL: ページ URL
   - 表示可能なページ: ページの表示可能性
   - リージョンとパネル間のスペース

2.6.4.3. ページパーミッションの定義

ユーザーは通常基礎となるアプリケーションコンテナー (Red Hat JBoss EAP の場合はデフォルトで **other** セキュリティードメインになります) に対してセットアップされた承認方法を使用して承認されますが、Red Hat JBoss Dashboard Builder には、個別ページまたは複数ページでパーミッション管理を実現する独自のロールベースアクセス制御 (RBAC) 管理ツールが含まれます。

ロールのページまたはすべてのワークスペースページでパーミッションを定義するには、以下の手順に従ってください。

1. On the top menu, click the **General configuration**  button : the management console is displayed.
2. 左側の **Workspace** ノードで、ページまたは **Pages** ノードを見つけます。
3. ページ/ページノードで、**Page permissions** ノードをクリックします。

4. 右側の **Page permissions** 領域で、以前に定義されたパーミッション定義 (該当する場合) を削除し、必要なロールの権利を定義します。

    a. **Permission assignation** テーブルで、**Select role** ドロップダウンメニューを探し、それぞれのロールを選択します。

    b. テーブルの **Actions** 列で、各パーミッションを無効または有効にします。

5. **Save** をクリックします。

2.6.4.4. パネル

パネルは GUI ウィジェットであり、ページに配置できます。主に以下の 3 つの種類のパネルがあります。

ダッシュボードパネル

主な BAM パネルであり、以下のものを含みます。

- データプロバイダーマネージャー: 利用可能なデータプロバイダーとデータプロバイダー管理オプションのリストがあるパネル
- フィルターおよびドリルダウン: データプロバイダーで定義されたページのインジケーターでフィルタリングを実現するためにすべての KPI とその値が表示されたパネル
- HTML エディターパネル: 静的なコンテンツがあるパネル
- 重要業績評価指数 (インジケーター): データプロバイダーのデータを視覚化するパネル

ナビゲーションパネル

ナビゲーション機能を提供し、以下のものを含むパネル

- Breadcrumb: 現在のページを参照する完全なページ階層があるパネル
- 言語メニュー: 利用可能なロケールがあるパネル (デフォルトでは中央上部)
- ログアウトパネル: 現在ログインしているユーザーの名前とログアウトボタンがあるパネル
- ページメニューカスタム: ワークスペース内のすべてのページに対するリンクが縦に配置され、ページの HTML ソースの一般的なコントロールがあるパネル
- 縦のページメニュー: ワークスペース内のすべてのページに対するリンクが縦に配置されたパネル (ページのリストは調整できます)
- 横のページメニュー: ワークスペース内のすべてのページに対するリンクが横に配置されたパネル (ページのリストは調整できます)
- ツリーメニュー: 管理やホーム (ラテラルメニューで、左側に表示される Showcase ワークスペースのホームページ) などの重要な機能に対するリンクがあるパネル
- カスタムワークスペースメニュー: 利用可能なワークスペースに対するリンク (ワークスペースのリストは調整できます) とワークスペースの HTML ソースに対する一般的なコントロールがあるパネル
- 横のワークスペースメニュー: 利用可能なワークスペースに対するリンク (ワークスペースのリストは調整できます) がある横方向のパネル
- 縦のワークスペースメニュー: 利用可能なワークスペースに対するリンク (ワークスペースのリストは調整できます) がある縦方向のパネル

システムパネル

システム設定と管理機能に対するアクセスを提供し、以下のものを含むパネル。

- データソースマネージャー: 外部データソース管理用のパネル
- エクスポートダッシュボード: ダッシュボードのパネルエクスポート
- ワークスペースのエクスポート/インポート: ワークスペースをエクスポートおよびインポートするパネル

2.6.4.5. パネルの追加

既存のパネルを追加するか、新しいパネルを作成するには、以下の手順に従ってください。

1. 各ページが開いてることを確認します (上部メニューの **Page** ドロップダウンメニューで、ページを選択します)。
2. In the top menu, click the **Create a new panel in current page**  button.
3. 表示されたダイアログボックスで、追加するパネルタイプを展開し (**Dashboard**、**Navigation**、または **System**)、追加するパネルをクリックします。
4. 左側の **Components** メニューから、既存のパネルインスタンスの名前または **Create panel** アイテムをページの必要な場所にドラッグアンドドロップします。

   新しいインジケーターを挿入する場合に、グラフ設定があるパネルビューが表示されます。グラフの詳細を定義し、ダイアログを閉じます。

   既存のインジケーターのインスタンスを追加する場合は、そのインジケーターが特定の元のページの KPI にリンクされているときにインジケーターを使用できないことがあります。このような場合は、新しいパネルを作成してください。
5. 必要に応じて、新しく追加されたパネルの内容を編集します。

2.6.5. Dashboard Builder のフィルター

フィルターは以下のように機能します。

- 複数のデータセットプロバイダー全体で「共有」プロパティーを定義できます。共有プロパティーとは、2 つ以上の異なるデータプロバイダーで同一 ID を持つプロパティーのことです。
- 共有プロパティーが含まれる 2 つ以上のデータプロバイダーを参照するダッシュボードを構築し、共有プロパティーでフィルターする場合、このプロパティーが含まれるすべての KPI がフィルターされます。
- 共有プロパティーは、「join」と似たフィルター動作を実装する場合に便利です。これにより、異なるデータプロバイダーに属する複数の KPI を同時にフィルターできます。
- join と似た動作を無効にするには、 **Data Provider Column Definition** 画面でプロパティー ID を調整します (プロパティー ID が一意になるようにし、データプロバイダー定義同士が競合しないようにしてください)。

パート III. ユーザー管理

第3章 ユーザーアカウント

3.1. ユーザーアカウント

JBoss Data Virtualization では、以下のアカウントタイプを使用できます。

JBoss EAP 管理ユーザー

管理コンソール、管理 CLI、JBoss Developer Studio、および管理 API で JBoss Data Virtualization インストールを管理するには、JBoss EAP 管理ユーザーが必要です。

JBoss Data Virtualization ユーザー

JBoss Data Virtualization ユーザーは、仮想データベース (VDB) にアクセスできます。これらのデータソースで実行できる操作を制御するユーザーのパーミッションを定義できます。

階層データベースユーザー

階層データベースユーザーは、提供された階層データベースにアクセスできます。このデータベースで実行できる操作を制御する各ユーザーのパーミッションを定義できます。

注記

ユーザー/ロールの名前と詳細は、オペレーティングシステムアカウントなどの他のアカウントとは独立しています。これらは JBoss EAP and JBoss Data Virtualization にのみ関連します。

3.2. データロール

認証されたユーザーはすべて VDB へアクセスできます。アクセスを制限するには、データロールを設定します。データロールは、Teiid Designer または動的 VDB の META-INF/vdb.xml ファイルで設定します。

データロール定義の一部として、データロールを **<mapped-role-name>** タグで指定された JAAS ロールにマップできます。JAAS ロールは web コンソールを使用して設定できます。

これらの JAAS ロールがユーザーにどのように関連付けられるかは、使用する特定の JAAS ログインモジュールによって異なります。たとえば、デフォルトの **UsersRolesLoginModule** では、プレーンテキストファイルでユーザーが JAAS ロールに関連付けられます。

データロールの詳細については、『Red Hat JBoss Data Virtualization Development Guide: Reference Material』 を参照してください。

重要

"admin" または "user" は JAAS ロール名として使用しないでください。これらは、Dashboard Builder パーミッション用に特別に予約されています。

3.3. JBoss EAP 管理ユーザーの追加

3.3.1. 管理インターフェースのユーザーの追加

以下の手順では、選択したインストール方法によって最初の管理ユーザーが作成されなかった場合に最初の管理ユーザーを作成する方法を説明します。最初の管理ユーザーは、Web ベースの管理コンソールおよび管理 CLI のリモートインスタンスを使用してリモートシステムから JBoss EAP 6 を設定および管理できます。

手順3.1 リモート管理インターフェース用の最初の管理ユーザーを作成

1. **`add-user.sh` または `add-user.bat` スクリプトを実行します。**

   ***`EAP_HOME`*`/bin/`** ディレクトリーへ移動します。ご使用のオペレーティングシステムに対応するスクリプトを呼び出します。

   Red Hat Enterprise Linux

   ```
   [user@host bin]$ ./add-user.sh
   ```

   Microsoft Windows Server

   ```
   C:\bin>  add-user.bat
   ```

2. **管理ユーザーの追加を選択します。**

   `ENTER` を押して、デフォルトのオプション **`a`** を選択して管理ユーザーを追加します。

   このユーザーは **`ManagementRealm`** に追加され、Web ベース管理コンソールまたはコマンドラインベース管理 CLI を使用して監理操作を実行することを許可されます。他のオプション **`b`** を選択すると、ユーザーが **`ApplicationRealm`** に追加され、特定のパーミッションは提供されません。このレルムはアプリケーションで使用するために提供されます。

3. **ユーザー名とパスワードを入力します。**

   ユーザー名とパスワードの入力を要求されたら入力します。入力後、パスワードを確認するよう指示されます。

4. **グループ情報を入力します。**

   ユーザーが属するグループを追加します。ユーザーが複数のグループに属する場合は、カンマ区切りリストを入力します。ユーザーがどのグループにも属さない場合は空白のままにします。

5. **情報を確認します。**

   情報を確認するよう指示されます。情報が正しければ **`yes`** を入力します。

6. **ユーザーがリモート JBoss EAP 6 サーバーインスタンスを表すかどうかを選択します。**

   管理者以外にも、場合によっては JBoss EAP 6 の別のインスタンスを表すユーザーを **`ManagementRealm`** で JBoss EAP 6 に追加する必要があることがあります。このユーザーは、メンバーとしてクラスターに参加することを認証できる必要があります。次のプロンプトでは、この目的のために追加されたユーザーを指定できます。**`yes`** を選択した場合は、ユーザーのパスワードを表すハッシュされた **`secret`** 値が提供されます。これは他の設定ファイルに追加する必要があります。このタスクでは、**`no`** を選択してください。

7. **追加ユーザーを入力します。**

必要な場合は、この手順を繰り返すと追加のユーザーを入力できます。また、稼働中のシステムにいつでもユーザーを追加することが可能です。デフォルトのセキュリティーレルムを選択する代わりに他のレルムにユーザーを追加すると、承認を細かく調整できます。

8. **非対話的にユーザーを作成します。**

   コマンドラインで各パラメーターを渡すと非対話的にユーザーを作成できます。ログや履歴ファイルにパスワードが表示されるため、この方法は共有システムでは推奨されません。管理レルムを使用した、コマンドの構文は次のとおりです。

   ```
   [user@host bin]$ ./add-user.sh username password
   ```

   アプリケーションレルムを使用するには、**-a** パラメーターを使用します。

   ```
   [user@host bin]$ ./add-user.sh -a username password
   ```

9. **--silent** パラメーターを渡すと add-user スクリプトの通常の出力を無効にできます。これは、**username** および **password** パラメーターが指定されている場合のみ適用されます。エラーメッセージは表示されます。

結果:

追加したすべてのユーザーは、指定したセキュリティーレルム内でアクティベートされます。**ManagementRealm** レルム内でアクティブなユーザーは、リモートシステムから JBoss EAP 6 を管理できます。

3.3.2. デフォルトのユーザーセキュリティー設定

はじめに

JBoss EAP 6 のすべての管理インターフェースはデフォルトで保護されます。このセキュリティーには、2 つの形式があります。

- ローカルインターフェースは、ローカルクライアントとローカルクライアントが接続するサーバーとの間の SASL コントラクトによって保護されます。このセキュリティーメカニズムは、ローカルファイルシステムにアクセスするクライアントの機能に基づきます。ローカルシステムへアクセスできるとクライアントによるユーザーの追加が可能で、他のセキュリティーメカニズムを無効にするよう設定を変更できるからです。これにより、ファイルシステムへ物理的にアクセスできると、他のセキュリティーメカニズムが不要になるという原則が厳守されます。このメカニズムは 4 つの手順で実現されます。

注記

HTTP を使用してローカルホストへ接続する場合でも、HTTP のアクセスはリモートと見なされます。

  - ローカル SASL メカニズムを用いて認証する要求が含まれるメッセージをクライアントがサーバーに送信します。
  - サーバーはワンタイムトークンを生成し、固有のファイルに書き込み、ファイルのフルパスが含まれるメッセージをクライアントへ送信します。
  - クライアントはファイルよりトークンを読み取り、サーバーへ送信し、ファイルシステムへローカルアクセスできるかを検証します。

- サーバーはトークンを検証し、ファイルを削除します。

- ローカル HTTP クライアントを含むリモートクライアントはレルムベースのセキュリティーを使用します。管理インターフェースを使用して JBoss EAP 6 インスタンスをリモートで設定するパーミッションを持つデフォルトのレルムは **ManagementRealm** です。このレルム (またはユーザーが作成したレルム) にユーザーを追加できるスクリプトが提供されます。ユーザーの追加の詳細については、『JBoss EAP 6 管理および設定ガイド』の章「ユーザー管理」を参照してください。ユーザーごとに、ユーザー名とハッシュ化されたパスワードがファイルに保存されます。

  管理対象ドメイン

  ***EAP_HOME*/domain/configuration/mgmt-users.properties**

  スタンドアロンサーバー

  ***EAP_HOME*/standalone/configuration/mgmt-users.properties**

  mgmt-users.properties の内容はマスクされていますが、機密ファイルとして扱う必要があります。ファイルモードを、ファイル所有者による読み書きのみが許可される **600** に設定することが推奨されます。

3.3.3. JBoss Data Virtualization ユーザーの追加

JBoss Data Virtualization のデフォルト設定では、ハッシュ化されたファイルを使用してユーザーおよびユーザーロールが定義されます。このデフォルト設定の新しいユーザーを追加するには、以下の手順に従ってください。

1. bin ディレクトリーに移動します。
2. ./add-user.sh スクリプトを実行します。
3. オプション「b) Application User (application-users.properties)」を選択し、希望のユーザー名とパスワードを追加します。

重要

"admin" および "user" ロール名は、Dashboard Builder パーミッション用に特別に予約されています。OData へのアクセスには odata グループが必要になります。

3.4. Modeshape ユーザーの追加

3.4.1. Modeshape パブリッシングユーザーの作成

Modeshape パブリッシングを使用するには、少なくとも 1 人のユーザーが接続および読み書きロールまたは接続および管理ロールを持っている必要があります。

デフォルトでは、すべてのロールがすべての匿名ロールユーザーに割り当てられます。

パート IV. 製品の設定

第4章 仮想データベース

4.1. 仮想データベースおよび JBoss Data Virtualization

JBoss Data Virtualization は、仮想データベース (VDB) を使用してエンタープライズデータ統合ソリューションを提供します。

4.2. 仮想データベースデプロイメント

仮想データベース (VDB) は、クライアントアプリケーションによりアクセスする前にデプロイする必要があります。

VDB は次の手段でデプロイできます。

ファイルデプロイメント

サーバーが開発者のワークステーションでローカル実行されている場合、開発段階の迅速なデプロイメントにはファイルデプロイメントが推奨されます。

管理コンソール

Web ベースの管理コンソールによるデプロイメントは、VDB をリモートサーバーにデプロイする最も単純な方法として推奨されます。

管理コマンドラインインターフェース

EAP 管理コマンドラインインターフェース (CLI) を使用したデプロイメントも簡単なデプロイメントオプションです。

AdminShell

AdminShell によるデプロイメントは、環境でアーティファクトのデプロイメントを自動化する場合などの高度なデプロイメントに推奨されます。

管理 API

管理 API によるデプロイメントは、他のアプリケーション内で VDB をデプロイする場合などの高度なデプロイメントに推奨されます。

注記

また、VDB は Teiid Designer 内でデプロイできます。『Red Hat JBoss Data Virtualization User Guide』 を参照してください。

重要

VDB を削除すると、そのすべてのリソースが自動的にクリーンアップされます。ただし、既存のセッションは自動的に終了しません。

警告

VDB をデプロイする場合、VDB を同じ名前のもので上書きすると、VDB バージョン管理が使用されてない限り、古い VDB へのすべての接続が終了します。

本番稼働システムでは VDB バージョン管理を使用することが推奨されます。

警告

ローカルでは VDB アーティファクトの名前を自由に付けることができますが、デプロイされた VDB アーティファクトのランタイム名には zip ファイルの場合は *.vdb 拡張子を付け、xmil ファイルの場合は *-vdb.xml を付ける必要があります。ファイル名が適切でないと、Teiid サブシステムは処理方法を認識できないため、デプロイメントに失敗します。

4.3. ファイルデプロイメントによる VDB のデプロイ

前提条件

- Red Hat JBoss Data Virtualization をインストールする必要があります。

手順4.1 ファイルデプロイメントによる VDB のデプロイ

1. **VDB を deploy ディレクトリーにコピー**

   VDB ファイルを ***EAP_HOME*/standalone/deployments** ディレクトリーにコピーします。

2. **マーカーファイルの作成**

   拡張子が **.dodeploy** の同じ名前の空のマーカーファイルを同じディレクトリーに作成します。たとえば、VDB 名が **enterprise.vdb** である場合、マーカーファイル名は **enterprise.vdb.dodeploy** である必要があります。

注記

これはスタンドアロンモードにのみ適用されます。ドメインモードの場合は、他の利用可能なメソッドのいずれかを使用する必要があります。

4.4. 管理コンソールでの VDB のデプロイ

前提条件

- Red Hat JBoss Data Virtualization をインストールする必要があります。
- JBoss Enterprise Application Platform (EAP) サーバーが稼働している必要があります。
- JBoss EAP 管理ユーザーが登録されている必要があります。

手順4.2 管理コンソールでの VDB のデプロイ

1. **ウェブブラウザーでコンソールを起動する**

   ウェブブラウザーで、http://localhost:9990/console/ を開きます。

2. **コンソールに対して認証する**

   プロンプトが表示されたら、JBoss EAP の管理者ユーザー名とパスワードを入力します。

3. **`Deployments` パネルを開く**

   `Runtime` ビューで **Server** → **Manage Deployments** を選択します。

4. **仮想データベースを追加する**

   a. **`Add`** ボタンを選択します。

   b. **`Choose File`** を選択し、デプロイする VDB ファイルを選択します。

   c. **`Next`** を選択してデプロイメント名を確認し、**`Save`** を選択します。

   d. **`En/Disable`** を選択して VDB を有効にします。

4.5. CLI を使用して VDB をデプロイ

前提条件

- Red Hat JBoss Data Virtualization をインストールする必要があります。
- JBoss Enterprise Application Platform (EAP) サーバーが稼働している必要があります。

手順4.3 CLI を使用して VDB をデプロイ

1. **コマンドラインインターフェースを開く**

   ***`EAP_HOME`*`/bin/jboss-cli.sh`** コマンドを実行します。

2. **サーバーに接続する**

   `connect` コマンドを実行します。

3. **仮想データベースをデプロイする**

   スタンドアロンモードの場合は、**`deploy` *`PATH/DATABASE`*`.vdb`** を実行します。

   ドメインモードの場合は、**`deploy -all-server-groups` *`PATH/DATABASE`*`.vdb`** を実行します。

注記

ドメインモードでは、デプロイメントオプションとして特定の "server-group" または利用可能なすべてのサーバーグループを選択する必要があります。

4.6. AdminShell を使用して VDB をデプロイ

前提条件

- Red Hat JBoss Data Virtualization をインストールする必要があります。
- JBoss Enterprise Application Platform (EAP) サーバーが稼働している必要があります。

手順4.4 AdminShell を使用して VDB をデプロイ

1. **インタラクティブ AdminShell インターフェースを開く**

   `./adminshell.sh` コマンドを実行します。

2. **接続を開く**

   インタラクティブ AdminShell 内で、**`connectAsAdmin()`** コマンドを実行します。

3. **仮想データベースをデプロイする**

   `deploy("`*`PATH/DATABASE`*`.vdb")` コマンドを実行します。

4. **接続を閉じる**

   `disconnect()` コマンドを実行します。

5. **インタラクティブシェルを終了する**

   a. **`exit`** コマンドを入力してインタラクティブシェルを終了します。

注記

ドメインモードで AdminShell を使用してデプロイする場合に、VDB は利用可能なすべてのサーバーにデプロイされます。

また、VDB は AdminShell コンソールまたは非インタラクティブ AdminShell を介したスクリプトを使用してデプロイできます。これらの詳細については、管理シェルに関するトピックを参照してください。

4.7. 管理 API を使用して VDB をデプロイ

VDB は、管理 API パッケージ (**`org.teiid.adminapi`**) 内の Admin インターフェースにより提供された **`deploy`** メソッドを使用してデプロイできます。

Red Hat JBoss Data Virtualization 向け Javadocs は Red Hat カスタマーポータルで利用できます。

注記

ドメインモードで管理 API を使用してデプロイする場合に、VDB は利用可能なすべてのサーバーにデプロイされます。

4.8. VDB 依存関係

JBoss Data Virtualization で仮想データベース (VDB) をデプロイする場合は、VDB で使用する物理データソースにアクセスするために依存ライブラリーと設定も提供する必要があります (***EAP_HOME*/*MODE*/deployments/*DATABASE*.vdb** ファイル内の **META-INF/vdb.xml** を参照してすべての依存物理データソースを識別できます)。

たとえば、Oracle とファイルソースを VDB で統合する場合は、Oracle ソースの JDBC ドライバー、必要なすべてのドキュメント、ファイルトランスレーターで必要な設定ファイルを提供する必要があります。

データソースインスタンスは、複数の VDB とアプリケーションで共有できます。ヘビーウェイトでリソース制限があるソースに対する接続の共有を検討します。

VDB と依存関係をデプロイしたら、クライアントアプリケーションは **JDBC** API を使用して接続できます。デプロイメントでエラーが発生した場合は、接続の試行が失敗し、メッセージがログに記録されます。管理コンソールを使用して (またはログファイルをチェックして) エラーを特定し、修正します。JDBC を使用して VDB に接続する方法については、『Red Hat JBoss Data Virtualization Development Guide: Server Development』を参照してください。

警告

一部のデータソース設定ファイルには、パスワードまたは他の機密情報が含まれることがあります。パスワードをプレーンテキストで保存することを回避する手順については、JBoss Enterprise Application Platform 『セキュリティーガイド』を参照してください。

4.9. データソースデプロイメント

4.9.1. Accumulo データソース

Accumulo データソースは、インストール時に JBoss EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。Accumulo データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid:add(jndi-name=java:/accumulo-ds, class-
name=org.teiid.resource.adapter.accumulo.AccumuloManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid/config-
```

```
properties=ZooKeeperServerList:add(value=localhost:2181)
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid/config-properties=Username:add(value=user)
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid/config-properties=Password:add(value=password)
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid/config-properties=InstanceName:add(value=instancename)
/subsystem=resource-adapters/resource-adapter=accumulo/connection-
definitions=teiid/config-properties=Roles:add(value=public)
/subsystem=resource-adapters/resource-adapter=accumulo:activate
runbatch
```

以下に、RAR ファイルで定義されたプロパティーを示します。

表4.1 プロパティ

プロパティー	説明	必要性	デフォルト
ZooKeeperServerList	ズーキーパーサーバーの場所のカンマ区切りのリスト。各場所には、host:port という形式でオプションのポートを含めることができます。	必要	なし
ZooKeeperServerList	ズーキーパーサーバーの場所のカンマ区切りのリスト。各場所には、host:port という形式でオプションのポートを含めることができます。	必要	なし
Username	接続ユーザーの名前	必要	なし
Password	接続ユーザーのパスワード	必要	なし
InstanceName	Accumulo インスタンス名	必要	なし
Password	接続ユーザーのパスワード	必要	なし
Roles	ユーザーのオプションの可視性。カンマ区切りで複数を指定します。	不必要	なし

この Accumulo コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=accumulo)
```

4.9.2. Amazon SimpleDB データソース

SimpleDB データソースは、インストール時に EAP にデプロイされた Teiid 固有の JCA コネクターを使用します。SimpleDB データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しいアクセスキーを提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=simpledb/connection-
definitions=simpledbDS:add(jndi-name=java:/simpledbDS, class-
name=org.teiid.resource.adapter.simpledb.SimpleDBManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=simpledb/connection-
definitions=simpledbDS/config-properties=AccessKey:add(value=xxx)
/subsystem=resource-adapters/resource-adapter=simpledb/connection-
definitions=simpledbDS/config-properties=SecretAccessKey:add(value=xxx)
/subsystem=resource-adapters/resource-adapter=simpledb:activate
runbatch
```

この SimpleDB コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=simpledb)
```

4.9.3. Cassandra データソース

Cassandra データソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。Cassandra データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=cassandra/connection-
definitions=cassandraDS:add(jndi-name=java:/cassandraDS, class-
name=org.teiid.resource.adapter.cassandra.CassandraManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=cassandra/connection-
definitions=cassandraDS/config-properties=Address:add(value=127.0.0.1)
/subsystem=resource-adapters/resource-adapter=cassandra/connection-
definitions=cassandraDS/config-properties=Keyspace:add(value=my-keyspace)
/subsystem=resource-adapters/resource-adapter=cassandra:activate
runbatch
```

この Cassandra コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=cassandra)
```

4.9.4. ファイルデータソース

ファイルデータソースは、インストール時に EAP にデプロイされた Teiid 固有の JCA コネクターを使用します。ファイルデータソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなど

を使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しいディレクトリー名と他のプロパティーを提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=file/connection-
definitions=fileDS:add(jndi-name=java:/fileDS, class-
name=org.teiid.resource.adapter.file.FileManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=file/connection-
definitions=fileDS/config-
properties=Parentdirectory:add(value=/home/rareddy/testing/)
/subsystem=resource-adapters/resource-adapter=file/connection-
definitions=fileDS/config-properties=AllowParentPaths:add(value=true)
/subsystem=resource-adapters/resource-adapter=file:activate
runbatch
```

このファイルコネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=file)
```

4.9.5. Google スプレッドシートデータソース

Google アカウントは、ClientLogin (ログインとパスワードが必要) または OAuth (ユーザーが更新トークンを提供する必要があるため、より複雑) のいずれかの方法で認証できます。

Google JCA コネクターには teiid-connector-google.rar という名前が付けられます。この例には、google.xml サンプルファイルが含まれます。JCA コネクターには認証を実現する複数の設定プロパティーがあります。JCA コネクターは 1 つのスプレッドシートに接続します。

表4.2 設定プロパティー

プロパティー	説明
AuthMethod	Google へのアクセス方法。このプロパティーは OAuth2 に設定することが可能で、その場合は RefreshToken を提供する必要があります。
SpreadsheetName	このコネクターのデータソースであるスプレッドシートの名前を持つ必須プロパティー。
BatchSize	一度にフェッチできる行の最大数。デフォルト値は 4096 です。

OAuth 更新トークンを取得するには、Google の認証サイトに移動します。

```
https://accounts.google.com/o/oauth2/auth?
scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive+https%3A%2F%2Fspreadsh
eets.google.com%2Ffeeds&redirect_uri=urn:ietf:wg:oauth:2.0:oob&response_type
=code&client_id=217138521084.apps.googleusercontent.com
```

次に、認証コードを以下の POST 要求にコピーし、コマンドラインで実行します (更新トークンが応答に含まれます)。

```
curl \--data-urlencode code=<AUTH_CODE> \
--data-urlencode client_id=217138521084.apps.googleusercontent.com \
--data-urlencode client_secret=gXQ6-lOkEjE1lVcz7giB4Poy \
--data-urlencode redirect_uri=urn:ietf:wg:oauth:2.0:oob \
--data-urlencode grant_type=authorization_code
https://accounts.google.com/o/oauth2/token
```

4.9.6. Infinispan データソース

Infinispan データソースは、インストール時に EAP にデプロイされた Teiid 固有の JCA コネクターを使用します。このコネクターは、Infinispan キャッシュの以下のモードをサポートするよう設定できます。

表4.3 レジストリープロパティー

キャッシュタイプ	このキャッシュの取得方法
Local Cache	JNDI
Local Cache	設定ファイル
Remote Cache	JNDI
Remote Cache	1 つまたは複数のホストポートを指定します。
Local Cache	1 つまたは複数の Hot Rod クライアントプロパティーファイルを指定します。

このコネクターには以下のプロパティーを設定できます。

表4.4 レジストリープロパティー

プロパティー名	必要性	プロパティーテンプレート	説明
CacheTypeMap	はい	cacheName:className[;pkFieldName] [,cacheName:className[;pkFieldName]..]	ルート Java オブジェクトクラス名をキャッシュにマップし、キャッシュに対するプライマリーキーである属性を指定します。
module	なし	なし	CacheTypeMap で定義されたキャッシュクラスを含む JBoss AS モジュールを指定します
CacheJndiName	なし	なし	CacheContainer を見つけるための JNDI 名
RemoteServerList	なし	host:port[;host:port….]	CacheTypeMap で定義されたキャッシュにアクセスするために一緒にクラスタ化するホストおよびポートを指定します。
ConfigurationFileNameForLocalCache	なし	なし	ローカルキャッシュを設定するための Infinispan 設定 xml ファイル。
HotRodClientPropertiesFile	なし	なし	リモートキャッシュに対する接続を設定するための HotRod プロパティーファイル。

データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しいディレクトリー名と他のプロパティーを提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=infinispan/connection-
definitions=infinispanDS:add(jndi-name=java:/infinispanDS, class-
name=org.teiid.resource.adapter.infinispan.InfinispanManagedConnectionFactor
y, enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=infinispan/connection-
definitions=infinispanDS/config-
properties=CacheTypeMap:add(value=trades:org.somewhere.Trade;tradeId)
/subsystem=resource-adapters/resource-adapter=infinispan/connection-
definitions=infinispanDS/config-properties=Module:add(value=org.somewhere)
/subsystem=resource-adapters/resource-adapter=infinispan/connection-
definitions=infinispanDS/config-
properties=CacheJndiName:add(value=java:/myCache)
runbatch
```

この Infinispan コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=infinispan)
```

重要

Classloading 問題の回避: サーブレシーブまたは他の種類の Web アプリケーションを使用してキャッシュの DefaultCacheManager を作成する場合は、アプリケーションに Infinispan jar 依存関係を含めずに、代わりにモジュール依存関係を追加してください。

4.9.7. Infinispan-DSL データソース

このリソースアダプターを使用するには、Red Hat JBoss Data Grid モジュールをインストールする必要があります。このトランスレーターは、リモートモードとローカルモードで異なる JBoss Data Grid モジュールを使用することに注意してください。詳細は JBoss Data Grid のドキュメント https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Data_Grid/6.4/html/Infinispan_Query_Guide/chap-The_Infinispan_Query_DSL.html を参照してください。この JBoss Data Grid ガイドには、Infinispan Query DSL の使用に関する基本情報が記載されています。ライブラリーモードに必要な jar は infinispan-embedded-query です。リモートクライアントサーバーモードに必要な jar は、デフォルトで jboss-datagrid-server とともにインストールされる infinispan-query です。

Infinispan-DSL データソースは、インストール時に EAP にデプロイされた Teiid 固有の JCA コネクターを使用します。このコネクターは、Hot Rod クライアントを使用してアクセスする Infinispan キャッシュの以下のモードをサポートするよう設定できます。

表4.5 プロパティ

キャッシュタイプ	プロパティー名	キャッシュの取得方法
Remote Cache	CacheJndiName	JNDI
Remote Cache	RemoteServerList	サーバーリスト: 1 つまたは複数のホストポートを指定します

キャッシュタイプ	プロパティー名	キャッシュの取得方法
Remote Cache	HotRodClientPropertiesFile	HotRod クライアントプロパティーファイル

これには、オブジェクトのシリアル化のために Google Protobuf を使用するよう設定された JDG 6.3 以上が必要です。Teiid が Lucene インデックスで設定されたローカルキャッシュのようにキャッシュを問い合わせることができます。

このコネクターには以下のプロパティーを設定できます。

表4.6 プロパティ

プロパティー	必要性	プロパティーテンプレート	説明
CacheTypeMap	はい	cacheName:className[;pkFieldName] [,cacheName:className[;pkFieldName]..]	ルート Java オブジェクトクラス名をキャッシュにマップし、キャッシュに対するプライマリーキーである属性を指定します。
ProtobinFile	はい	なし	jar にパッケージ化された Google Protobin ファイルへのパス (/quickstart/addressbook.protobin など)
MessageMarshallers	はい	marshaller [,marshaller,..]	各メッセージマーシャラーがマップされたクラス名を含みます (class:marshaller, [class:marshaller,..])。シリアル化のために登録されます。
MessageDescriptor	はい	なし	キャッシュ内のルートオブジェクトに対するメッセージ記述子クラス名
module	なし	なし	ロードする必要があるキャッシュクラスを含む JBoss EAP モジュールを指定します。
CacheJndiName	なし	なし	CacheContainer を見つけるための JNDI 名
RemoteServerList	なし	host:port[;host:port….]	キャッシュにアクセスするために一緒にクラスタ化するホストおよびポートを指定します。
HotRodClientPropertiesFile	なし	なし	リモートキャッシュに対する接続を設定するための HotRod プロパティーファイル。

以下に、JDG remote-query クイックスタートに接続するために使用する resource-adapter の XML コードを示します。

```
<resource-adapter id="infinispanRemQS">
     <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.dsl"/>
     <connection-definitions>
         <connection-definition class-
name="org.teiid.resource.adapter.infinispan.dsl.InfinispanManagedConnectionF
actory" jndi-name="java:/infinispanRemote" enabled="true" use-java-
context="true" pool-name="infinispanDS">
             <config-property name="CacheTypeMap">
```

```
addressbook:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person;id
             </config-property>
             <config-property name="ProtobinFile">
                  /quickstart/addressbook.protobin
             </config-property>
             <config-property name="MessageDescriptor">
                 quickstart.Person
             </config-property>
             <config-property name="Module">
                 com.client.quickstart.pojos
             </config-property>
             <config-property name="MessageMarshallers">
org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person:org.jboss.as.qu
ickstarts.datagrid.hotrod.query.marshallers.PersonMarshaller,org.jboss.as.qu
ickstarts.datagrid.hotrod.query.domain.PhoneNumber:org.jboss.as.quickstarts.
datagrid.hotrod.query.marshallers.PhoneNumberMarshaller,org.jboss.as.quickst
arts.datagrid.hotrod.query.domain.PhoneType:org.jboss.as.quickstarts.datagri
d.hotrod.query.marshallers.PhoneTypeMarshaller
             </config-property>
             <config-property name="RemoteServerList">
                 127.0.0.1:11322
             </config-property>
         </connection-definition>
     </connection-definitions>
 </resource-adapter>
```

4.9.8. JDG HotRod データソース

Red Hat JBoss Data Grid HotRod データソースは、インストール時にデプロイされている Red Hat JBoss Data Virtualization 固有の JCA コネクターを使用します。これは、Hot Rodクライアントを使用する以下のキャッシュタイプをサポートするように設定できます。

表4.7 キャッシュタイプ

リモートキャッシュ	リモートキャッシュ	キャッシュの取得手段
リモートキャッシュ	CacheJndiName	JNDI の使用
リモートキャッシュ	RemoteServerList	サーバーリスト: 1 つ以上の host:port's を指定します
リモートキャッシュ	HotRodClientPropertiesFile	HotRod クライアントプロパティーファイル

- (オプション 1) 最小限、JDG 6.2 - この場合、protobuf の定義ファイルと pojo がキャッシュされるための pojo マーシャラーを指定する必要があります。
- (オプション 2) 最小限、JDG 6.6 - これは、pojo に注釈があり、これが JDG による protobuf 定義 と pojo マーシャラーの作成を開始する場合に使用できます。

pojo クラスは、キャッシュにデータ保存する際に使用するオブジェクトです。以下を実行するために構築されます。

- キャッシュがインデックス対応となっていることを活用するには、クラスに注釈を付けます。JDG Indexing With Protobuf Annotations (https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Data_Grid/6.6/html-single/Infinispan_Query_Guide/index.html#Custom_Fields_Indexing_with_Protobuf) を参照してください。

- クラスは jar にパッケージ化して、モジュールとしてデプロイできるようにします。

この pojo を設定するには、pojo jar をモジュールとして Red Hat JBoss EAP サーバーにデプロイし、 -vdb.xml 内で "lib" プロパティーを定義して適切なモジュール名を割り当てます。これは、以下のテンプレートを使用して実行できます。

```
<property name ="lib" value ="{pojo_module_name}"></property>
```

これはキャッシュからの読み込みおよびキャッシュへの書き込みに必須となります。

表4.8 必須プロパティー

プロパティー名	プロパティーテンプレート	説明
CacheTypeMap	cacheName:className[;pkFieldName[:cacheKeyJavaType]]	ルート Java オブジェクトクラス名をキャッシュにマップします。また、キャッシュに対するプライマリーキーとなるクラス属性を指定します。これはオプションではあるものの更新には必須となります。クラス属性タイプが異なる場合は、プラマリーキーの Java タイプを指定します。

以下のプロパティーを設定して RemoteCacheManager の作成およびアクセス方法を定義します。

表4.9 設定プロパティー

プロパティー名	必要性	プロパティーテンプレート	説明
CacheJndiName	なし	NA	これは、CacheContainer を見つけるための JNDI 名です。
RemoteServerList	なし	host:port\[;host:port….\]	キャッシュにアクセスするために一緒にクラスタ化するホストおよびポートを指定します。
HotRodClientPropertiesFile	なし	NA	リモートキャッシュに対する接続を設定するための HotRod プロパティーファイルです。

pojo に注釈がない場合は、以下のプロパティーが必要になります。

表4.10 設定プロパティー

プロパティー名	必要性	プロパティーテンプレート	説明
ProtobinFile	はい	NA	jar にパッケージ化された Google Protobin ファイルへのパス (/quickstart/addressbook.protobin など)

プロパティー名	必要性	プロパティーテンプレート	説明
MessageMarshallers	はい	marshaller \ [,marshaller,..\]	各メッセージマーシャラーがマップされたクラス名を含みます (class:marshaller,\ [class:marshaller,..\])。シリアル化のために登録されます。
MessageDescriptor	はい	キャッシュ内のルートオブジェクトに対するメッセージ記述子クラス名です。	リモートキャッシュに対する接続を設定するための HotRod プロパティーファイルです。

外部実体化にリモートキャッシュを使用する場合は、以下のプロパティーの設定が必要になります。

表4.11 設定プロパティー

プロパティー名	必要性	説明
StagingCacheName	はい	実体化に使用するステージングキャッシュのキャッシュ名
AliasCacheName	はい	実体化に使用するキャッシュのエイリアス追跡に使用するエイリアスキャッシュのキャッシュ名

データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、リソースアダプターの XML ソースを表示しています。このコードは、JDG remote-query quick start:me への接続に使用されます。

```
 <resource-adapter id="infinispanRemQS">
                    <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.hotrod"/>
                    <connection-definitions>
                        <connection-definition class-
name="org.teiid.resource.adapter.infinispan.hotrod.InfinispanManagedConnecti
onFactory" jndi-name="java:/infinispanRemote" enabled="true" use-java-
context="true" pool-name="infinispanDS">
                            <config-property name="CacheTypeMap">

addressbook:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person;id
                            </config-property>
                            <config-property name="ProtobinFile">
                                 /quickstart/addressbook.protobin
                            </config-property>
                            <config-property name="MessageDescriptor">
                                quickstart.Person
                            </config-property>
                            <config-property name="Module">
                                com.client.quickstart.pojos
                            </config-property>
                            <config-property name="MessageMarshallers">
org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Person:org.jboss.as.qu
ickstarts.datagrid.hotrod.query.marshallers.PersonMarshaller,org.jboss.as.qu
ickstarts.datagrid.hotrod.query.domain.PhoneNumber:org.jboss.as.quickstarts.
datagrid.hotrod.query.marshallers.PhoneNumberMarshaller,org.jboss.as.quickst
```

```
arts.datagrid.hotrod.query.domain.PhoneType:org.jboss.as.quickstarts.datagri
d.hotrod.query.marshallers.PhoneTypeMarshaller
                            </config-property>
                            <config-property name="RemoteServerList">
                                127.0.0.1:11322
                            </config-property>
                        </connection-definition>
                    </connection-definitions>
                </resource-adapter>
```

以下は、外部実体化のための設定例です。

```
                <resource-adapter id="infinispanRemQSDSL">
                    <module slot="main" id="org.jboss.teiid.resource-
adapter.infinispan.hotrod"/>
                    <connection-definitions>
                        <connection-definition class-
name="org.teiid.resource.adapter.infinispan.hotrod.InfinispanManagedConnecti
onFactory" jndi-name="java:/infinispanRemoteDSL" enabled="true" use-java-
context="true" pool-name="infinispanRemoteDSL">
                            <config-property name="CacheTypeMap">

addressbook_indexed:org.jboss.as.quickstarts.datagrid.hotrod.query.domain.Pe
rson;id
                            </config-property>
                            <config-property name="StagingCacheName">
                                addressbook_indexed_mat
                            </config-property>
                            <config-property name="AliasCacheName">
                                aliasCache
                            </config-property>
                            <config-property name="Module">
                                com.client.quickstart.addressbook.pojos
                            </config-property>
                            <config-property name="RemoteServerList">
                                127.0.0.1:11322
                            </config-property>
                        </connection-definition>
                    </connection-definitions>
                </resource-adapter>
```

4.9.9. JDBC データソース

4.9.9.1. JDBC データソース

以下に Oracle データソースを設定する例を示します。このプロセスはデータベースベンダーに関係なくほぼ同じです。通常は、JDBC jar と接続 URL およびユーザークレデンシャルなどの設定が変わります。

これらは、**jboss-install/docs/teiid/datasources** ディレクトリー内のすべてのデータソースの設定テンプレートです。データソースを JBoss EAP に追加する方法もここで詳細に説明されています。データソースを作成するには以下の 2 つの方法があります。

最初に、必要な JDBC jar ファイルをデプロイします。たとえば、Oracle データソースを作成する場合は、最初に "ojdbc6.jar" ファイルをデプロイする必要があります: **`deploy /path/to/ojdbc6.jar`**

注記

JBoss EAPis がスタンドアロンモードで実行されている場合は、この 'ojdbc6.jar" を **`jboss-install>/standalone/deployments`** ディレクトリーに手動でコピーして CLI ツールを使用せずに自動的にデプロイすることもできます。

ここで、このドライバーを使用してデータソースを作成します。データソースを作成するには多くの方法があります (CLI、AdminShell、または管理コンソールを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
/subsystem=datasources/data-source=oracel-ds:add(jndi-name=java:/OracleDS,
driver-name=ojdbc6.jar, connection-url=jdbc:oracle:thin:
{host}:1521:orcl,user-name={user}, password={password})
/subsystem=datasources/data-source=oracel-ds:enable
```

4.9.9.2. JDBC データソースの設定

JBoss Data Virtualization が JDBC データソースに接続する場合は、以下の 2 つの手順に従って、JBoss EAP インスタンスでデータソースを設定する必要があります。

1. JDBC JAR ドライバーファイルをインストール (またはデプロイ) します。
2. JBoss EAP でデータソースとして作成 (または設定) します。

注記

データソースの設定の包括的な情報については、『Red Hat JBoss Enterprise Application Platform 管理および設定ガイド』を参照してください。

4.9.9.3. 設定例

設定例 (サーバー設定ファイルに含まれます) は、***`EAP_HOME`*`/docs/teiid/datasources/`** ディレクトリーにあります。

注記

この設定では、使用しているインストールに応じてファイルパスとプロパティーを調整する必要があります。JNDI 名は VDB で使用されているのと同じ JNDI 名である必要があります。

4.9.9.4. 管理 CLI で JDBC ドライバーをインストール

手順4.5 管理 CLI で JDBC ドライバーをインストール

1. 管理 CLI を起動します: **./EAP_HOME/bin/jboss-cli.sh**
2. **connect** コマンドを入力します。
3. **deploy *PATH/FILE*.jar** コマンドを入力します: **deploy ojdbc6.jar**
4. **quit** コマンドを入力します。

4.9.9.5. 管理コンソールを用いた JDBC ドライバーのインストール

サマリー

アプリケーションが JDBC データソースに接続する前に、データソースベンダーの JDBC ドライバーを JBoss EAP 6 が使用できる場所にインストールする必要があります。JBoss EAP 6 では、これらのドライバーを他のデプロイメントと同じようにデプロイできます。そのため、管理対象ドメインを使用する場合は、サーバーグループ内の複数のサーバー全体でドライバーをデプロイできます。

前提条件

このタスクを実行する前に、以下の前提条件を満たしている必要があります。

- データベースのベンダーから JDBC ドライバーをダウンロードする必要があります。

注記

JDBC 4 対応のドライバーは自動的に認識され、名前とバージョンによってシステムへインストールされます。JDBC JAR は、Java サービスプロバイダーのメカニズムを使用して識別されます。このような JAR には、JAR のドライバークラスの名前が含まれる META-INF/services/java.sql.Driver テキストが含まれています。

手順4.6 JDBC ドライバー JAR の編集

JDBC ドライバー JAR が JDBC 4 未対応である場合、以下の方法でデプロイ可能にすることができます。

1. 空の一時ディレクトリーに移動するか、空の一時ディレクトリーを作成します。
2. META-INF サブディレクトリーを作成します。
3. META-INF/services サブディレクトリーを作成します。
4. JDBC ドライバーの完全修飾クラス名を示す 1 行が含まれる、META-INF/services/java.sql.Driver ファイルを作成します。
5. JAR コマンドラインツールを使用して、次のように JAR を更新します。

```
jar \-uf jdbc-driver.jar META-INF/services/java.sql.Driver
```

6. 「管理コンソールにログインします。」
7. 管理ドメインを使っている場合は、JAR ファイルをサーバーグループにデプロイします。それ以外の場合は、サーバーにデプロイします。詳細については、『JBoss EAP 6 管理および設定ガイド』の項「管理コンソールでのデプロイ」を参照してください。

結果:

JDBC ドライバーがデプロイされ、アプリケーションが使用できるようになります。

4.9.9.6. 管理インターフェースによる非 XA データソースの作成

サマリー

ここでは、管理コンソールまたは管理 CLI のいずれかを使用して非 XA データソースを作成する手順について取り上げます。

前提条件

- JBoss EAP 6 サーバーが稼働している必要があります。

注記

バージョン 10.2 以前の Oracle データソースでは非トランザクション接続とトランザクション接続が混在するとエラーが発生したため、<no-tx-separate-pools/> パラメーターが必要でした。一部のアプリケーションでは、このパラメーターが不要になりました。

注記

ドライバーリストの重複、選択したドライバーがプロファイルで使用できない、プロファイルのサーバーが稼働していないとドライバーが表示されないなどの問題を防ぐため、JBoss EAP 6.4 以降ではドメインモードの管理コンソールを使用してデータソースが作成される間は、モジュールとしてインストールされ、プロファイルから適切に参照される JDBC ドライバーのみが検出可能です。

手順4.7 管理 CLI または管理コンソールのいずれかを使用したデータソースの作成

- A. **管理 CLI**

    a. CLI ツールを起動し、サーバーに接続します。

    b. 以下の管理 CLI コマンドを実行して非 XA データソースを作成し、適切に変数を設定します。

注記

DRIVER_NAME の値は、JDBC ドライバー JAR にある **`/META-INF/services/java.sql.Driver`** ファイルにリストされたクラスの数によって異なります。クラスが 1 つしかない場合、この値は JAR の名前になります。クラスが複数ある場合、この値は JAR + driverClassName + "_" + majorVersion +"_" + minorVersion の名前になります。失敗した場合は、以下のエラーがログに記録されます。

```
JBAS014775:    New missing/unsatisfied dependencies
```

たとえば、MySQL 5.1.31 ドライバーで必要な *DRIVER_NAME* 値は **`mysql-connector-java-5.1.31-bin.jarcom.mysql.jdbc.Driver_5_1`** です。

```
data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --
driver-name=DRIVER_NAME  --connection-url=CONNECTION_URL
```

c. データソースを有効にします。

```
data-source enable --name=DATASOURCE_NAME
```

B. **管理コンソール**

a. 管理コンソールへログインします。

b. **管理コンソールの Datasources パネルに移動します。**

i. コンソールの上部から **Configuration** タブを選択します。

ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。

iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。

iv. コンソールの左側にあるメニューより **Datasources** を選択します。

c. **新しいデータソースを作成します。**

i. **Datasources** パネルの上部にある **Add** を選択します。

ii. **Create Datasource** ウィザードで新しいデータソースの属性を入力し、**Next** ボタンを押します。

iii. **Create Datasource** ウィザードで JDBC ドライバーの詳細を入力し、**Next** をクリックします。

iv. **Create Datasource** ウィザードで接続設定を入力します。

v. **Test Connection** ボタンをクリックしてデータソースへの接続をテストし、設定が正しいことを確認します。

vi. **Done** をクリックして終了します。

結果:

非 XA データソースがサーバーに追加されます。**standalone.xml** または **domain.xml** ファイル、および管理インターフェースで追加を確認できます。

4.9.9.7. 管理インターフェースによる XA データソースの作成

サマリー

ここでは、管理コンソールまたは管理 CLI のいずれかを使用して XA データソースを作成する手順について取り上げます。

注記

バージョン 10.2 以前の Oracle データソースでは非トランザクション接続とトランザクション接続が混在するとエラーが発生したため、<no-tx-separate-pools/> パラメーターが必要でした。一部のアプリケーションでは、このパラメーターが不要になりました。

手順4.8 管理 CLI または管理コンソールのいずれかを使用した XA データソースの作成

A. **管理 CLI**

a. 「管理 CLI の起動」.

b. 以下の管理 CLI コマンドを実行して XA データソースを作成し、適切に変数を設定します。

注記

DRIVER_NAME の値は、JDBC ドライバー JAR にある **/META-INF/services/java.sql.Driver** ファイルにリストされたクラスの数によって異なります。クラスが 1 つしかない場合、この値は JAR の名前になります。クラスが複数ある場合、この値は JAR + driverClassName + "_" + majorVersion +"_" + minorVersion の名前になります。失敗した場合は、以下のエラーがログに記録されます。

```
JBAS014775:    New missing/unsatisfied dependencies
```

たとえば、MySQL 5.1.31 ドライバーで必要な *DRIVER_NAME* 値は **mysql-connector-java-5.1.31-bin.jarcom.mysql.jdbc.Driver_5_1** です。

```
xa-data-source add --name=XA_DATASOURCE_NAME --jndi-
name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-
class=XA_DATASOURCE_CLASS
```

c. **XA データソースプロパティーの設定**

i. **サーバー名の設定**

次のコマンドを実行し、ホストのサーバー名を設定します。

```
/subsystem=datasources/xa-data-
source=XA_DATASOURCE_NAME/xa-datasource-
properties=ServerName:add(value=HOSTNAME)
```

ii. **データベース名の設定**

次のコマンドを実行し、データベース名を設定します。

```
/subsystem=datasources/xa-data-
source=XA_DATASOURCE_NAME/xa-datasource-
properties=DatabaseName:add(value=DATABASE_NAME)
```

d. データソースを有効にします。

```
xa-data-source enable --name=XA_DATASOURCE_NAME
```

B. **管理コンソール**

a. 「管理コンソールにログインします。」.

b. **管理コンソールの `Datasources` パネルに移動します。**

i. コンソールの上部から **`Configuration`** タブを選択します。

ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。

iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。

iv. **Datasources** を選択します。

c. **`XA Datasource`** タブを選択します。

d. **新しい XA データソースを作成します。**

i. **追加** をクリックします。

ii. **`Create XA Datasource`** ウィザードに新しい XA データソースの属性を入力し、**`Next`** をクリックします。

iii. **`Create XA Datasource`** ウィザードに JDBC ドライバーの詳細を入力し、**`Next`** をクリックします。

iv. XA プロパティーを入力し、**`Next`** をクリックします。

v. **`Create XA Datasource`** ウィザードで接続設定を入力します。

vi. **`Test Connection`** ボタンをクリックして XA データソースへの接続をテストし、設定が正しいことを確認します。

vii. **Done** をクリックして終了します。

結果:

XA データソースがサーバーに追加されます。追加内容は **`standalone.xml`** または **`domain.xml`** ファイルのどちらかと、管理インターフェースで確認することができます。

4.9.9.8. データソースのパラメーター

表4.12 非 XA および XA データソースに共通のデータソースパラメーター

Parameter	説明
jndi-name	データソースの一意の JNDI 名。
pool-name	データソースの管理プール名。
enabled	データソースが有効かどうかを指定します。
use-java-context	データソースをグローバルの JNDI にバインドするかどうかを指定します。

Parameter	説明
spy	JDBC レイヤーで **spy** 機能を有効にします。この機能は、データソースへの JDBC トラフィックをすべてログに記録します。ロギングカテゴリーの **jboss.jdbc.spy** もロギングサブシステムのログレベルである **DEBUG** に設定する必要があることに注意してください。
use-ccm	キャッシュ接続マネージャーを有効にします。
new-connection-sql	接続プールに接続が追加された時に実行する SQL ステートメント。
transaction-isolation	次のいずれかになります。 - TRANSACTION_READ_UNCOMMITTED - TRANSACTION_READ_COMMITTED - TRANSACTION_REPEATABLE_READ - TRANSACTION_SERIALIZABLE - TRANSACTION_NONE
url-selector-strategy-class-name	インターフェース **org.jboss.jca.adapters.jdbc.URLSelectorStrategy** を実装するクラス。
セキュリティー	セキュリティー設定である子要素が含まれます。表4.17「セキュリティーパラメーター」を参照してください。
validation	検証設定である子要素が含まれます。表4.18「検証パラメーター」を参照してください。
timeout	タイムアウト設定である子要素が含まれます。表4.19「タイムアウトパラメーター」を参照してください。
statement	ステートメント設定である子要素が含まれます。表4.20「ステートメントのパラメーター」を参照してください。

表4.13 非 XA データソースのパラメーター

Parameter	説明
jta	非 XA データソースの JTA 統合を有効にします。XA データソースには適用されません。
connection-url	JDBC ドライバーの接続 URL。
driver-class	JDBC ドライバークラスの完全修飾名。
connection-property	**Driver.connect(url,props)** メソッドに渡される任意の接続プロパティー。各 connection-property は、文字列名と値のペアを指定します。プロパティー名は名前、値は要素の内容に基づいています。
pool	プーリング設定である子要素が含まれます。表4.15「非 XA および XA データソースに共通のプールパラメーター」を参照してください。
url-delimiter	高可用性 (HA) クラスター化されたデータベースの connection-url にある URL の区切り文字。

表4.14 XA データソースのパラメーター

Parameter	説明
xa-datasource-property	実装クラス **XADataSource** に割り当てるプロパティー。*name=value* で指定されます。 **set*Name*** という形式で setter メソッドが存在する場合、プロパティーは **set*Name*(*value*)** という形式の setter メソッドを呼び出すことで設定されます。
xa-datasource-class	実装クラス **javax.sql.XADataSource** の完全修飾名。
driver	JDBC ドライバーが含まれるクラスローダーモジュールへの一意な参照です。許可される形式は *driverName#majorVersion.minorVersion* です。
xa-pool	プーリング設定である子要素が含まれます。表4.15「非 XA および XA データソースに共通のプールパラメーター」および表4.16「XA プールパラメーター」を参照してください。
recovery	リカバリー設定である子要素が含まれます。表4.21「リカバリーパラメーター」を参照してください。

表4.15 非 XA および XA データソースに共通のプールパラメーター

Parameter	説明
min-pool-size	プールが保持する最小接続数
max-pool-size	プールが保持可能な最大接続数
prefill	接続プールのプレフィルを試行するかどうかを指定します。要素が空の場合は **true** を示します。デフォルト値は、**false** です。
use-strict-min	**min-pool-size** に達した後、アイドル接続スキャンがさらに閉じられた接続のマーク付けを厳密に停止するかどうかを指定します。
flush-strategy	エラーの場合にプールをフラッシュするかどうかを指定します。有効な値は次の通りです。 - FailingConnectionOnly - IdleConnections - EntirePool デフォルト値は **FailingConnectionOnly** です。
allow-multiple-users	複数のユーザーが getConnection(user, password) メソッドを使いデータソースへアクセスするかどうか、および内部プールタイプがこの動作に対応するかを指定します。

表4.16 XA プールパラメーター

Parameter	説明
is-same-rm-override	**javax.transaction.xa.XAResource.isSameRM(XAResource)** クラスが **true** あるいは **false** のどちらを返すかを指定します。
interleaving	XA 接続ファクトリーのインターリービングを有効にするかどうかを指定します。

Parameter	説明
no-tx-separate-pools	コンテキスト毎に sub-pool を作成するかどうかを指定します。これには Oracle のデータソースが必要ですが、このデータソースは JTA トランザクションの内部、外部に関わらず、XA 接続の利用ができなくなります。 このオプションを使用すると 2 つの実際のプールが作成されるため、プールサイズの合計が **max-pool-size** の 2 倍になります。
pad-xid	Xid のパディングを行うかどうかを指定します。
wrap-xa-resource	XAResource を **org.jboss.tm.XAResourceWrapper** インスタンスでラップするかどうかを指定します。

表4.17 セキュリティーパラメーター

Parameter	説明
user-name	新規接続の作成に使うユーザー名
password	新規接続の作成に使うパスワード
security-domain	認証処理を行う JAAS security-manager 名が含まれます。この名前は、JAAS ログイン設定の application-policy/name 属性に相関します。
reauth-plugin	物理接続の再認証に使う再認証プラグインを定義します。

表4.18 検証パラメーター

Parameter	説明
valid-connection-checker	**SQLException.isValidConnection(Connection e)** メソッドを提供し接続を検証するインターフェース **org.jboss.jca.adaptors.jdbc.ValidConnectionChecker** の実装。例外が発生すると接続が破棄されます。存在する場合、**check-valid-connection-sql** パラメーターが上書きされます。
check-valid-connection-sql	プール接続の妥当性を確認する SQL ステートメント。これは、管理接続をプールから取得し利用する場合に呼び出される場合があります。

Parameter	説明
validate-on-match	接続ファクトリーが指定のセットに対して管理された接続に一致させようとした時に接続レベルの検証を実行するかどうかを示します。 通常、**validate-on-match** に true を指定したときに **background-validation** を true に指定することはありません。クライアントが使用する前に接続を検証する必要がある場合に **Validate-on-match** が必要になります。このパラメーターはデフォルトでは false になっています。
background-validation	接続がバックグラウンドスレッドで検証されることを指定します。**validate-on-match** を使用しない場合、バックグラウンドの検証はパフォーマンスを最適化します。**validate-on-match** が true のときに **background-validation** を使用すると、チェックが冗長になることがあります。バックグラウンド検証では、不良の接続がクライアントに提供される可能性があります (検証スキャンと接続がクライアントに提供されるまでの間に接続が悪くなります)。そのため、クライアントアプリケーションはこの接続不良の可能性に対応する必要があります。
background-validation-millis	バックグラウンド検証を実行する期間 (ミリ秒単位)。
use-fast-fail	true の場合、接続が無効であれば最初に接続を割り当てしようとした時点で失敗します。デフォルト値は **false** です。
stale-connection-checker	ブール値の **isStaleConnection(SQLException e)** メソッドを提供する **org.jboss.jca.adapters.jdbc.StaleConnectionChecker** のインスタンス。このメソッドが **true** を返すと、**SQLException** のサブクラスである **org.jboss.jca.adapters.jdbc.StaleConnectionException** に例外がラップされます。
exception-sorter	ブール値である **isExceptionFatal(SQLException e)** メソッドを提供する **org.jboss.jca.adapters.jdbc.ExceptionSorter** のインスタンス。このメソッドは、例外が **connectionErrorOccurred** メッセージとして **javax.resource.spi.ConnectionEventListener** のすべてのインスタンスへブロードキャストされるかどうかを検証します。

表4.19 タイムアウトパラメーター

Parameter	説明

Parameter	説明
use-try-lock	**`lock()`** の代わりに **`tryLock()`** を使用します。これは、ロックが使用できない場合に即座に失敗するのではなく、設定された秒数間ロックの取得を試みます。デフォルト値は **`60`** 秒です。たとえば、タイムアウトを 5 分に設定するには、**`<use-try-lock>`**300**`</use-try-lock>`** を設定します。
blocking-timeout-millis	接続待機中にブロックする最大時間 (ミリ秒)。この時間を超過すると、例外が発生します。これは、接続許可の待機中のみブロックし、新規接続の作成に長時間要している場合は例外が発生しません。デフォルト値は 30000 (30 秒) です。
idle-timeout-minutes	アイドル接続が切断されるまでの最大時間 (分単位)。実際の最大時間は idleRemover のスキャン時間によって異なります。idleRemover のスキャン時間はプールの最小 **`idle-timeout-minutes`** の半分になります。
set-tx-query-timeout	トランザクションがタイムアウトするまでの残り時間を基にクエリーのタイムアウトを設定するかどうかを指定します。トランザクションが存在しない場合は設定済みのクエリーのタイムアウトが使用されます。デフォルト値は **`false`** です。
query-timeout	クエリーのタイムアウト (秒)。デフォルト値はタイムアウトなしです。
allocation-retry	例外を発生させる前に接続の割り当てを再試行する回数。デフォルト値は **`0`** で、初回の割り当て失敗で例外が発生します。
allocation-retry-wait-millis	接続の割り当てを再試行するまで待機する期間 (ミリ秒単位)。デフォルト値は 5000 (5 秒) です。
xa-resource-timeout	ゼロでない場合、この値は **`XAResource.setTransactionTimeout`** メソッドへ渡されます。

表4.20 ステートメントのパラメーター

Parameter	説明
track-statements	接続がプールへ返され、ステートメントが準備済みステートメントキャッシュへ返された時に、閉じられていないステートメントをチェックするかどうかを指定します。false の場合、ステートメントは追跡されません。 **有効な値** - **`true`**: ステートメントと結果セットが追跡され、ステートメントが閉じられていない場合は警告が出力されます。 - **`false`**: ステートメントと結果セットのいずれも追跡されません。 - **`nowarn`**: ステートメントは追跡されますが、警告は出力されません。これがデフォルト設定となっています。

Parameter	説明
prepared-statement-cache-size	LRU (Least Recently Used) キャッシュにある接続毎の準備済みステートメントの数。
share-prepared-statements	閉じずに同じステートメントを 2 回要求した場合に、同じ基盤の準備済みステートメントを使用するかどうかを指定します。デフォルト値は **`false`** です。

表4.21 リカバリーパラメーター

Parameter	説明
recover-credential	リカバリーに使用するユーザー名とパスワードのペア、あるいはセキュリティードメイン。
recover-plugin	リカバリーに使用される **`org.jboss.jca.core.spi.recovery.RecoveryPlugin`** クラスの実装。

4.9.9.9. JDBC ドライバーをダウンロードできる場所

下表は、JBoss EAP 6 と使用される一般的なデータベースの JDBC ドライバーをダウンロードできる場所を示しています。

注記

これらのリンク先は他社の Web サイトであるため、Red Hat は管理しておらず、積極的に監視も行っていません。ご使用のデータベースの最新ドライバーについては、データベースベンダーのドキュメントおよび Web サイトを確認してください。

表4.22 JDBC ドライバーをダウンロードできる場所

ベンダー	ダウンロード場所
MySQL	http://www.mysql.com/products/connector/
PostgreSQL	http://jdbc.postgresql.org/
Oracle	http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
IBM	http://www-306.ibm.com/software/data/db2/java/
Sybase	http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect
Microsoft	http://msdn.microsoft.com/data/jdbc/

4.9.10. LDAP データソース

LDAP データソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。ldap データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=ldap/connection-
definitions=ldapDS:add(jndi-name=java:/ldapDS, class-
name=org.teiid.resource.adapter.ldap.LDAPManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=ldap/connection-
definitions=ldapDS/config-properties=LdapUrl:add(value=ldap://ldapServer:389)
/subsystem=resource-adapters/resource-adapter=ldap/connection-
definitions=ldapDS/config-properties=LdapAdminUserDN:add(value=
{cn=???,ou=???,dc=???})
/subsystem=resource-adapters/resource-adapter=ldap/connection-
definitions=ldapDS/config-properties=LdapAdminUserPassword:add(value={pass})
/subsystem=resource-adapters/resource-adapter=ldap/connection-
definitions=ldapDS/config-properties=LdapTxnTimeoutInMillis:add(value=-1)
/subsystem=resource-adapters/resource-adapter=ldap:activate
runbatch
```

この LDAP コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=ldap)
```

4.9.11. MongoDB データソース

MongoDB データソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。MongoDB データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=mongodb/connection-
definitions=mongodbDS:add(jndi-name="java:/mongoDS", class-
name=org.teiid.resource.adapter.mongodb.MongoDBManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=mongodb/connection-
definitions=mongodbDS/config-properties=RemoteServerList:add(value="
{host}:27017")
/subsystem=resource-adapters/resource-adapter=mongodb/connection-
definitions=mongodbDS/config-properties=Database:add(value="{db-name}")
/subsystem=resource-adapters/resource-adapter=mongodb:activate
runbatch
```

以下に、RAR ファイルで定義されたプロパティーを示します。

表4.23 プロパティ

プロパティー	説明	必要性	デフォルト

プロパティー	説明	必要性	デフォルト
RemoteServerList	サーバーの場所のカンマ区切りのリスト。各場所には、host:port という形式でオプションのポートを含めることができます。	不必要	該当なし
Username	接続ユーザーの名前	不必要	なし
Password	接続ユーザーのパスワード	不必要	なし
Database	MongoDB データベース名	必要	なし

この MongoDB コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=mongodb)
```

4.9.12. Apache Phoenix データソース

以下は、Phoenix データソースの設定方法の例です。これは、Apache HBase トランスレーターの使用前に必要となります。

Phoenix データソースの設定テンプレートは EAP_HOME>/docs/teiid/datasources ディレクトリーにあります。データソースを Red Hat JBoss EAP に追加する方法もここに記載されています。

Phoenix データソースの設定は JDBC データソースの設定とほぼ同じです。最初に、Phoenix ドライバー jar をデプロイします。以下の CLI コマンドを使用して Phoenix ドライバーをデプロイします。

```
module add --name=org.apache.phoenix --resources=/path/to/phoenix-[version]-
client.jar --
dependencies=javax.api,sun.jdk,org.apache.log4j,javax.transaction.api
/subsystem=datasources/jdbc-driver=phoenix:add(driver-name=phoenix,driver-
module-name=org.apache.phoenix,driver-class-
name=org.apache.phoenix.jdbc.PhoenixDriver
```

ドライバー jar は Phoenix のドキュメントからダウンロードできます。

次に、JDBC データベースの作成と同様に、デプロイされたドライバーを基にデータソースを作成します。以下の CLI コマンドを使用してデータソースを作成します。

```
/subsystem=datasources/data-source=phoenixDS:add(jndi-name=java:/phoenixDS,
driver-name=phoenix, connection-url=jdbc:phoenix:{zookeeper quorum server},
enabled=true, use-java-context=true, user-name={user}, password={password})
```

URL、ドライバー、およびその他のプロパティーが適切に設定されていることを確認してください。JNDI 名は VDB で使用した JNDI 名と一致する必要があります。ドライバー名は前のステップでデプロイしたドライバーと一致する必要があります。

URL は HBase ズーキーパー quorum サーバーと一致する必要があります。例は次のとおりです。

```
 jdbc:phoenix [ :<zookeeper quorum> [ :<port number> ] [ :<root node> ] ],
'jdbc:phoenix:127.0.0.1:2181'
```

user-name/password - Phoenix 接続のユーザークレデンシャル

Phoenix テーブルを既存の HBase テーブルにマッピングするには、2 つのステップを行う必要があります。最初のステップでは、phoenix-[version]-server.jar を各 HBase リージョンサーバーのクラスパスにインストールします。この jar を HBase の lib にコピーすると簡単です (詳細は Phoenix のドキュメントを参照してください)。

次に、DDL を実行して Phoenix テーブルを既存の HBase テーブルにマップする必要があります。DDL は Phoenix コマンドラインまたは JDBC から実行できます。

以下の構造で既存の HBase Customer をマッピングする例は次のとおりです。

以下のとおり、HBase Customer テーブルには customer と sales の 2 つのカラムファミリーがあり、各カラムファミリーには 2 つのカラム修飾子があります (customer には name と city、sales には product と amount)。

```
CREATE TABLE IF NOT EXISTS "Customer"("ROW_ID" VARCHAR PRIMARY KEY,
"customer"."city" VARCHAR, "customer"."name" VARCHAR, "sales"."amount"
VARCHAR, "sales"."product" VARCHAR)
```

Phoenix テーブルを既存の HBase テーブルにマップする方法の詳細は、Phoenix ドキュメントを参照してください。

Phoenix テーブルの新規作成は、既存 HBase テーブルへのマッピングと同様です。Phoenix は存在しないすべてのメタデータ (テーブル、カラムファミリー) を作成します。上記の例と同様に、Phoenix/HBase Customer テーブルを作成する DDL は次のようになります。

```
CREATE TABLE IF NOT EXISTS "Customer"("ROW_ID" VARCHAR PRIMARY KEY,
"customer"."city" VARCHAR, "customer"."name" VARCHAR, "sales"."amount"
VARCHAR, "sales"."product" VARCHAR)
```

4.9.13. Salesforce データソース

Salesforce データソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。salesforce データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS:add(jndi-name=java:/sfDS, class-
name=org.teiid.resource.adapter.salesforce.SalesForceManagedConnectionFactor
y, enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS/config-
properties=URL:add(value=https://www.salesforce.com/services/Soap/u/22.0)
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS/config-properties=username:add(value={user})
```

```
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS/config-properties=password:add(value={password})
/subsystem=resource-adapters/resource-adapter=salesforce:activate
runbatch
```

この Salesforce コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=salesforce)
```

4.9.14. Solr データソース

Solr データソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。Solr データソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しい URL とユーザー認証情報を提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=solr/connection-
definitions=solrDS:add(jndi-name=java:/solrDS, class-
name=org.teiid.resource.adapter.solr.SolrManagedConnectionFactory,
enabled=true, use-java-context=true)
/subsystem=resource-adapters/resource-adapter=solr/connection-
definitions=solrDS/config-
properties=url:add(value=http://localhost:8983/solr/)
/subsystem=resource-adapters/resource-adapter=solr/connection-
definitions=solrDS/config-properties=CoreName:add(value=collection1)
/subsystem=resource-adapters/resource-adapter=solr:activate
runbatch
```

この Solr コネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=solr)
```

4.9.15. Red Hat JBoss Data Virtualization による Apache Solr の統合

この例では、Red Hat JBoss Data Virtualiation を使って CSV ファイルから Apache Solr にデータを移動する方法を説明します。そして、Apache Solr にクエリを発行し、DV を使って検索のコンテンツを取得します。Red Hat JBoss Data Virtualization で検索結果が利用可能になったら、他のデータソースと統合してこれをさらに強化することができます。

前提条件

- この例では、赤ちゃんの名前の頻度に関する統計値を収集するデータベースを使用します。これを取得するには、"namesbystate.zip" を http://catalog.data.gov/dataset/baby-names-from-social-security-card-applications-data-by-state-and-district-of- からダウンロードし、"/babynames" という 名前のサブディレクトリーに解凍します。

- Apache Solr を http://lucene.apache.org/solr/ からダウンロードし、"/solr" という 名前のサブディレクトリーにインストールします。

  1. 一時ディレクトリーを作成し、"/solr/example/solr/collection1" をここにコピーします。
  2. "/solr/example/solr/collection1" の名前を "/solr/example/solr/babynames" に変更します。
  3. "/solr/example/solr/babynames/data" ディレクトリーのコンテンツ (ある場合) を削除します。
  4. "/solr/example/solr/babynames/core.properties" ファイルを削除します。
  5. テキストエディターで "/solr/example/solr/babynames/conf/schema.xml" ファイルを開き、"fields" および "copyFields" の下にある要素をすべて削除します。
  6. 以下を "fields" の下に追加します。

     ```
         <field indexed="true" name="_version_" stored="true" type="long"/>
         <field indexed="true" name="_root_" stored="false" type="string"/>
         <field indexed="true" name="id" stored="true" type="string"
     required="true" multiValued="false" />
         <field indexed="true" name="state" stored="true" type="string"
     omitNorms="true"/>
         <field indexed="true" name="name" required="true" stored="true"
     type="string"/>
         <field indexed="true" name="gender" stored="true" type="string"
     omitNorms="true"/>
         <field indexed="true" name="birthyear" stored="true" type="int"/>
         <field indexed="true" name="totalcount" stored="true" type="int"/>
         <field indexed="true" name="text" stored="false"
     type="text_general" multiValued="true"/>
     ```

     これらのフィールドは、上記の baby names CSV ドキュメントにあるデータ向けのスキーマを表しています。schemal.xml ファイルは Solr 向けの内部ドキュメント構造を定義します。

  7. ファイルを保存してから閉じます。
  8. 以下のコマンドを発行して、Apache Solr を起動します。

     ```
     cd "/solr/example"
     java -jar start.jar
     ```

  9. http://localhost:8983/solr に移動します。
  10. 左側にある **`Core Admin`** (**`Unload`** がある場合はこれも) をクリックします。
  11. **`Add Core`** をクリックして、名前に "babies" を、instanceDir に "babynames" を入力します。

      これで Solr は、ドキュメントを保存かつ検索できるコアで設定されました。

ここからは VDB または Dynamic VDB を構築する選択肢があります。まず最初に Teiid Designer を使った VDB の作成方法を説明します。

1. Teiid Designer を起動し、**`Teiid Designer`** パースペクティブに切り替えます。
2. Red Hat JBoss Data Virtualization が適切に設定され、これに接続していることを確認します。
3. 新規の "Teiid Model Project" を作成し、"BabyNames" に名前を変更します。
4. "File - import" を選択してから、"Teiid Designer/File Source Flat - Source and View Model" を選択し

ます。

5. ファイルベースのモデルを作成します。
6. "babies" の表を右クリックして、"Modeling - Preview Data" を選択するとデータのプレビューが表示されます。これが実行できることを確認してください。
7. "File - import" を選択してから "Teiid Designer/Teiid Connection - Source Model" を選択し、**`Next`** をクリックします。
8. "Data the Source to use for Import" ダイアログボックスで "New" をクリックします。
9. Create Data Source ダイアログボックスで名前に "solr-ds" が、ドライバーに "solr" が、"AllowCompression" が true に設定されていることを確認します。
10. Ok をクリックして、Next をクリックします。
11. "translator" が "solr" に設定されていることを確認して、"Next" をクリックします。
12. ダイアログでは "babies" の表でモデルに DDL が表示されます (下図)。Finish ボタンに達するまでクリックします。

```
 CREATE FOREIGN TABLE babies (
        id string OPTIONS (SEARCHABLE 'Searchable'),
        birthyear integer OPTIONS (SEARCHABLE 'Searchable'),
        name string OPTIONS (SEARCHABLE 'Searchable'),
        state string OPTIONS (SEARCHABLE 'Searchable'),
        gender string OPTIONS (SEARCHABLE 'Searchable'),
       totalcount integer OPTIONS (SEARCHABLE 'Searchable')
    ) OPTIONS (UPDATABLE TRUE);
```

13. この VDB 内の全モデルで "babynames" という名前の VDB を構築し、右クリックでサーバーにデプロイします。

上記で述べた他のオプションは、動的 VDB を構築することです。(通常の VDB を構築する場合は、この手順を省略。)

1. テキストエディターで standalone.xml を開きます。
2. この XML を "resource-adapter" セクションに追加します。

```
<!-- Data Source for Flat File -->
       <resource-adapter id="file-ds">
                        <module slot="main"
id="org.jboss.teiid.resource-adapter.file"/>
                        <transaction-
support>NoTransaction</transaction-support>
                        <connection-definitions>
                            <connection-definition class-
name="org.teiid.resource.adapter.file.FileManagedConnectionFactory"
jndi-name="java:/file-ds" enabled="true" pool-name="file-ds">
                                <config-property
name="ParentDirectory">
                                    /babynames
                                </config-property>
                            </connection-definition>
                        </connection-definitions>
                    </resource-adapter>
```

```
<!-- Data Source for Solr -->
                    <resource-adapter id="solr-ds">
                        <module slot="main"
id="org.jboss.teiid.resource-adapter.solr"/>
                        <transaction-
support>XATransaction</transaction-support>
                        <connection-definitions>
                            <connection-definition class-
name="org.teiid.resource.adapter.solr.SolrManagedConnectionFactory"
jndi-name="java:/solr-ds" enabled="true" pool-name="solr-ds">
                                <config-property name="CoreName">
                                    babies
                                </config-property>
                                <config-property name="url">

http://localhost:8983/solr/babies
                                </config-property>
                            </connection-definition>
                        </connection-definitions>
                    </resource-adapter>
```

3. Red Hat JBoss Data Virtualization を起動します。
4. 以下の動的 VDB ファイルを作成し、"babynames-vdb.xml" という名前で保存します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb name="babynames" version="1">
    <model name="solr">
        <source name="solr" translator-name="solr" connection-jndi-
name="java:/solr-ds"/>
    </model>
    <model name="file_source">
        <source name="file" translator-name="file" connection-jndi-
name="java:/file-ds"/>
    </model>

    <model name = "file" visible = "true" type = "VIRTUAL" >
         <metadata type = "DDL"><![CDATA[
              CREATE VIEW babies (
                name varchar,
                state varchar,
                gender varchar,
                birthyear integer,
                totalcount integer
                ) AS
                  SELECT A.name, A.state, A.gender, A.birthyear,
A.totalcount FROM
           (EXEC file_source.getTextFiles('VA.TXT')) AS f,
                  TEXTTABLE(f.file COLUMNS state string, gender
string, birthyear integer, name string, totalcount integer) AS A;
      ]]
        </metadata>
    </model>
</vdb>
```

5. コンソールまたは CLI を使用して VDB をデプロイします。

以下のステップを実行して統合をテストします。

1. コンソールまたはログファイルをチェックするか、http://localhost:9990/console/App.html#vdb-runtime の Web コンソールを使用して VDB がアクティブであることを確認します。
2. JBDS Data Tools や Squirrel といったツールを利用してテストクエリを実行します。最初にファイルソースをチェックしてから、Solr ソースをチェックします (最初は Solr に何もないはずです)。

```
    select * from file.babies
```

```
    select * from solr.babies
```

3. ファイルソースのレコードを Solr に挿入します。

```
    insert into solr.babies (id, name, state, gender, birthyear,
totalcount)
      select f.name, f.name, f.state, f.gender, f.birthyear,
f.totalcount from file.babies as f
```

   131638 行に影響があることが分かるはずです。

4. これで以下のような SQL コマンドが発行できます。

```
 select * from solr.babies
select * from solr.babies where name = 'Mary'
select * from solr.babies where name like '*ary'
select * from solr.babies where birthyear > 1910 and birthyear < 1920
```

5. RDBMS、Salesforce、Web Service などの他のモデルがある場合は、これらソースと Solr の間で表をつなげることができます。また、Solr ソースに INESRT/UPDATE/DELETE のような SQL クエリを発行して、ストア内のドキュメントを管理することもできます。

4.9.16. Web サービスデータソース

Web サービスデータソースは、インストール時に EAP にデプロイされた Data Virtualization 固有の JCA コネクターを使用します。ファイルデータソースを作成するには多くの方法があります (CLI、AdminShell、管理コンソールなどを使用)。以下の例では、スタンドアロンモードとドメインモードの両方で動作する CLI ツールを使用します。

サーバーに接続したら、CLI を使用して以下のコマンドを実行します。また、正しいエンドポイントと他のプロパティーを提供し、以下の "connection-definitions" コマンドをコピーしてコネクターで必要なプロパティーを追加します。さらに、VDB で使用した JNDI 名に一致するように JNDI 名を編集します。

```
batch
/subsystem=resource-adapters/resource-adapter=webservice/connection-
definitions=wsDS:add(jndi-name=java:/wsDS, class-
name=org.teiid.resource.adapter.ws.WSManagedConnectionFactory, enabled=true,
use-java-context=true)
/subsystem=resource-adapters/resource-adapter=webservice/connection-
definitions=wsDS/config-properties=EndPoint:add(value={end_point})
/subsystem=resource-adapters/resource-adapter=webservice:activate
runbatch
```

この Web サービスコネクターでサポートされたすべてのプロパティーを見つけるには、CLI で次のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=webservice)
```

Web サービスデータソースは Wsdl プロパティーを使用した WSDL の指定をサポートします。Wsdl プロパティーが設定された場合は、 ServiceName、EndPointName、および NamespaceUri プロパティーも設定する必要があります。Wsdl プロパティーは URL またはファイルの場所、あるいは使用する WSDL になります。サーバーを再起動する必要があります。

表4.24 レジストリープロパティー

プロパティー	アプリケーション	必要性	デフォルト	説明
EndPoint	HTTP および SOAP	True	該当なし	HTTP の場合は URL、SOAP の場合はサービスエンドポイント。
SecurityType	HTTP および SOAP	false	なし	Web サービスで使用する認証タイプ。許可される値は "None"、"HTTPBasic"、"WSSecurity"、および "Kerberos" です。
AuthUserName	HTTP および SOAP	false	該当なし	HTTPBasic および WsSecurity で使用される認証の名前の値。
AuthPassword	HTTP および SOAP	false	該当なし	HTTPBasic および WsSecurity で使用される認証のパスワードの値。
ConfigFile	HTTP および SOAP	False	該当なし	CXF クライアント設定ファイルまたは URL。
ConfigName	HTTP および SOAP	False	該当なし	このプロパティーは非推奨であることに注意してください。
EndPointName	HTTP および SOAP	False	Teiid	この接続で使用するエンドポイント QName のローカル部分。CXF ファイルで定義されたものに一致する必要があります。
ServiceName	SOAP	False	該当なし	この接続で使用するサービス QName のローカル部分。
NamespaceUri	SOAP。	False	http://teiid.org	この接続で使用するサービス QName のネームスペース URI。

プロパティー	アプリケーション	必要性	デフォルト	説明
RequestTimeout	HTTP および SOAP	False	該当なし	要求のタイムアウト。
ConnectTimeout	HTTP および SOAP	False	該当なし	接続のタイムアウト。
Wsdl	SOAP。	False	該当なし	Web サービス用 WSDL ファイルまたは URL。

各 Web サービスデータソースでは特定の CXF 設定ファイルとポート設定を選択できます。ConfigFile 設定プロパティーは CXF Bus の Spring XML 設定ファイルと接続で使用するポート設定を指定します。設定ファイルが指定された場合は、システムデフォルト設定が使用されます。

このデータソースでは、ポート設定を 1 つだけ使用できます。ConfigName プロパティーにより、使用するポート QName のローカル名を明示的に設定できます。設定ファイルの QName のネームスペース URI はデータソースの WSDL/ネームスペース設定に一致するか、デフォルト値である http://teiid.org を使用する必要があります。CXF 設定ファイルの使用例については、CXF ドキュメンテーションと WS-Security やロギングなどに関する項を参照してください。

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:http-
conf="http://cxf.apache.org/transports/http/configuration"

xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
        http://cxf.apache.org/schemas/configuration/http-conf.xsd
        http://www.springframework.org/schema/beans
        >

    <http-conf:conduit name="{http://teiid.org}configName.http-conduit">
        <http-conf:client ConnectionTimeout="120000"
ReceiveTimeout="240000"/>
    </http-conf:conduit>
</beans>
```

コンジット名 {http://teiid.org}configName.http-conduit, the namespace, {http://teiid.org} では、ネームスペースデータソースプロパティーで設定できます。通常、これは、wsdl 設定を指定する場合のみ行う必要があります。ローカル名は .http-conduit の後に続き、configName 設定に基づきます (デフォルト値は teiid)。

注記

タイムアウトを設定するには Spring 設定を使用する必要がありません。ConnectionTimeout および ReceiveTimeout は、それぞれリソースアダプターの connectTimeout プロパティーと requestTimeout プロパティーを使用して設定できます。

WS-Security の使用を有効にするには、SecurityType を WSSecurity に設定する必要があります。この場合、Teiid は WSDL が使用するサービスを定義することを期待しません。したがって、Spring XML 設定ファイルは必要ではなく、代わりに関連するすべてのポリシー設定を含む必要があります。一般的な設定の場合と同様に、各データソースは、使用する単一のポート設定のみの指定に限定されます: **`batch /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS:add(jndi-name=java:/wsDS, class-name=org.teiid.resource.adapter.ws.WSManagedConnectionFactory, enabled=true, use-java-context=true) /subsystem=resource-adapters/resource-`**

adapter=webservice/connection-definitions=wsDS/config-properties=ConfigFile:add(value=${jboss.server.home.dir}/standalone/configuration/xxx-jbossws-cxf.xml) /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS/config-properties=ConfigName:add(value=port_x) /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS/config-properties=SecurityType:add(value=WSSecurity) /subsystem=resource-adapters/resource-adapter=webservice:activate runbatch

以下に、タイムスタンプを SOAP ヘッダーに追加する **xxx-jbossws-cxf.xml** ファイルを示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://cxf.apache.org/jaxws
          http://cxf.apache.org/schemas/jaxws.xsd">

    <jaxws:client name="{http://teiid.org}port_x"
        createdFromAPI="true">
        <jaxws:outInterceptors>
            <bean/>
            <ref bean="Timestamp_Request"/>
        </jaxws:outInterceptors>
    </jaxws:client>

    <bean

        id="Timestamp_Request">
        <constructor-arg>
            <map>
                <entry key="action" value="Timestamp"/>
            <map>
        </constructor-arg>
    </bean>

</beans>
```

QName {http://teiid.org}port_x により、クライアントポート設定がデータソースインスタンスに一致することに注意してください。ここで、ネームスペースはネームスペース設定またはデフォルトの http://teiid.org に一致します。この設定には、異なるローカル名を持つ他のポート設定を含めることができます。

WS-Security Kerberos は、WSDL プロパティーが resource-adapter 接続設定で定義され、WSDL ベースプロシージャーが使用されいている場合のみサポートされます。WSDL ファイルには、WS-Policy セクションを含める必要があります。これにより、WS-Policy セクションはエンドポイントで正しく解釈され、適用されます。以下に、CXF 設定の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:http="http://cxf.apache.org/transports/http/configuration"
    xmlns:jaxws="http://cxf.apache.org/jaxws"
    xmlns:cxf="http://cxf.apache.org/core"
    xmlns:p="http://cxf.apache.org/policy"
    xmlns:sec="http://cxf.apache.org/configuration/security"
```

```
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/jaxws
http://cxf.apache.org/schemas/jaxws.xsd
http://cxf.apache.org/transports/http/configuration
http://cxf.apache.org/schemas/configuration/http-conf.xsd
http://cxf.apache.org/configuration/security
http://cxf.apache.org/schemas/configuration/security.xsd
http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd
http://cxf.apache.org/policy http://cxf.apache.org/schemas/policy.xsd">
    <bean
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigure
r"/>
    <cxf:bus>
        <cxf:features>
            <p:policies/>
            <cxf:logging/>
        </cxf:features>
    </cxf:bus>

    <jaxws:client name="
{http://webservices.samples.jboss.org/}HelloWorldPort"
createdFromAPI="true">
        <jaxws:properties>
            <entry key="ws-security.kerberos.client">
                <bean
class="org.apache.cxf.ws.security.kerberos.KerberosClient">
                    <constructor-arg ref="cxf"/>
                    <property name="contextName" value="alice"/>
                    <property name="serviceName"
value="bob@service.example.com"/>
                </bean>
            </entry>
        </jaxws:properties>
    </jaxws:client>
</beans>
```

次に、以下のように **`standalone-teiid.xml`** ファイルの 'security' サブシステムで security-domain を設定します。

```
<security-domain name="alice" cache-type="default">
    <authentication>
        <login-module code="Kerberos" flag="required">
            <module-option name="storeKey" value="true"/>
            <module-option name="useKeyTab" value="true"/>
            <module-option name="keyTab" value="/home/alice/alice.keytab"/>
            <module-option name="principal" value="alice@EXAMPLE.COM"/>
            <module-option name="doNotPrompt" value="true"/>
            <module-option name="debug" value="true"/>
            <module-option name="refreshKrb5Config" value="true"/>
        </login-module>
    </authentication>
</security-domain>
```

また、CXF 設定プロパティーは、特定またはすべてのポートに対する要求および応答のロギングを制御するために使用できます。ロギング (有効な場合) は、org.apache.cxf.interceptor コンテキストに対する INFO レ

ベルで実行されます: **batch /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS:add(jndi-name=java:/wsDS, class-name=org.teiid.resource.adapter.ws.WSManagedConnectionFactory, enabled=true, use-java-context=true) /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS/config-properties=ConfigFile:add(value=${jboss.server.home.dir}/standalone/configuration/xxx-jbosssws-cxf.xml) /subsystem=resource-adapters/resource-adapter=webservice/connection-definitions=wsDS/config-properties=ConfigName:add(value=port_x) /subsystem=resource-adapters/resource-adapter=webservice:activate runbatch**

対応する **xxx-jbosssws-cxf.xml** ファイルは以下のとおりです。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://cxf.apache.org/jaxws
          http://cxf.apache.org/schemas/jaxws.xsd">

    <jaxws:client name="{http://teiid.org}port_y"
        createdFromAPI="true">
        <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature"/>
        </jaxws:features>
    </jaxws:client>

</beans>
```

CXF 設定プロパティーは、HTTP トランスポートの低レベルの側面を制御するために使用することもできます。

```


<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"

xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration

http://cxf.apache.org/schemas/configuration/http-conf.xsd
                           http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

  <http-conf:conduit name="{http://teiid.org}port_z.http-conduit">
     <!-- WARNING ! disableCNcheck=true should NOT be used in production -->
     <http-conf:tlsClientParameters disableCNcheck="true" />

  </http-conf:conduit>
</beans>
```

HTTPS を使用するには、CXF ファイルを以下のように設定します (HTTPBasic やkerberos なども設定できます)。

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sec="http://cxf.apache.org/configuration/security"
    xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
    xmlns:jaxws="http://java.sun.com/xml/ns/jaxws"
    xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
http://cxf.apache.org/schemas/configuration/http-conf.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://cxf.apache.org/configuration/security
http://cxf.apache.org/schemas/configuration/security.xsd">

    <http-conf:conduit name="*.http-conduit">
        <http-conf:client ConnectionTimeout="120000"
ReceiveTimeout="240000"/>
        <http-conf:tlsClientParameters secureSocketProtocol="SSL">
          <sec:trustManagers>
            <sec:keyStore type="JKS" password="changeit"
file="/path/to/truststore.jks"/>
          </sec:trustManagers>
        </http-conf:tlsClientParameters>
    </http-conf:conduit>
</beans>
```

kerberos サポートは、http://cxf.apache.org/docs/client-http-transport-including-ssl-support.html#ClientHTTPTransport%28includingSSLsupport%29-SpnegoAuthentication%28Kerberos%29 で定義された SPNEGO に基づきます。kerberos サポートには 2 つのタイプがあり、最初のタイプは Negotiation です。この設定では、トークンをネゴシエートするために REST サービスが Kerberos JAAS ドメイン (web サービスにアクセスするために使用されます) で指定されます。この最初のタイプの場合は、以下のように **standalone.xml** ファイルでセキュリティードメインを作成します。

```
<security-domain name="MY_REALM" cache-type="default">
     <authentication>
         <login-module code="Kerberos" flag="required">
             <module-option name="storeKey" value="true"/>

             <module-option name="useKeyTab" value="true"/>
             <module-option name="keyTab"
value="/home/username/service.keytab"/>
             <module-option name="principal"
value="host/testserver@MY_REALM"/>

             <module-option name="doNotPrompt" value="true"/>
             <module-option name="debug" value="false"/>
         </login-module>
     </authentication>
</security-domain>
```

以下のように、**jboss-cxf-xxx.xml** ファイルを設定します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:sec="http://cxf.apache.org/configuration/security"
        xmlns:http-
conf="http://cxf.apache.org/transports/http/configuration"
```

```
xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
http://cxf.apache.org/schemas/configuration/http-conf.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://cxf.apache.org/configuration/security
http://cxf.apache.org/schemas/configuration/security.xsd">

       <http-conf:conduit name="*.http-conduit">
           <http-conf:authorization>
               <sec:AuthorizationType>Negotiate</sec:AuthorizationType>
               <sec:Authorization>MY_REALM</sec:Authorization>
           </http-conf:authorization>
       </http-conf:conduit>
</beans>
```

以下のように、リソースアダプターの作成で、これらのプロパティーを定義する必要があります。

```
<config-property name="ConfigFile">path/to/jboss-cxf-xxxx.xml</config-
property>
<config-property name="ConfigName">test</config-property>
```

重要

上記の設定では、"ConfigName" の値が指定されますが、JAX-RS クライアントで現在使用されている cxf フレームワークではそれを使用するオプションが提供されません。このため、このリソースアダプターのすべての HTTP 通信に適用する "*.http-conduit" を使用します。

2 つ目の方法は Delegation です。ユーザーが JDBC/ODBC を使用して Kerberos ですでに Teiidor にログインしている場合や、web 階層で SPNEGO を使用し、Teiid へのパススルー認証を使用する場合は、Kerberos のために新しいトークンをネゴシエートする必要はありません。システムは既存のトークンを委任できます。委任を設定するには、"negotiation" と jboss-cxf-xxx.xml ファイルで定義されたようにセキュリティードメインをセットアップします。ただし、 jboss-cxf-xxx.xml ファイルから次の行を削除してください (新しいトークンがネゴシエートされません)。

```
<sec:Authorization>MY_REALM</sec:Authorization>
```

Web サービスリソースアダプターの作成で以下のプロパティーを追加します。最初の部分では、"kerberos" セキュリティーを使用することを定義し、2 つ目の部分では、データソースで使用するセキュリティードメインを定義します。この場合は、ログインしているユーザーにパススルーするセキュリティードメインを使用します。

```
<config-property name="SecurityType">Kerberos</config-property>
<security>
     <security-domain>passthrough-security</security-domain>
</security>
```

"passthrough-security" セキュリティードメインを設定するには、以下のように "security" サブシステム設定を追加する必要があります。

```
<security-domain name="passthrough-security" cache-type="default">
    <authentication>
```

```
        <login-module code="org.teiid.jboss.PassthroughIdentityLoginModule"
flag="required" module="org.jboss.teiid">
            <module-option name="username" value="guest"/>
            <module-option name="password" value="guest"/>
        </login-module>
    </authentication>
</security-domain>
```

注記

ユーザー名とパスワードはオプションです。コンテキストで認証済みサブジェクトが利用できない場合は、これらにより、単純な静的ユーザーを簡単に作成できるようになります。ただし、サブジェクトが必要な kerberos トークンを持っていないため、このユーザーは kerberos 認証を使用しません。

4.10. 管理コンソールを使用したデプロイ済み仮想データベースの管理

4.10.1. 管理コンソールを使用したデプロイ済み仮想データベースの管理

管理者は、**Virtual Databases** パネル内からデプロイ済み仮想データベースを管理できます。

4.10.2. 仮想データベースパネルを開く

1. JBoss 管理コンソールにログインします。
2. **Runtime** タブを選択します。
3. ナビゲーションツリーから、**Status** → **Subsystems** → **Virtual Databases** を選択します。

4.11. リソースアダプター

4.11.1. JBoss Data Virtualization のリソースアダプター

JBoss Data Virtualization は JDBC データソースを除くサポート対象の各データソースに対して JCA アダプターを提供します。サーバー設定ファイルで指定されたように、以下のものがリソースアダプター識別子になります。

- ファイルアダプター - **file**
- Google スプレッドシートアダプター - **google**
- Red Hat JBoss Data Grid (6.1 & 6.2) アダプター - **infinispan**
- LDAP アダプター - **ldap**
- Salesforce アダプター - **salesforce**
- Web サービスアダプター - **webservice**
- Mongo DB アダプター (テクニカルプレビュー) - **mongodb**

注記

JDBC トランスレーターのリソースアダプターは、デフォルトで JBoss EAP により提供されます。

4.11.2. リソースアダプターの設定

VDB で必要なデータソースのリソースアダプターを設定するには、管理コンソール、AdminShell、または管理 CLI を使用します (スタンドアロンモードでは、サーバー設定ファイルを直接編集します)。

注記

ドメインモードでは、管理 CLI、管理コンソール、または AdminShell を使用してデータソースを設定する必要があります。管理コンソールを使用してリソースアダプターを設定する方法の例については、JBoss Enterprise Application Platform 『管理および設定ガイド』を参照してください。

重要

CLI スクリプトまたは AdminShell のいずれかを使用してリソースアダプターを設定する場合、既存のリソースアダプターの名前を付けることはできません (Teiid Designer はリソースアダプターに一意名を作成してこれに対応します)。

この理由を以下に示します。

- 同じ名前を持つリソースアダプターが追加されると、サーバーの再起動が必要になります。これは、Red Hat JBoss EAP の JCA システムの設計によるもので、同じ名前を持つリソースアダプターの複数のインスタンスはサーバーの起動後のみ認識されます。
- 名前を指定してリソースアダプターを削除すると、同じ名前のすべてのインスタンスが削除されます。

4.11.3. 設定例

設定例 (サーバー設定ファイルに含まれます) は、***EAP_HOME*/docs/teiid/datasources/** ディレクトリーにあります。

注記

この設定では、使用しているインストールに応じてファイルパスとプロパティーを調整する必要があります。JNDI 名は VDB で使用されているのと同じ JNDI 名である必要があります。

4.11.4. リソースアダプタープロパティー

設定プロパティーの完全なリストについては、管理 CLI 内で以下のコマンドを実行します。

```
/subsystem=teiid:read-rar-description(rar-name=ADAPTER_ID)
```

ADAPTER_ID は、サーバー設定ファイルで指定されたリソースアダプターの識別子を表します。

4.11.5. CLI スクリプトを使用してリソースアダプターを設定

リソースアダプターを設定するには、この方法が推奨されます。

JBoss Data Virtualization は、提供される各リソースアダプターを設定する管理 CLI サンプルスクリプトを提供します。特定のリソースアダプターのスクリプトは、***EAP_HOME*/docs/teiid/datasources/*DATASOURCE*** ディレクトリーにあります。

管理 CLI を使用してリソースアダプターを設定する場合は、プロパティーファイルを提供する必要があります。サンプルプロパティーファイルは、サンプルスクリプトと同じディレクトリーで提供されます。

これらのスクリプトを実行するには、以下のコマンドを実行します (JBoss Data Virtualization が実行された後)。

```
./EAP_HOME/bin/jboss-cli.sh --connect --
file=EAP_HOME/docs/teiid/datasources/DATASOURCE/SCRIPT.cli --
properties=EAP_HOME/docs/teiid/datasources/DATASOURCE/SCRIPT.properties
```

管理 CLI の詳細については、JBoss Enterprise Application Platform 『管理および設定ガイド』を参照してください。

注記

これらのスクリプトでは、使用しているインストールに応じてファイルパスとプロパティーを調整する必要があります。JNDI 名は VDB で使用されているのと同じ JNDI 名である必要があります。

4.11.6. ファイルアダプタープロパティー

以下の表では、ファイルリソースアダプターに設定できる設定プロパティーについて説明しています。

設定プロパティー	例	説明
ParentDirectory		データファイルが格納されるディレクトリー。
FileMapping	file1.txt=fileX.txt,file2.txt=fileY.txt	特定の相対パス (大文字と小文字を区別) を代替の場所にリダイレクトするよう FileMapping を設定します。文字列値は、key=value(,key=value)* という形式でマップを指定します。これはオプションです。
AllowParentPaths	true	パスで '..' を許可しない場合は AllowParentPaths を false に設定します。これにより、親ディレクトリーに含まれないファイルが要求されることが回避されます。これはオプションです。

4.11.7. Google スプレッドシートリソースアダプタープロパティー

4.11.7.1. Google スプレッドシートリソースアダプタープロパティー

以下の表では、Google スプレッドシートリソースアダプターに対して設定できる設定プロパティーについて説明しています。

設定プロパティー	説明
AuthMethod	これは、Google にアクセスするために使用する認証方法です。設定が **OAuth2** の場合は、**RefreshToken** を提供する必要があります。
RefreshToken	**AuthMethod=OAuth2** の場合のみ必須です。
Username	Google アカウントのユーザー名。**AuthMethod=ClientLogin**の場合のみ必須です。
Password	Google アカウントのパスワード。**AuthMethod=ClientLogin** の場合のみ必須です。
SpreadsheetName	このリソースアダプターが接続するスプレッドシートの名前。必須です。
BatchSize	一度にフェッチできる行の最大数。デフォルト値は 4096 です。

4.11.7.2. OAuth 更新トークンの取得

OAuth 認証で Google スプレッドシートリソースアダプターを使用する場合は、 OAuth 更新トークンを取得する必要があります。

手順4.9 OAuth 更新トークンの取得

1. **承認コードを取得する**

   次のリンクをクリックします: Get Authorization Code

   Allow access をクリックして、**Teiid Google Connector** がスプレッドシートが存在する Google アカウントにアクセスすることを許可します。

2. **更新トークンを取得する**

   前の手順の承認コードを以下の POST 要求の **code** フィールドにコピーし、コマンドラインから実行します。

   ```
   curl \--data-urlencode code=AUTH_CODE \
   --data-urlencode client_id=217138521084.apps.googleusercontent.com \
   --data-urlencode client_secret=gXQ6-lOkEjE1lVcz7giB4Poy \
   --data-urlencode redirect_uri=urn:ietf:wg:oauth:2.0:oob \
   --data-urlencode grant_type=authorization_code
   https://accounts.google.com/o/oauth2/token
   ```

   更新トークンが応答に含まれます。

4.11.8. JBoss Data Grid リソースアダプタープロパティー

JBoss Data Grid (JDG) リソースアダプターは、以下のキャッシュモードをサポートするよう設定できます。

キャッシュタイプ	キャッシュの取得手段
リモートキャッシュ	JNDI の使用
リモートキャッシュ	指定された 1 つ以上のホストとポートの組み合わせ
リモートキャッシュ	指定された HotRod クライアントプロパティーファイルの参照

以下の表は、リソースアダプターに設定できる設定プロパティーを示しています。

プロパティー名	必須	プロパティーテンプレート	description
CacheTypeMap	Y	cacheName:className[;pkFieldName] [,cacheName:className[;pkFieldName]..]	このプロパティーは、ルート Java クラス名をキャッシュにマップし、プライマリーキーを識別します。
module	N		このプロパティーは、CacheTypeMap で定義されたキャッシュクラスを含む JBoss EAP モジュールを指定します。
CacheJndiName	N		これは、CacheContainer を見つけるために使用する JNDI 名です。
RemoteServerList	N	host:port[;host:portâ¦.]	このプロパティーは、CacheTypeMap で定義されたキャッシュにアクセスするために一緒にクラスタ化するホスト (およびポート) を指定します。
ConfigurationFileNameForLocalCache	N		これは、ローカルキャッシュを設定するための XML 設定ファイルです。
HotRodClientPropertiesFile	N		これは、リモートキャッシュに対する接続を設定するための HotRod プロパティーファイルです。

注記

Teiid Designer を使用して pojo のビューのリバースエンジニアリングを行うと、BigDecimal データタイプがビューで定義されます。シリアル化に使用される Google Protobuf の場合、残念ながら複雑なデータタイプは C または C++ に変換することができません。このため、プリミティブなデータタイプのみを使用することが推奨されます。(複雑なデータタイプを含むビューを実体化する場合、もしくは複雑なデータタイプを含む POJO を格納している既存の JDG キャッシュがある場合、この状況が発生します。)

protobuffer は BigDecimal を直接サポートしないので、以下の 3 つのオプションがあります。

1. すべてのプリミティブデータタイプを使用する。
2. 変換を処理するマーシャラーを実装する。その場合、.proto ファイルも作成する必要があります。(ファイル作成については、Red Hat JBoss Data Grid を参照してください。)
3. BigdDecimal を文字列に変換するビューを作成し、そのビューを実体化する。

4.11.9. JDG HotRod トランスレーター

Infinispan HotRod トランスレーターはタイプ ispn-hotrod で知られており、java オブジェクトをリモートの Red Hat JBoss Data Grid Cache からシリアル化のために Google Protobuf を使用して Hot Rod クライアント経由で読み取ることができます。これにより Red Hat JBoss Data Virtualization が JDG DSL を使用してリモートキャッシュにクエリできるようになります。

このトランスレーターはキャッシュからオブジェクトを取得し、行と列に変換してキャッシュに書き込めるようにし、外部の実体化を使用してクエリのパフォーマンス改善を可能にします。

コネクターには以下の機能があります。

- Compare Criteria - EQ
- Compare Criteria Ordered - LT, GT, LE, GE - supportsCompareCriteriaOrdered トランスレーター上書きが true に設定されている場合。デフォルトは false です。.
- And/Or Criteria
- In Criteria
- Like Criteria
- Order By
- INSERT, UPDATE, DELETE (トランザクション以外)

以下のものは処理向けに JDG へプッシュされませんが、Red Hat JBoss Data Virtualization 内で処理されます。

- Not (NE)
- IsNull

以下の制限があります。

- 'Not' のサポートは無効になっています。
- boolean data type: 挿入の値が指定されていない場合、または protobuf 定義ファイルでデフォルト値が定義されていない場合は JDG が例外をスローします。

- char data type: Protobuf データタイプでサポートされているタイプではありません (https://developers.google.com/protocol-buffers/docs/proto#scalar)。protobuf マーシャラーで変換を処理するか、char のデータタイプで Red Hat JBoss Data Virtualization ビューを作成する必要があります。
- 1-to-Many が現在サポートしているのは Collection または Array のみで、Maps はサポートされていません。
- 書き込みトランザクションは、Hot Rod クライアント使用時には JDG でサポートされません。

pojo クラスは、キャッシュにデータ保存する際に使用するオブジェクトです。以下を実行するために構築されます。

- キャッシュがインデックス対応となっていることを活用するには、クラスに注釈を付けます。JDG Indexing With Protobuf Annotations (https://access.redhat.com/documentation/en-US/Red_Hat_JBoss_Data_Grid/6.6/html-single/Infinispan_Query_Guide/index.html#Custom_Fields_Indexing_with_Protobuf) を参照してください。
- クラスは jar にパッケージ化して、モジュールとしてデプロイできるようにします。

以下に例を示します。

```
public class Person {

  @ProtoField(number = 2, required = true)
  public String name;
  @ProtoField(number = 1, required = true)
  public int id;
  @ProtoField(number = 3)
  public String email;
  private List<PhoneNumber> phones;

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public int getId() {
     return id;
  }

  public void setId(int id) {
     this.id = id;
  }

  public String getEmail() {
     return email;
  }

  public void setEmail(String email) {
     this.email = email;
  }

  public List<PhoneNumber> getPhones() {
```

```
      return phones;
   }

   public void setPhones(List<PhoneNumber> phones) {
      this.phones = phones;
   }
}
```

pojo の使用を設定するには、pojo jar をモジュールとして Red Hat JBoss EAP サーバーにデプロイし、 -vdb.xml 内で "lib" プロパティーを定義して適切なモジュール名を割り当てます。これには、以下のテンプレートを使用できます。

```
     <property name ="lib" value ="{pojo_module_name}"></property>
```

キャッシュ内のオブジェクトを表記するメタデータの定義には、いくつかのオプションがあります。

- Teiid Designer 内の Teiid Connection Importer を使用して、オブジェクトキャッシュに基づく物理ソースモデルを作成することが推奨されます。
- Teiid Designer を使用してオブジェクトキャッシュに基づく物理ソースモデルを手動で作成するには、以下の定義要件を使用します。
- 使用するデータソースを定義するだけのシンプルな VDB は以下のようになります。

  ```
      <model name="People" type="Physical">
          <property name="importer.useFullSchemaName" value="false"/>

          <source name="infinispan-hotrod-connector" translator-name="ispn-
  hotrod" connection-jndi-name="java:/infinispanRemoteDSL" />
      </model>
  ```

  メタデータは、キャンセル内の定義済みオブジェクトのリバースエンジニアリングで解決されます。これは物理ソースモデル構築で Teiid Designer Teiid Connection Importer を使用している場合に便利です。

- DDL を使用してメタデータを定義することもできます。

  コンテナークラス PhoneNumber も関係を定義する外部キーの例として表示されることに注意してください。

  ```
  <vdb name="PeopleVDB" version="1">
      <model name="People" visible="true">
          <property name="importer.useFullSchemaName" value="false"/>

          <source name="infinispan-cache-dsl-connector" translator-
  name="ispn-hotrod" connection-jndi-name="java:/infinispanRemote" />

          <metadata type="DDL"><![CDATA[

   CREATE FOREIGN TABLE Person (
   PersonObject object OPTIONS (NAMEINSOURCE 'this', UPDATABLE FALSE,
  SEARCHABLE 'Unsearchable', NATIVE_TYPE 'java.lang.Object'),
   id integer NOT NULL OPTIONS (SEARCHABLE 'Searchable', NATIVE_TYPE 'int'),
   name string OPTIONS (SEARCHABLE 'Searchable', NATIVE_TYPE
  'java.lang.String'),
  ```

```
 email string OPTIONS (SEARCHABLE 'Searchable', NATIVE_TYPE
'java.lang.String'),
 CONSTRAINT PK_ID PRIMARY KEY(id)
) OPTIONS (NAMEINSOURCE 'PersonsCache', UPDATABLE TRUE);

CREATE FOREIGN TABLE PhoneNumber (
 number string OPTIONS (NAMEINSOURCE 'phone.number', SEARCHABLE
'Searchable', NATIVE_TYPE 'java.lang.String'),
 type string OPTIONS (NAMEINSOURCE 'phone.type', SEARCHABLE 'Searchable',
NATIVE_TYPE 'java.lang.String'),
 id integer NOT NULL OPTIONS (SELECTABLE FALSE, UPDATABLE FALSE,
SEARCHABLE 'Searchable', NATIVE_TYPE 'int'),
 CONSTRAINT FK_PERSON FOREIGN KEY(id) REFERENCES Person (id) OPTIONS
(NAMEINSOURCE 'phones')
) OPTIONS (NAMEINSOURCE 'PersonsCache', UPDATABLE TRUE);

          ]> </metadata>
    </model>

</vdb>
```

- キャッシュ内の Google 登録してあるクラスには、対応する表が作成されます。
- root クラスの表では、クラス内の属性をマッピングしたプライマリーキーを定義する必要があります。

  クラス内の属性のデータタイプは、JDG キャッシュキーのデータタイプと一致する必要があります。
- 表の "name in source" (NIS) は、この表/クラスが保存される JDG キャッシュの名前になります。
- 表のコラムは、定義済みクラスに対応する Google protobuf 定義から作成されます。
- コラムの protobuf 定義がインデックス化されている、または pojo クラスが属性/メソッドの注釈付きの場合は、コラムは SEARCHABLE として特定されます。
- 反復可能として定義される属性 (コレクション、アレイなど) またはコンテナークラスは、1-to-* 関係としてサポートされ、対応する定義済みクラスを持つことになります (これらが検索される場合)。
- 1-to-* 関係クラスには root クラス/表にマッピングする外部キーが必要になります。ここでは、外部キーのソース内の名前は、それらの子オブジェクトにアクセスするための root クラスメソッド名になります。これはクラスメソッドであって、Google protobuf 定義の参照ではないことに注意してください。
- コンテナー/子クラスは、NIS にピリオドが含まれる属性を持つことになります。これは、これが Google protobuf 定義と DSL クエリで使用されることが予想されるものにマッピングするためです。

このトランスレーターは、外部実体化のためのキャッシュ使用をサポートします。ただし、[Infinispan-HotRod resource-adapter] とトランスレーターで特定の設定変更が必要になります。

外部実体化は、BEFORE_LOAD_SCRIPT と AFTER_LOAD_SCRIPT でのネイティブクエリの使用で有効になります。トランスレーターの上書きは、ネイティブクエリを有効に設定する必要があります。

```
SupportsNativeQueries=true
```

以下のプロパティーを定義する必要があります。

表4.25 設定プロパティー

スクリプト	ネイティブクエリ	説明
teiid_rel:MATVIEW_BEFORE_LOAD_SCRIPT	truncate cache	staging cache として特定されたキャッシュを切り詰めます。
teiid_rel:MATVIEW_AFTER_LOAD_SCRIPT	swap cache names	キャッシュのエイリアスをスワップし、プラマリーキャッシュが最近読み込まれたキャッシュをポイントするようにします。

以下の例は、DDL の読み込みスクリプトを定義します。

```
..
"teiid_rel:MATVIEW_BEFORE_LOAD_SCRIPT" 'execute
StockMatCache.native(''truncate cache'');',
"teiid_rel:MATVIEW_LOAD_SCRIPT" 'insert into StockMatCache.Stock (productId,
symbol, price, companyName) SELECT  A.ID, S.symbol, S.price, A.COMPANY_NAME
FROM Stocks.StockPrices AS S, Accounts.PRODUCT AS A WHERE S.symbol =
A.SYMBOL',
"teiid_rel:MATVIEW_AFTER_LOAD_SCRIPT"  'execute StockMatCache.native(''swap
cache names'');',
```

RDBMS 内でのクエリ動作をシミュレートし、表の名前を変更するために、ネイティブクエリが使用されます。これは、JDG が現在キャッシュの名前変更をサポートしていないためです。ネイティブクエリが "staging" キャッシュのクリーンアップと、キャッシュエイリアスのスワップを開始します。

また、ネイティブクエリの実効は直接クエリ手順のサポートを通じて行われます。実行される手順は、ネイティブと呼ばれます。

警告

この機能はセキュリティーリスクがあるため、デフォルトではオフになっています。これは、ソースに対して誰かがコマンドを実行することを可能にします。この機能を有効にするには、SupportsDirectQueryProcedure と呼ばれる上書き実行プロパティーを true に設定します。

この JCA Resource トランスレーターの Infinispan-HotRod リソースアダプターを参照してください。JNDI、server list、または Hot Rod プロパティーでキャッシュコンテナーをルックアップするように設定できます。

4.11.10. LDAP アダプタープロパティー

以下の表は、LDAP リソースアダプターに設定できる設定プロパティーを示しています。

設定プロパティー	プロパティーテンプレート	説明
LdapUrl		LDAP ディレクトリー URL。このプロパティーは必須です。
LdapAdminUserDN	cn=???,ou=???,dc=???	LDAP 管理ユーザー DN。このプロパティーは必須です。
LdapAdminUserPassword		LDAP 管理パスワード。このプロパティーは必須です。
LdapTxnTimeoutInMillis		LDAP トランザクションタイムアウト (ミリ秒単位)。-1 = タイムアウトなし。このプロパティーは必須です。

4.11.11. Salesforce アダプタープロパティー

以下の表は、Salesforce リソースアダプターに設定できる設定プロパティーを示しています。

設定プロパティー	説明
URL	接続先の URL。
username	ユーザー名。
password	パスワード。
requestTimeout	これは、タイムアウトを設定するオプションのプロパティーであり、CXF 設定により指定することもできます。
connectTimeout	これは、タイムアウトを設定するオプションのプロパティーであり、CXF 設定により指定することもできます。
configFile	このプロパティーを使用して、SalesForce サービスの特定の設定を提供します。この設定には、ネームスペース "urn:partner.soap.sforce.com" で "SforceService" サービスの設定を含める必要があります。

4.11.12. Web サービスアダプタープロパティー

以下の表は、Web サービスリソースアダプターに設定できる設定プロパティーを示しています。

設定プロパティー	例	説明
EndPoint		Web サービス用エンドポイント URL
SecurityType	HTTPBasic	http 基本セキュリティーの使用
AuthUserName		http 基本セキュリティーの使用
AuthPassword		http 基本セキュリティーの使用
RequestTimeout		これは、タイムアウトを設定するオプションのプロパティーであり、CXF 設定により指定することもできます。
ConnectTimeout		これは、タイムアウトを設定するオプションのプロパティーであり、CXF 設定により指定することもできます。
ConfigFile		これらのプロパティーを使用して、このサービスの特定の CXF 設定を提供します。このファイルには、"EndPointName" プロパティーで定義された名前の設定を含める必要があります。
EndPointName	WebSVC	ConfigFile とともに使用します。これらのプロパティーは、このサービスの特定の CXF 設定を提供します。

4.11.13. MongoDB アダプタープロパティー

以下の表は、MongoDB リソースアダプターに設定できる設定プロパティーを示しています。

設定プロパティー	例	説明
RemoteServerList	localhost:27017	host:port[;host:port...]* という形式の MongoDB サーバーリスト
データベース		データベース名。
ユーザー名		このプロパティーをパスワードとともに使用してクレデンシャルを提供します。
パスワード		このプロパティーをユーザー名とともに使用してクレデンシャルを提供します。

第5章 バージョン管理

5.1. 仮想データベースバージョン管理

仮装データベース (VDB) バージョン管理を使用すると、VDB の複数のバージョンを同時にデプロイし、特定のシナリオで使用するバージョンを指定できます。

アプリケーションが JBoss Data Virtualization に接続した場合に、接続プロパティーを使用して VDB の必要なバージョンを設定できます (詳細については、『Red Hat JBoss Data Virtualization User Guide』を参照)。

特定のバージョンが設定された場合は、その VDB にのみ接続できます。バージョンが設定されてない場合は、適切なバージョンが見つかるまで、デプロイされた VDB が検索されます。この機能により、よりダイナミックな移行シナリオがサポートされます。

5.2. VDB バージョンの設定

バージョンの設定は、**`vdb.xml`** ファイル (ダイナミック VDB で有用) を使用するか、またはデプロイメントファイルの命名規則で指定することによって (***`VDBNAME.VERSION`*`.vdb`** など) 行えます。デプロイヤーは、適切なバージョン番号を選択します。すでに現在のデプロイメントに一致する VDB の名前とバージョンの組み合わせが存在する場合は、前の VDB への接続が終了し、キャッシュエントリがフラッシュされます。新しい接続は、新しい VDB に対して行われます。

5.3. 仮想データベース接続タイプ

VDB がデプロイされたら、接続タイプと呼ばれるプロパティーを設定できます。このプロパティーを設定することで、VDB への接続タイプを決定します。接続タイプは以下のいずれかになります。

- **NONE**: 新しい接続を許可しません。
- **BY_VERSION**: (デフォルト設定) では、バージョンが指定された場合、またはバージョンが最も初期の BY_VERSION VDB であり、ANY とマークされた VDB がない場合にのみ接続が許可されます。
- **ANY**: 指定されたバージョンを使用して、または使用せずに接続を許可します。

一部のアプリケーションだけを新しいバージョンの VDB に移行する場合は、BY_VERSION に設定します。これにより、新しいバージョンを認識するアプリケーションだけがそのバージョンを使用できます。

特定の複数のアプリケーションのみを現在の VDB バージョンに維持する場合は、現在の VDB をそのバージョンで参照する接続設定を更新する必要があります。新しくデプロイされた VDB の接続タイプは、ANY に 設定され、すべての新しい接続を新しいバージョンに対して行えるようになります。

このシナリオでロールバックを行う必要がある場合、新しくデプロイされる VDB の接続タイプは NONE または BY_VERSION に設定されます。

5.4. 管理 API を使用して VDB 接続タイプを設定

VDB 接続タイプは、管理 API パッケージ (**`org.teiid.adminapi`**) 内の Admin インターフェースにより提供された **`changeVDBConnectionType`** メソッドを使用して変更できます。

Red Hat JBoss Data Virtualization 向け Javadocs は Red Hat カスタマーポータルで利用できます。

第6章 CXF 設定

6.1. CXF 設定

設定で CXF を使用する場合は、データソース設定ファイルとポート設定を指定する必要があります。

ConfigFile プロパティー (リソースアダプターサブシステム用のサーバー設定ファイルで設定される) は、Spring XML 設定ファイルを指定します (設定ファイルが指定されない場合は、システムのデフォルト設定が使用されます)。

EndPointName プロパティー (同様にリソースアダプターサブシステム用のサーバー設定ファイルで設定される) は、ポート設定を指定します。これを必要なポート QName のローカル部分に設定します。

設定ファイルプロパティーは以下のコマンドで指定します。

```
/subsystem=resource-adapters/resource-adapter=webservice/connection-
definitions=wsDS/config-
properties=ConfigFile:add(value=${jboss.server.home.dir}/standalone/configur
ation/xxx-jbossws-cxf.xml)
```

6.2. Web サービスデータソース用 CXF の設定

前提条件

- Web サービスデータソースが設定されている必要があります。

手順6.1 Web サービスデータソース用 CXF の設定

1. **Web サービス CXF `ConfigFile` の指定**

   管理 CLI 内から以下のコマンドを実行してデータソースの CXF 設定ファイルを指定します。

   ```
   /subsystem=resource-adapters/resource-adapter=webservice/connection-
   definitions=wsDS/config-properties=ConfigFile:add(value=CONFIG-
   FILE.xml)
   ```

2. **Web サービス CXF `EndPointName` の指定**

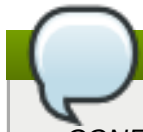
   管理 CLI 内から以下のコマンドを実行し、値としてポート QName (ローカル部分のみ) を使用してポート設定を指定します。

   ```
   /subsystem=resource-adapters/resource-adapter=webservice/connection-
   definitions=wsDS/config-properties=EndPointName:add(value=CONFIG-NAME)
   ```

3. **CXF 設定ファイルの作成/編集**

   ***EAP_HOME*/*MODE*/configuration/*CONFIG-FILE*.xml** 設定ファイルを開くか、作成します。

   ```
   <http-conf:conduit name="{NAMESPACE}CONFIG-NAME.http-conduit">
   ...
   </http-conf:conduit>
   ```



注記

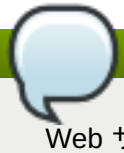
CONFIG-NAME は、上述したものと同じであり、デフォルト値は **teiid**です。*NAMESPACE* は設定ファイルの QName に対するネームスペース URI であり、データソースの WSDL/ネームスペース設定に一致する必要があります (または、デフォルトの "http://teiid.org" を使用します)。ネームスペースはネームスペースデータソースプロパティーを介して設定できます。通常、これは WSDL 設定を提供する場合のみ行う必要があります。

以下に、タイムアウトを設定する CXF ファイルの例を示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:http-
conf="http://cxf.apache.org/transports/http/configuration"

xsi:schemaLocation="http://cxf.apache.org/transports/http/configuratio
n
  http://cxf.apache.org/schemas/configuration/http-conf.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd">

  <http-conf:conduit name="{NAMESPACE}CONFIG-NAME.http-conduit">
    <http-conf:client ConnectionTimeout="120000"
ReceiveTimeout="240000"/>
  </http-conf:conduit>
</beans>
```



注記

Web サービスデータソースの場合、CXF 設定は非バイナリーコールにのみ適用できます。

6.3. Web サービスデータソース用 CXF の設定: WS-Security

前提条件

- Web サービスデータソースが設定され、**ConfigFile** プロパティーと **EndPointName** プロパティーが CXF に対して設定されている必要があります。

手順6.2 Web サービスデータソース用 CXF の設定: WS-Security

1. **CXF SecurityType の指定**

   管理 CLI 内から以下のコマンドを実行して、**SecurityType** の値として **WSSecurity** を使用します。

   ```
   /subsystem=resource-adapters/resource-adapter=webservice/connection-
   definitions=wsDS/config-properties=SecurityType:add(value=WSSecurity)
   ```

2. **CXF 設定ファイルの変更**

Web サービスデータソース用の CXF 設定ファイルを開き、必要なプロパティーを追加します。

以下に、タイムスタンプを SOAP ヘッダーに追加する Web サービスデータソース CXF 設定ファイルの例を示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://cxf.apache.org/jaxws
          http://cxf.apache.org/schemas/jaxws.xsd">

    <jaxws:client name="{http://teiid.org}.teiid"
        createdFromAPI="true">
        <jaxws:outInterceptors>
            <ref bean="Timestamp_Request"/>
        </jaxws:outInterceptors>
    </jaxws:client>

    <bean
        id="Timestamp_Request">
        <constructor-arg>
            <map>
                <entry key="action" value="Timestamp"/>
            </map>
        </constructor-arg>
    </bean>

</beans>
```

注記

- WSDL は、使用するサービスを定義することを目的としません。
- Spring XML 設定ファイルは、関連するポリシー設定を含む必要があります。
- クライアントポート設定は、**CONFIG-NAME** によりデータソースインスタンスに対して照合されます。この設定には、異なるローカル名を持つ他のポート設定が含まれることがあります。

参照

- WS-Security および CXF 設定オプションの詳細については、http://cxf.apache.org/docs/ws-security.html を参照してください。

6.4. Web サービスデータソース用 CXF の設定: ロギング

CXF 設定は、特定またはすべてのポートに対する要求および応答のロギングを制御できます。ロギング (有効な場合) は、**org.apache.cxf.interceptor** コンテキストに対する **INFO** レベルで実行されます。

前提条件

- Web サービスデータソースが設定され、**ConfigFile** プロパティーと **EndPointName** プロパティーが CXF に対して設定されている必要があります。

手順6.3 Web サービスデータソース用 CXF の設定: ロギング

- **CXF 設定ファイルの変更**

  Web サービスデータソース用の CXF 設定ファイルを開き、必要なロギングプロパティーを追加します。

  以下に、ロギングを有効にする Web サービスデータソースの CXF 設定ファイルの例を示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
          http://cxf.apache.org/jaxws
          http://cxf.apache.org/schemas/jaxws.xsd">

    <jaxws:client name="{http://teiid.org}teiid"
        createdFromAPI="true">
        <jaxws:features>
            <bean class="org.apache.cxf.feature.LoggingFeature"/>
        </jaxws:features>
    </jaxws:client>

</beans>
```

参照

- CXF ロギング設定オプションの詳細については、http://cxf.apache.org/docs/debugging-and-logging.html を参照してください。

6.5. Web サービスデータソース用 CXF の設定: トランスポート設定

CXF 設定は、HTTP トランスポートの低レベルの側面を制御することもできます。

前提条件

- Web サービスデータソースが設定され、**ConfigFile** プロパティーと **EndPointName** プロパティーが CXF に対して設定されている必要があります。

手順6.4 Web サービスデータソース用 CXF の設定: トランスポート設定

- Web サービスデータソース用の CXF 設定ファイルを開き、必要なトランスポートプロパティーを追加します。

  以下に、ホスト名検証を無効にする Web サービスデータソースの CXF 設定ファイルの例を示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:jaxws="http://cxf.apache.org/jaxws"

xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
```

```
            http://cxf.apache.org/schemas/configuration/http-conf.xsd
            http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd">

    <http-conf:conduit name="{http://teiid.org}teiid.http-conduit">
          <http-conf:tlsClientParameters disableCNcheck="tru" />
    </http-conf:conduit>

</beans>
```

警告

本番稼働環境では、**disableCNcheck=true** を使用しないでください。

6.6. Web サービスデータソース用 CXF の設定: SSL サポート (HTTPS)

HTTPS を使用する場合は、以下のように CXF 設定を指定できます。

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:sec="http://cxf.apache.org/configuration/security"
    xmlns:http-conf="http://cxf.apache.org/transports/http/configuration"
    xmlns:jaxws="http://java.sun.com/xml/ns/jaxws"
    xsi:schemaLocation="http://cxf.apache.org/transports/http/configuration
http://cxf.apache.org/schemas/configuration/http-conf.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://cxf.apache.org/configuration/security
http://cxf.apache.org/schemas/configuration/security.xsd">

    <http-conf:conduit name="*.http-conduit">
        <http-conf:client ConnectionTimeout="120000"
ReceiveTimeout="240000"/>
        <http-conf:tlsClientParameters secureSocketProtocol="SSL">
          <sec:trustManagers>
            <sec:keyStore type="JKS" password="changeit"
file="/path/to/truststore.jks"/>
          </sec:trustManagers>
        </http-conf:tlsClientParameters>
    </http-conf:conduit>
</beans>
```

http-conduit ベースの設定の詳細については、http://cxf.apache.org/docs/client-http-transport-including-ssl-support.html を参照してください。また、HTTPBasic や Kerberos などのサービスも設定できます。

6.7. Salesforce データソース用 CXF の設定

前提条件

- Salesforce データソースが設定されている必要があります。

手順6.5 Salesforce データソース用 CXF の設定

1. 管理 CLI 内から以下のコマンドを実行してデータソースの CXF 設定ファイルを指定します。

```
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS/config-properties=ConfigFile:add(value=CONFIG-
FILE.xml)
```

2. 管理 CLI 内から以下のコマンドを実行し、値としてポート QName (ローカル部分のみ) を使用してポート設定を指定します (このケースでは **Soap**)。

```
/subsystem=resource-adapters/resource-adapter=salesforce/connection-
definitions=sfDS/config-properties=EndPointName:add(value=Soap)
```

3. ***EAP_HOME*/*MODE*/configuration/*CONFIG-FILE*.xml** 設定ファイルを開くか、作成します。**EndPointName** の値として **Soap** を使用して QName のネームスペース URI を **{urn:partner.soap.sforce.com}** に設定します。

```
<http-conf:conduit name="{urn:partner.soap.sforce.com}Soap.http-
conduit">
...
</http-conf:conduit>
```

以下に、タイムアウト値を設定する CXF ファイルの例を示します。

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:http-
conf="http://cxf.apache.org/transports/http/configuration"

xsi:schemaLocation="http://cxf.apache.org/transports/http/configuratio
n
  http://cxf.apache.org/schemas/configuration/http-conf.xsd
  http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd">

  <http-conf:conduit name="{urn:partner.soap.sforce.com}Soap.http-
conduit">
    <http-conf:client ConnectionTimeout="120000"
ReceiveTimeout="240000"/>
  </http-conf:conduit>
</beans>
```

注記

Salesforce データソース用 CXF 設定は、http バス設定にのみ使用されます。これは WS-Security には使用されません。Salesforce には独自のセキュリティー認証があります。

第7章 ロギング

7.1. ロギングの概要

JBoss EAP 6 は、独自の内部使用とデプロイされたアプリケーションによる使用のために設定可能な高度なロギング機能を提供します。ロギングサブシステムは JBoss LogManager を基盤とし、JBoss Logging 以外にも複数のサードパーティーアプリケーションのロギングフレームワークをサポートします。

ロギングサブシステムは、ログカテゴリーとログハンドラーのシステムを使用して設定されます。ログカテゴリーはキャプチャーするメッセージを定義し、ログハンドラーはこれらのメッセージの処理方法を定義します (ディスクへの書き込みやコンソールへの送信など)。

ロギングプロファイルは、一意に名前が付けられたロギング設定のセットを作成し、他のロギング設定に依存しないアプリケーションへ割り当てることが可能です。ロギングプロファイルの設定はメインのロギングサブシステムとほぼ同じです。

7.2. デフォルトのログファイルの場所

これらは、デフォルトのロギング設定に対して作成されたログファイルです。デフォルトの設定では、周期的なログハンドラーを使用してサーバーログファイルが書き込まれます。サーバーはデフォルトで、teiid-command.log および teiid-audit.log に書き込むように設定されています。

表7.1 スタンドアロンサーバーのデフォルトログファイル

ログファイル	説明
***EAP_HOME*/standalone/log/server.log**	サーバーログ。サーバー起動メッセージなど、すべてのサーバーログメッセージが含まれます。
***EAP_HOME*/standalone/log/gc.log**	ガベッジコレクションのログ。ガベッジコレクションすべての詳細が含まれます。

表7.2 管理対象ドメイン用のデフォルトログファイル

ログファイル	説明
***EAP_HOME*/domain/log/host-controller.log**	ホストコントローラーのブートログ。ホストコントローラーの起動に関連するログメッセージが含まれます。
***EAP_HOME*/domain/log/process-controller.log**	プロセスコントローラーのブートログ。プロセスコントローラーの起動に関連するログメッセージが含まれます。
***EAP_HOME*/domain/servers/*SERVERNAME*/log/server.log**	名前付きサーバーのサーバーログ。サーバー起動メッセージなど、そのサーバーのすべてのログメッセージが含まれます。

7.3. JBoss Data Virtualization ログカテゴリー

EAP ログファイル内では、JBoss Data Virtualization に固有なすべての情報には、**org.teiid** ログカテゴリー識別子の接頭辞が付けられます。

JBoss Data Virtualization に関係ないサードパーティーのコードと他のコンポーネント (統合された org.jboss コンポーネントを含む) のログには、**org.teiid** ではなく独自のログカテゴリー識別子の接頭辞が付けられます。

以下の表は、JBoss Data Virtualization に関連するカテゴリーの概要を示しています。完全なリストについては、**standalone.xml** (ドメインモードの場合は **domain.xml**) を参照してください。

表7.3 JBoss Data Virtualization ログカテゴリー

ログカテゴリー	説明
com.arjuna	サードパーティートランザクションマネージャー。これには、JBoss Data Virtualization のトランザクションだけではなくすべてのトランザクションに関する情報が含まれます。
org.teiid	すべての JBoss Data Virtualization ログのルートカテゴリー識別子。注記: org.teiid には、この表に示されているよりも多いコンテキストがある場合があります。
org.teiid.PROCESSOR	クエリー処理ログ。org.teiid.PLANNER も参照してください。
org.teiid.PLANNER	クエリー計画ログ。
org.teiid.SECURITY	セッション/認証イベント。org.teiid AUDIT_LOG も参照してください。
org.teiid.TRANSPORT	ソケットトランスポートに関連するイベント。
org.teiid.RUNTIME	ワーク管理およびシステムの起動/終了に関連するイベント。
org.teiid.CONNECTOR	コネクターログ。
org.teiid.BUFFER_MGR	バッファーおよびストレージ管理ログ。
org.teiid.TXN_LOG	すべてのトランザクション操作の詳細なログ。
org.teiid.COMMAND_LOG	「コマンドロギング」 を参照してください。
org.teiid.AUDIT_LOG	「監査ロギング」 を参照してください。
org.teiid.ADMIN_API	管理 API ログ。
org.teiid.ODBC	ODBC ログ。

注記

ロギングの詳細については、JBoss Enterprise Application Platform 『管理および設定ガイド』 を参照してください。

7.4. コマンドロギング

コマンドロギングは、JBoss Data Virtualization に送信されたユーザーコマンド、クエリー計画コマンド (クエリー計画が実行された場合)、およびコネクターにより実行されるデータソースコマンドを取得します。

ユーザーコマンド "START USER COMMAND" は、JBoss Data Virtualization でクエリーが初めて実行されたときにログに記録されます。これには、クエリーがキューで待機した時間は含まれません。対応するユーザーコマンド "END USER COMMAND" は、要求が完全な場合 (つまり、ステートメントが閉じられた場合、またはすべてのバッチが取得された場合) にログに記録されます。各ユーザークエリーに対してこれらのペアが 1 つだけ存在します。

クエリー計画コマンド "PLAN USER COMMAND" は、JBoss Data Virtualization でクエリー計画プロセスが完了したときにログに記録されます。対応する終了ログエントリーはありません。

Non-plan user イベントは INFO レベルでログ記録されます。

データソースコマンド "START DATA SRC COMMAND" は、クエリーがデータソースに送信されたときにログに記録されます。対応するデータソースコマンド "END SRC COMMAND" は、実行が閉じられたとき (つまり、すべての行が読み取られたとき) にログに記録されます。JBoss Data Virtualization により実行される各データソースクエリーに対して 1 つのペア、ユーザーのクエリーに応じて複数のペアが存在すること

があります。

この情報が取得される場合は、全体的なクエリー実行時間を計算できます。また、各ソースクエリー実行時間も計算できます。全体的なクエリー実行時間がパフォーマンスの問題を示している場合は、どこに問題が存在するのかを調べるために各データソース実行時間を確認してください。

この情報が取得される場合は、Teiid で全体的なクエリー実行時間を計算できます。また、各ソースクエリー実行時間も計算できます。全体的なクエリー実行時間がパファーマンスの問題を示している場合は、どこに問題が存在するのかを調べるために各データソース実行時間を確認してください。

デフォルトのログの場所に対してコマンドロギングを有効にするには、org.teiid.COMMAND_LOG コンテキストのロギングの DEBUG レベルを有効にします。これは管理コンソールを使って有効または無効にできます。Web コンソールを使用してコマンドロギングをオンにすることもできます。

別のファイルの場所に対してコマンドロギングを有効にするには、org.teiid.COMMAND_LOG コンテキストの DETAIL ロギングに対して個別のファイルアペンダーを設定します。この例は以下で示されており、standalone.xml ファイルにもあります。

```
<periodic-rotating-file-handler name="COMMAND_FILE">
    <level name="DEBUG" />
    <formatter>
        <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"
/>
    </formatter>
    <file relative-to="jboss.server.log.dir" path="command.log" />
    <suffix value=".yyyy-MM-dd" />
</periodic-rotating-file-handler>

<logger category="org.teiid.COMMAND_LOG">
    <level name="DEBUG" />
    <handlers>
        <handler name="COMMAND_FILE" />
    </handlers>
</logger>
```

カスタムロギングソリューションの開発については、『Red Hat JBoss Data Virtualization Development Guide: Server Development』を参照してください (ファイルベース (または他の組み込み log4j) ロギングが十分でない場合)。

以下は、データソースコマンドの例であり、コマンドログに出力される内容を示しています。

```
2012-02-22 16:01:53,712 DEBUG [org.teiid.COMMAND_LOG]
(Worker1_QueryProcessorQueue11 START DATA SRC COMMAND: startTime=2012-02-22
16:01:53.712
requestID=Ku4/dgtZPYk0.5 sourceCommandID=4 txID=null modelName=DTHCP
translatorName=jdbc-simple sessionID=Ku4/dgtZPYk0
principal=user@teiid-security
sql=HCP_ADDR_XREF.HUB_ADDR_ID, CPN_PROMO_HIST.PROMO_STAT_DT FROM
CPN_PROMO_HIST, HCP_ADDRESS, HCP_ADDR_XREF
WHERE (HCP_ADDRESS.ADDR_ID = CPN_PROMO_HIST.SENT_ADDR_ID) AND
(HCP_ADDRESS.ADDR_ID = HCP_ADDR_XREF.ADDR_ID) AND
(CPN_PROMO_HIST.PROMO_STAT_CD NOT LIKE 'EMAIL%') AND
(CPN_PROMO_HIST.PROMO_STAT_CD <> 'SENT_EM') AND
(CPN_PROMO_HIST.PROMO_STAT_DT > {ts'2010-02-22 16:01:52.928'})
```

以下の情報に注意してください。

- modelName: これは、クエリーが発行されるデータソースの物理モデルを表しています。
- translatorName: データソースと通信するのに使用するトランスレーターのタイプを示します。
- principal: クエリーを実行したユーザーアカウントを示します。
- startTime/endTime: アクションの時間。実行されるタイプコマンドに基づきます。
- sql: 実行のためにトランスレーターに送信されたコマンド。必ずしも実際のデータソースに送信される最終の sql コマンドとは限らず、クエリーエンジンがプッシュする内容を示します。

7.5. 監査ロギング

監査ロギングは、パーミッションの適用や認証の合否を含む重要なセキュリティーイベントを取得します。

カスタムロギングソリューションの開発については、『Red Hat JBoss Data Virtualization Development Guide: Server Development』を参照してください (ファイルベース (または他の組み込み log4j) ロギングが十分でない場合)。

7.6. 監査およびコマンドロギングの有効化

監査およびコマンドロギングは Web コンソールから有効にする必要があります。また、一部のユーザーにロギングロールを付与する必要もあります。

注記

データベースを使用してロギングを保存している場合は、Dashboard ロギングワークスペースを使用できます。

Teiid ユーザーはロギングロール権限が必要になります (ほとんどの場合で、インストーラーから設定します)。

第8章 クラスタリング

8.1. Red Hat JBoss Data Virtualization でのクラスタリング

クラスタリングは、以下の機能を使用してシステムのパフォーマンスを向上させるために使用できます。

ロードバランシング

複数のノード間の負荷分散については、『Red Hat JBoss Data Virtualization Development Guide: Client Development』を参照してください。

フェイルオーバー

複数のノードとのフェイルオーバーについては、『Red Hat JBoss Data Virtualization Development Guide: Client Development』を参照してください。

分散キャッシング

クラスターモードでは、resultsset キャッシュと内部実体化キャッシュは自動的にクラスターで共有されます。このため、それ以上の設定は必要ありません。

イベント分散

クラスタリングが設定されている場合、メタデータとデータの変更はクラスターのすべてのメンバーに分散されます。

8.2. JBoss Data Virtualization でのクラスタリングの有効化

各 JBoss EAP ノードに JBoss Data Virtualization がインストールされていることと、クラスターを開始する前に、**`standalone-ha.xml`** プロファイルまたは **`standalone-full-ha.xml`** プロファイルのいずれかを使用して JBoss EAP が起動されていることを確認します (これらのプロファイルは名前が示すように高可用性またはクラスタリングを提供します)。

詳細については、『Red Hat JBoss Data Virtualization Installation Guide』 と Red Hat JBoss Enterprise Application Platform のドキュメンテーションを参照してください。

注記

クラスタリングが使用されている場合は、ファイルデプロイメントを介して VDB などのアーティファクトをデプロイできません。VDB デプロイメントの他のメソッドを使用してください。

パート V. 監視およびパフォーマンス

第9章 モニタリング

9.1. Red Hat JBoss Data Virtualization の監視

Red Hat JBoss Data Virtualization は、現在の処理状態に関する情報を提供します。この情報は、負荷およびスループットをチューニング、監視、および管理するのに役に立つことがあります。実行時データは、Web コンソール、AdminShell、Admin API などの管理ツールを使用してアクセスできます。

9.2. クエリー/セッションの詳細

名前	説明
現在のセッション	現在接続されているセッションのリスト
現在の要求	現在実行している要求のリスト
現在のトランザクション	現在実行しているトランザクションのリスト
クエリー計画	特定の要求のクエリー計画の取得

セッション、クエリー、およびトランザクションを終了する管理オプションがあります。

9.3. セッション/クエリーメトリックス

名前	プロパティー	説明	コメント
セッション数	sessionCount	現在アクティブなユーザー接続の数を示します。	ピーク時にセッション数が制限されないようにするために、max-sessions-allowed (デフォルト値は 5000) が適切に設定されていることをチェックし、sessions-expiration-timelimit を見直します。
クエリー数	queryCount	現在アクティブなクエリーの数を示します。	
アクティブなクエリー計画数	ENGINE_STATISTIC.active-plans-count	現在処理中のクエリー計画の数。	スループットを最大化するには、QueryEngine とスレッドのパフォーマンスチューニングに関する項を参照してください。
待機中のクエリー計画数	ENGINE_STATISTIC.waiting-plans-count	現在待機中のクエリー計画の数。	
待機中の最大クエリー計画ウォーターマーク	ENGINE_STATISTIC.max-waitplan-watermark	最後にサーバーが起動してから待機しているクエリー計画の最大数。	

名前	プロパティー	説明	コメント
長時間実行中のクエリー	longRunningQueries	クエリーしきい値 (query-threshold-in-seconds.) を超えた現在実行中のクエリーをリストします。	1 つまたは複数のクエリーが長時間リソースを消費していることを警告するアラートを設定します。長時間実行している場合は、オプションで要求をキャンセルしたり、しきい値を増やしたりします。

9.4. バッファーマネージャーメトリックス

名前	プロパティー	説明	コメント
ディスク書き込み数	ENGINE_STATISTIC.buffermgr-disk-write-count	バッファーマネージャーのディスク書き込み数。	
ディスク読み取り数	ENGINE_STATISTIC.buffermgr-disk-read-count	バッファーマネージャーのディスク読み取り数。	
キャッシュ書き込み数	ENGINE_STATISTIC.buffermgr-cache-write-count	バッファーマネージャーのキャッシュ書き込み数。	
キャッシュ読み取り数	ENGINE_STATISTIC.buffermgr-cache-read-count	バッファーマネージャーのキャッシュ読み取り数。	
ディスク使用容量 (MB)	ENGINE_STATISTIC.buffermgr-diskspace-used-mb	バッファーファイルで現在使用されているストレージ容量を示します。	max-buffer-space の設定に基づいて使用済みバッファー容量が許容できないレベルである場合に警告するアラートを設定します。
総使用メモリー容量 (KB)	ENGINE_STATISTIC.total-memory-inuse-kb	現在のメモリー使用量の推定値 (キロバイト単位)。	
アクティブな計画別総使用メモリー容量 (KB)	ENGINE_STATISTIC.total-memory-inuse-active-plans-kb	アクティブな計画別の現在のメモリー使用量の推定値 (キロバイト単位)。	

9.5. キャッシュメトリックス

名前	プロパティー	説明
準備された計画キャッシュサイズ	PREPARED_PLAN_CACHE.total-entries	現在キャッシュにあるエントリーの数。
準備された計画キャッシュの要求数	PREPARED_PLAN_CACHE.request-count	キャッシュに対して行われた要求の合計数。
準備された計画キャッシュのヒット率 (%)	PREPARED_PLAN_CACHE.hit-ratio	正のキャッシュヒット率。
結果セットキャッシュサイズ	QUERY_SERVICE_RESULT_SET_CACHE.total-entries	現在キャッシュにあるエントリーの数。
結果セットキャッシュの要求数	QUERY_SERVICE_RESULT_SET_CACHE.request-count	キャッシュに対して行われた要求の合計数。

名前	プロパティー	説明
結果セットキャッシュヒット率 (%)	QUERY_SERVICE_RESULT_SET_CACHE.hit-ratio	正のキャッシュヒット率。

第10章 パフォーマンスチューニング

10.1. メモリー管理に関する考慮事項

以下に、メモリー管理に関する設定についての一部の情報を記載します。

Red Hat JBoss Data Virtualization では、BufferManager 使ってメモリーとディスク使用量の両方を追跡しています。パフォーマンスを高めるには、BufferManager を適切に設定することが重要になります。ただし、ほとんどのインスタンスでは、JVM とスケールし、最大アクティブプランの設定などの他のプロパティーを考慮することから、デフォルト設定で十分なものになっています。CLI で以下のコマンドを実行すると BufferManager で可能な全設定が表示されます。

```
/subsystem=teiid:read-resource
```

"buffer-service" で始まるプロパティーはすべて、BufferManager の設定に使用できます。以下は、これらの設定変更に使用できる CLI コマンドです。

```
/subsystem=teiid:write-attribute(name=buffer-service-use-disk,value=true)
/subsystem=teiid:write-attribute(name=buffer-service-encrypt-
files,value=false)
/subsystem=teiid:write-attribute(name=buffer-service-processor-batch-
size,value=256)
/subsystem=teiid:write-attribute(name=buffer-service-max-open-
files,value=64)
/subsystem=teiid:write-attribute(name=buffer-service-max-file-
size,value=2048)
/subsystem=teiid:write-attribute(name=buffer-service-max-processing-
kb,value=-1)
/subsystem=teiid:write-attribute(name=buffer-service-max-reserve-kb,value=-
1)
/subsystem=teiid:write-attribute(name=buffer-service-max-buffer-
space,value=51200)
/subsystem=teiid:write-attribute(name=buffer-service-max-inline-
lobs,value=true)
/subsystem=teiid:write-attribute(name=buffer-service-memory-buffer-
space,value=-1)
/subsystem=teiid:write-attribute(name=buffer-service-max-storage-object-
size,value=8388608)
/subsystem=teiid:write-attribute(name=buffer-service-memory-buffer-off-
heap,value=false)
```

警告

これらのプロパティーは、プロパティー自体と発生している問題を理解するまで変更しないでください。

主な設定の説明は以下の通りです。

- max-reserve-kb (デフォルト値は -1) - この設定により、バッファマネージャーがメモリー内に保持できるバッチの合計サイズ (キロバイト単位) が決定されます。この数字には、ソフト参照 (インデックスページなど) または弱い参照で保持される永続バッチは含まれません。-1 のデフォルト値の場合は、VM で利用可能な最大ヒープに基づいて通常の最大値が自動的に計算されます。自動計算される値では、64

ビットのアーキテクチャーを前提とし、バッファーの使用が最初の 300 メガバイト (AS とその他の Red Hat JBoss Data Virtualization 目的の使用を担う) を超えるギガバイトの 40% とそれを超えるメモリーの 50% に制限されます。他に注意すべき点は、メモリーのバッファースペースのサイズが指定されないと、最大の予約領域が実質上割り当てられることになります。バッチトラッキングのオーバーヘッドを含めるよう、最大予約には多少の調節がなされます。

デフォルト設定で 8GB VM サイズの場合、max-reserve-kb は最大で (((1024-300) * 0.4) + (7 * 1024 * 0.5)) = 4373.6 MB または 4,478,566 KB を使用します。

- 予約領域の 25% 以上が使用されていて有効になっていると、バッファーマネージャーは正規の値キャッシュを自動的に使用します。この結果、類似の値セットが Red Hat JBoss Data Virtualization によって読み取られる場合にメモリー使用量が大幅に削減されることがあります。ただし、ルックアップコストが発生します。Red Hat JBoss Data Virtualization を使用して小さいデータセットまたは類似性が非常に高いデータセットを処理するときにメモリーを節約する場合は、値キャッシュの有効化を検討してください。

  メモリー消費量は、実際の列値と、値キャッシュが有効であるかどうかに応じて名目ターゲットよりも大幅に大きく、または小さくなることがあります。組み込みではない大きいオブジェクトはデフォルトの推測サイズを超えることがあります。メモリー不足エラーが発生した場合は、max-reserve-kb の値を低く設定します。また、ソース LOB 値はバッチが永続化されたときに削除されなかったメモリー参照により保持されることに注意してください。LOB を大量に使用する場合は、LOB 参照に関連付けられた他のメモリーのバッファーのサイズが適切に設定されていることを確認する必要があります。

- max-processing-kb (デフォルト値は -1) - この設定により、max-reserve-kb に基づいて保持されるメモリー以外に、1 つのアクティブな計画で使用できるバッチの合計サイズ (キロバイト単位) が決まります。すべてのアクティブな計画がアクティブな場合に Red Hat JBoss Data Virtualization で必要な通常の最小メモリーは #active-plans*max-processing-kb です。-1 のデフォルト値の場合、VM で利用可能な最大ヒープとアクティブな最大計画数に基づいて一般的な最大値が自動的に計算されます。自動計算される値では、64 ビットのアーキテクチャを前提とし、名目のバッチ処理用の容量はメモリー合計の 10% 未満に制限されます。

  20 のアクティブな計画と 8GB VM サイズを含むデフォルト設定の場合、max-processing-kb は (.07 * 8 * 1024)/20^.8 = 537.4 MB/11 = 52.2 MB または 53,453 KB (1 つの計画あたり) になります。これは、1 つの計画あたり約 53 MB で 0 から 1060 MB の名目範囲が予約されることを示しています。max-processing-kb を自分で調節することについては、慎重になってください。通常、計画でメモリーが制限され、パフォーマンスが低下している場合を除いて、調節の必要はありません。

- max-file-size (デフォルト値は 2GB) - 各中間結果バッファー、一時 LOB、および一意なテーブルは、独自のバッファーファイルセットに格納され、各ファイルは max-file-size メガバイトに制限されます。インストールで内部実体化を使用したり、SQL/XML を頻繁に使用したり、大量の行数を処理したりする場合は、max-buffer-space を増加させて、このようなすべてのファイルに利用可能なストレージ領域を拡大することを検討してください。

- processor-batch-size (デフォルト値は 256) - クエリプロセッサーのバッチのターゲット列の数を指定します。バッチは、保存された結果などの線形データストアや一時的な表ページの表示に使用されます。Teiid は、列のデータ幅の予測に基づいて processor-batch-size を名目予測の 2KB に対して機能するサイズに調節します。ベースとなる値は、データ幅予測によって、2 倍または最大で 3 分の 1 にすることが可能です。たとえば、(整数などの) 単一の小規模固定幅のコラムバッチは、processor-batch-size * 8 列のサイズになります。(文字列などの) 数百もの変動幅データがあるバッチは、processor-batch-size / 8 列のサイズになります。プロセッサーバッチサイズの増加で最初の 2 倍以降は、max-storage-object-size も相対的に増加させてストレージサイズが大きくなったバッチに対応してください。

  大規模な VM サイズやデータセットを使用する場合は、さらなる考慮が必要になります。Red Hat JBoss Data Virtualization には、バッチ/ページあたりおよそ 100-200 バイトという無視できないオーバーヘッドがあります。数千万の列があるデータセットの処理中にメモリー不足という問題が発生したら、processor-batch-size を増やしてより大きなバッチと表ページの割り当ての強制を検討してください。ー

般的なガイドラインとしては、4 GB を超える Red Hat JBoss Data Virtualization の実効ヒープを 2 倍にする場合は processor-batch-size も 2 倍にします。たとえば、8 GB ヒープでは processor-batch-size = 512、16 GB ヒープでは processor-batch-size = 1024 といったようにします。

- max-storage-object-size (デフォルト値は 8288608 または 8MB) - バッファー済み管理オブジェクトの最大サイズ (バイト単位) で、個別のバッチページサイズを表します。processor-batch-size を増加した場合、または (数百コラムの) 非常に幅広い結果セットを使用している場合、max-storage-object-size のデフォルト設定 (8MB) は小さすぎることがあります。また、inline-lobs の設定でも、小さな lob を含むバッチのサイズを増やすことができます。max-storage-object-size のサイズはシリアライズ化されたサイズであり、max-reserve-kb に使用される Java メモリーフットプリント推定値よりもローデータサイズにもっと近いものになります。max-storage-object-size はメモリーバッファーのパフォーマンスを低下させるおそれがあるので、これを memory-buffer-space に対して大きすぎる設定にしないでください。メモリーバッファーは、memory-buffer-space の各 max-storage-object-size の 1 ライターのみを一度にサポートします。この値は、processor-batch-size を調節しなければ、変更する必要は通常ありません。これを変更する場合は、processor-batch-size の増加に合わせて調整してください。

  バッチが見つからないことで例外が発生し、サーバーログに "TEIID30001 Max block number exceeded" が表示される場合は、max-storage-object-size を増やして大きいストレージオブジェクトをサポートします。別の方法では、processor-batch-size を小さくすることもできます。memory-buffer-space (デフォルト値は -1) - Red Hat JBoss Data Virtualization バッファーマネージャーが使用するバイトバッファーとして割り当てられたオンまたはオフヒープメモリーのサイズを制御します。この設定のデフォルト値は -1 で、この場合、オンヒープまたはオフヒープであるか、または max-reserve-kb の値に基づいて設定が自動的に決定されます。メモリーバッファーは、memory-buffer-space の各 max-storage-object-size で同時ライターを 1 つだけサポートします。他のスペースは、シリアル化されたバッチのキャッシュとして使用されます。

  デフォルト設定のままにすると、計算されるメモリーバッファースペースは、max-reserve-kb サイズの約 40% となります。メモリーバッファーがオンヒープで max-reserve-kb が自動計算されると、メモリーバッファースペースから実効 max-reserve-kb が引かれます。メモリーバッファーがオフヒープで max-reserve-kb が自動計算されると、そのサイズは多少小さくなり、仮想マシン内の機能メモリーの実効性が高まります。memory-buffer-off-heap (デフォルト値は false) - ヒープに割り当てずにシステムメモリーにアクセスするバッファーマネージャーのメモリーバッファーを活用します。memory-buffer-off-heap を "true" に設定すると、Red Hat JBoss Data Virtualization のメモリーバッファーはオフヒープに割り当てられます。インストールが Red Hat JBoss Data Virtualization かどうか、また利用可能なシステムメモリーの量によって、これはオンヒープ割り当てとすることが好ましい場合もあります。第一の利点は、追加のガーベッジコレクションチューニングなしで Red Hat JBoss Data Virtualization 用にメモリー使用量を追加できることです。これは、VM に 32GB を超えるメモリーが必要な場合に特に重要です。オフヒープ割り当てを使用する場合は、java プロセスがメモリーを使用できるようにする必要があります。memory-buffer-space の値を高く設定しすぎると、VM がメモリーに留まらずにスワップすることがあります。また、オフヒープバッファーサイズが大きい場合 (数ギガバイト以上) は、VM 設定を調整する必要がある場合があります。

Sun VM の場合、該当する VM 設定は MaxDirectMemorySize と UseLargePages です。たとえば、

```
-XX:MaxDirectMemorySize=12g -XX:+UseLargePages
```

上記の内容を VM プロセス引数に追加すると、Red Hat JBoss EAP または EAP 内で実行中のアプリケーションで必要となる可能性がある追加の直接メモリーに対応する約 11GB の Red Hat JBoss Data Virtualization メモリーバッファー (memory-buffer-space 設定) が効果的に割り当てられます。

10.2. スケーラビリティーに関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、スケーラビリティーに関連する設定についての追加情報を提供します。

buffer-service-processor-batch-size

デフォルト値は 256 です。このプロパティーは、クエリープロセッサー内で内部的に送信されるバッチの最大行数を指定します。非常に大きい VM サイズまたはデータセットが使用される場合は、追加の考慮事項があります。JBoss Data Virtualization には、バッチ/テーブルページごとのオーバーヘッドの無視できない量 (100～200 バイト) があります。関連するデータ型に応じて、完全な各バッチ/テーブルは可変行数 (プロセッサーバッチサイズ以上または以下を 2 の累乗数で乗算) を表します。

非常に大きいデータセットを使用していてメモリー問題が発生した場合は、大きいバッチとテーブルページの割り当てを強制するために **buffer-service-processor-batch-size** プロパティーを増加することを検討してください。

buffer-service-max-storage-object-size

デフォルト値は 8288608、つまり 8MB です。この値は、バッファリングされた管理対象オブジェクトの最大サイズ (バイト単位) であり、個々のバッチページサイズを表します。

buffer-service-processor-batch-size が増加された場合、または非常に幅広い結果セットを使用している場合、**buffer-service-max-storage-object-size** のデフォルト設定 (8MB) は小さすぎることがあります。また、インライン LOBS は、バッチにそれが含まれる場合はこのサイズに寄与します。**buffer-service-max-storage-object-size** のサイズは、シリアライズ化されたサイズであり、**buffer-service-max-reserve-kb** に使用される Java メモリーフットプリント推定値よりもローデータサイズにもっと近くなります。

buffer-service-max-storage-object-size は、メモリーバッファーのパフォーマンスが低下するため、**buffer-service-memory-buffer-space** よりもあまり大きく設定しないでください。メモリーバッファーは **buffer-service-memory-buffer-space** の各 **buffer-service-max-storage-object-size** に対して同時ライターを 1 つだけサポートします。

注記

JBoss Data Virtualization 一時テーブル (内部的なマテリアライズにも使用されます) は、テーブルごとに 2^31-1 行だけサポートできます。

buffer-service-memory-buffer-space

デフォルト値は -1 です。これにより、JBoss Data Virtualization バッファーマネージャーが使用するバイトバッファーとして割り当てられたオンまたはオフヒープメモリーの量が制御されます。この設定のデフォルト値は -1 です。これにより、オンヒープまたはオフヒープであるか、あるいは **buffer-service-max-reserve-kb** の値に基づいて設定が自動的に決定されます。

注記

デフォルト設定のままの場合は、計算されるメモリーバッファー領域が **buffer-service-max-reserve-kb** 設定の約 4 分の 1 になります。メモリーバッファーがオフヒープであり、**buffer-service-max-reserve-kb** 設定が自動的に計算される場合は、メモリーバッファー領域が効果的な **buffer-service-max-reserve-kb** から引かれます。

buffer-service-memory-buffer-off-heap

デフォルト値は false です。バッファーマネージャーメモリーバッファーを使用してヒープに割り

当てずにシステムメモリーにアクセスするかどうかを決定します。**buffer-service-memory-buffer-off-heap** を **true** に設定すると、JBoss Data Virtualization メモリーバッファーオフヒープが割り当てられます。インストールが JBoss Data Virtualization 専用であるかどうかと利用可能なシステムメモリーの量に応じて、これは、オンヒープ割り当てよりも適切な場合があります。

オフヒープメモリーの第一の利点は、追加のガーベッジコレクションチューニングなしで JBoss Data Virtualization 用にメモリー使用量を追加できることです。これは、JVM に 32GB を超えるメモリーが必要な場合に特に重要です。オフヒープ割り当てを使用する場合は、プロセスがメモリーを使用できるようにする必要があります。値 **buffer-service-memory-buffer-space** が大きすぎると、JVM がメモリーに留まらずにスワップすることがあります。また、オフヒープバッファーサイズが大きい場合 (数ギガバイト以上) は、JVM 設定を調整する必要があります。

Sun JVM の場合、該当する JVM 設定は **MaxDirectMemorySize** と **UseLargePages** です。たとえば、

```
-XX:MaxDirectMemorySize=12g -XX:+UseLargePages
```

上記の内容を JVM プロセス引数に追加すると、JBoss EAP またはアプリケーションで必要となる可能性がある追加の直接メモリーに対応する約 11GB の JBoss Data Virtualization メモリーバッファー (**buffer-service-memory-buffer-space** プロパティー) が効果的に割り当てられます。

10.3. ディスク使用に関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、ディスク使用量に関連する設定についての追加情報を提供します。

buffer-service-max-buffer-space

デフォルト値は 51200 です。テーブルページと結果バッチについて、バッファーマネージャーは、特定のストレージサイズ専用のファイルの数を制限します。ただし、Large Object (LOB) 値を作成する場合 (たとえば、SQL/XML を使用)、通常、LOB が 8KBのメモリーサイズの許容値を超えると、LOB ごとに 1 つのファイルが作成されます。使用量が大きい場合は、定期的に最適化するパーティション上のバッファディレクトリーを参照することを検討してください。デフォルトでは、JBoss Data Virtualization はディスク領域を最大 50GB 使用します。これは、JBoss Data Virtualization が書き込むバイト数によって追跡されます。大きいデータセットの場合は、**buffer-service-max-buffer-space** プロパティーを増加する必要があることがあります。

10.4. スレッドに関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、スレッディングに関連する設定についての追加情報を提供します。

max-threads

デフォルト値は 64 です。クエリーエンジンには、スレッドの使用を決定する複数の設定があります。**max-threads** は、クエリーエンジンの作業 (処理計画、トランザクション制御操作、処理ソースクエリーなど) 用のプロセスプールで利用可能なスレッドの合計数を設定します。

大量の利用可能なプロセッサの場合、または大量の同時ソース要求に関連する非トランザクションクエリーを発行する必要がある場合は、システムで最大スレッド数を増加することを検討してください。

max-active-plans

デフォルトは 20 です。この値は常に **max-threads** よりも小さくなければなりません。デフォルトでは、thread-count-for-source-concurrency は (max-threads / max_active_plans) * 2 で算出され、各ユーザークエリーの同時ソースリクエストを処理できるスレッドを決定します。長期実行されるクエリーが多いワークロード場合や、使用できるプロセッサーの数が多いシステムの場合は、max-active-plans を増加することを検討してください。max-threads および max-active-plans を増加するとメモリーの問題が発生する場合は、バッファーマネージャーが保持するヒープの量を少なくするか、各計画で消費するメモリー行のベース数を制限するために processor-batch-size を減少することを検討してください。

大量の利用可能なプロセッサで大量のクエリーまたはシステムを長時間実行するワークロードの場合は、**max-active-plans** を増加することを検討してください。**max-threads** と **max-active-plans** を増加するときにメモリー問題が発生した場合は、各計画で消費するメモリー行のベース数を制限するために **buffer-service-processor-batch-size** を減少することを検討してください。

thread-count-for-source-concurrency

デフォルト値は 0 です。この値は、常に **max-threads** よりも小さくする必要があります。このプロパティーは、ユーザー要求ごとに同時に実行するソースクエリーの数を設定します。0 は 2 * (**max-threads** / **max-active-plans**) に基づいて計算されたデフォルト値を使用することを示します。これを 1 に設定すると、処理スレッドによりすべてのソースクエリーが連続して実行されます。1 よりも大きい値を設定すると、同時に実行するソースクエリーの最大数が制限されます。

デフォルト値を使用すると、各ユーザー要求に 6 つの同時実行ソースクエリーが許可されます。計算されたデフォルト値がワークロードに該当しない場合 (たとえば、クエリーで長時間実行する同時ソースクエリーをさらに生成する場合) は、この値を調整することを検討してください。

また、**max-socket-threads** については 「トランスポートに関する考慮事項」 を参照してください。

10.5. キャッシュに関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、キャッシングに関連する設定についての追加情報を提供します。

キャッシュチューニングに関する JBoss Data Virtualization 設定は、以下の 2 つに分けられます。

- 結果セットキャッシュチューニング
- 準備された計画キャッシュチューニング

キャッシュ統計は、管理コンソールまたは AdminShell で取得できます。この統計は、キャッシュパラメーターをチューニングし、ヒット比率を確保するために使用できます。

計画は現在メモリーに完全に保持され、メモリーフットプリントは非常に大きい可能性があります。準備されたステートメントまたは仮想プロシージャーを大量に使用する場合は、計画キャッシュのサイズを JBoss Data Virtualization が使用するギガバイト数に応じて増加できます。

結果キャッシュパラメーターがキャッシュ結果エントリー (最大数やエビクションなど) を制御する一方で、結果バッチ自体はバッファーマネージャーによってアクセスされます。結果キャッシュのサイズを増加する場合は、十分なバッファー領域を確保するためにバッファーマネージャー設定をチューニングする必要があることがあります。

結果セットと準備された計画キャッシュには、データおよびメタデータイベントにより無効にされたエントリーがあります。デフォルトでは、これらのイベントは JBoss Data Virtualization からコマンドを実行することにより取得されます (カスタマイズの詳細については、『Red Hat JBoss Data Virtualization Developer Guide』を参照してください)。JBoss Data Virtualization には、準備された計画とともにコンパイルされた形式の更新計画またはトリガーアクションが格納されているため、メタデータが変更されると、その変更がすぐに反映されることがあります。

すぐに変更されるソースで効率性を上げるために、結果セットキャッシュのデフォルトの **resultset-cache-max-staleness** は 60 秒になっています。結果セットキャッシュと基礎となるデータとの整合性を上げる場合は、この値を減少させることを検討してください。設定値が 0 であっても、完全なトランザクションの整合性は保証されません。

警告

これらのキャッシュを無効にしたり、制限したりすると、パフォーマンスが低下します。

10.6. トランスポートに関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、トランスポートに関連する設定についての追加情報を提供します。

JBoss Data Virtualization は、デフォルトで odbc、jdbc、および組み込みの 3 つのトランスポートを提供します。それぞれに対してトランスポート設定 (以下に示されるような設定) が指定されます。

max-socket-threads

デフォルト値は 0 です。初期要求処理専用のスレッドの最大数が決定されます。ゼロは、利用可能な最大プロセッサ数のシステムデフォルト値を使用することを示します。ソケットスレッドは NIO 非ブロック IO 操作を処理し、ブロックなしで実行できるすべての操作を直接処理します。長時間実行されている操作の場合、ソケットスレッドはクエリーエンジンでワークをキューに格納します (クエリーエンジンには、スレッドの使用を決定する **max-threads** と **max-active-plans** の 2 つのプロパティーがあります)。

すべての JDBC/ODBC ソケットの操作は非ブロックであり、**max-socket-threads** の数をマシンの効果的な最大並列処理数よりも大きく設定すると、パフォーマンスが低下することがあります。

input-buffer-size

デフォルト値は、システムデフォルト値を使用する 0 です。トランスポートに対して **input-buffer-size** を調整する前に、各クライアントが新しいソケット接続を作成することに注意してください。この値の増加は、クライアント数が制限されている場合のみ行ってください。

output-buffer-size

デフォルト値は、システムデフォルト値を使用する 0 です。トランスポートに対して **output-buffer-size** を調整する前に、各クライアントが新しいソケット接続を作成することに注意してください。この値の増加は、クライアント数が制限されている場合のみ行ってください。

JDBC クライアントは、クライアントアプリケーションのクラスパスに指定された **teiid-client-settings.properties** ファイルを介した SSL クライアント接続プロパティー以外に、低レベルトランスポート値を調整する必要がある場合があります (サンプルファイルは ***EAP_HOME*/modules/system/layers/base/org/jboss/teiid/client/main/teiid-client-*VERSION*.jar** ファイル内にあります)。

注記

通常のインストールでは、これらの設定を調整する必要がありません。

10.7. Large Object (LOB)

Large Object (LOB) はデータから構成されます。JBoss で使用される主な Large Object ランタイムデータ型は以下の 3 つです。

1. Binary (BLOB)
   - 画像や音声などのマルチメディアオブジェクトを含みます。
2. Character (CLOB)
   - ASCII 文字を含みます。
3. Extensible Markup Language (XML)
   - テキストデータを含みます。

LOB および JBoss

JBoss Data Services Connector API は、LOB への参照を返します (JBoss Data Services サーバーで許可された場合)。JBoss Data Services サーバーまたは JDBC ドライバーは、一度にすべてのデータを取得する代わりにストリームからデータにアクセスできます。これは以下の複数の理由により役に立ちます。

- 結果セットをユーザーに返すときにメモリー使用量を削減します。
- 結果セットで少ないデータを渡すことにより、パフォーマンスを向上します。
- ユーザーが常に LOB データを使用することを前提とする代わりに、必要なときに LOB にアクセスするようにします。
- 固定された JBoss Data Services メモリー使用量内での任意の大きいデータ値の処理を有効にします。

これらの利点は、コネクター自体が一度に LOB 全体をマテリアライズしない場合に実現されます。たとえば、JDBC API は BLOB および CLOB データのストリーミングインターフェースをサポートします。

ソース LOB 値は、一時的な場所にコピーされるのではなく、通常参照によりアクセスされます。ソース LOB がメモリーセーフの方法で返されるようにする必要があります。

LOB は、作成され、ストリーミングされるときに分割されます。各部分のサイズは、クライアントがフェッチするときに設定できます。

キャッシュされた LOB はコピーされ、ソース LOB への参照に依存しません。

Teiid によって作成された一時 LOB は、結果セットまたはステートメントが閉じられるとクリーンアップされます。ステートメントを閉じることではなく、クリーンアップを基にした暗黙的なガベッジコレクションに依存するには、クエリー SELECT teiid_session_set('clean_lobs_onclose', false) を実行し、Teiid セッ

ション変数 clean_lobs_onclose を false に設定します。これは、データソース定義の新しい接続 SQL から実行できます。Designer によって生成された REST VDB など、暗黙的な動作に依存するローカルクライアントがある場合に使用できます。

10.8. LOB に関する考慮事項

管理 CLI を使用してすべての JBoss Data Virtualization 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは、ラージオブジェクト (LOBs) に関連する設定についての追加情報を提供します。

`lob-chunk-size-in-kb`

LOB および XML ドキュメントは JBoss Data Virtualization サーバーから JDBC API にストリーミングされます。通常、これらの値はサーバーメモリーでマテリアライズされず、メモリー不足の問題が回避されます。スタイルシートまたは XQuery を使用する場合は、XML ドキュメント全体をサーバーでマテリアライズする必要があります。XMLQuery 関数または XMLTable 関数を使用し、ドキュメントプロジェクションが適用される場合であっても、大きいドキュメントに対してメモリーの問題が発生することがあります。

LOB は、作成され、ストリーミングされるときに分割されます。各部分の最大サイズは、クライアントがフェッチするときに **`lob-chunk-size-in-kb`** プロパティーで設定できます。

デフォルト値は 100 です。非常に大きい LOB を使用する場合は、結果をストリーミングするラウンドトリップの量を減少するために **`lob-chunk-size-in-kb`** を増加することを検討してください。設定値が大きすぎると、サーバーまたはクライアントでメモリーの問題が発生することがあります。

ソース LOB 値は、一時的な場所にコピーされるのではなく、通常参照によりアクセスされます。したがって、ソース LOB がメモリーセーフの方法で返されるようにする必要があります。これは、LOB の VM メモリーを消費しないソースドライバーベンダーに特に当てはまります。

10.9. 他のパフォーマンスチューニングに関する注意点

管理 CLI を使用してすべての JBoss Data Services 設定に関する情報を見つけることができますが (「JBoss Data Virtualization の設定」 を参照)、このセクションでは **`max-source-rows`** 設定に関する追加情報を提供します。

`max-source-rows`

JBoss Data Services をデプロイメント環境で使用する場合は、ソースが大量の大きなデータを取得することを回避するために **`max-source-rows`** を小さい値 (たとえば、10000) に設定することを検討してください。**`exception-on-max-source-rows`** プロパティーを **`true`** に設定したままにすると、指定された行数よりも多い行数を取得することが試行されたことを示す例外が開発者に通知されます。

パート VI. 参考書

第11章 設定全般

11.1. JBoss Data Virtualization の設定

以下の種類の JBoss Data Virtualization 設定を表示および変更できます。

- バッファーサービス設定
- キャッシュ設定 (結果セットおよび準備された計画キャッシュ設定を含む)
- ランタイムエンジンデプロイヤーの設定
- 承認バリデーターおよびポリシーディサイダーの設定
- トランスポートおよび SSL の設定
- トランスレーターの設定

JBoss Data Virtualization の利用可能なすべての設定を参照するには、管理 CLI 内で以下のコマンドを実行します。

```
/subsystem=teiid:read-resource-description
```

注記

トランスレーターおよびトランスポート(SSL を含む) の設定の詳細については、「管理 CLI を使用したトランスポートおよび SSL の設定の管理」 と 「管理 CLI を使用したトランスレーター設定の管理」 を参照してください。

11.2. 管理 CLI を使用した JBoss Data Virtualization 設定の表示

JBoss Data Virtualization の現在のすべての設定を参照するには、管理 CLI 内で以下のコマンドを実行します。

```
/subsystem=teiid:read-resource
```

特定の JBoss Data Virtualization 設定を表示するには、管理 CLI 内で以下のコマンドを実行します。

```
/subsystem=teiid:read-attribute(name=SETTING_NAME)
```

例:

```
/subsystem=teiid:read-attribute(name=max-active-plans)
```

注記

現行トランスレーターおよびトランスポート(SSL を含む) の設定を確認するには、「管理 CLI を使用したトランスポートおよび SSL の設定の管理」 と 「管理 CLI を使用したトランスレーター設定の管理」 を参照してください。

11.3. 管理 CLI を使用した JBoss Data Virtualization 設定の変更

特定の JBoss Data Virtualization 設定を編集するには、管理 CLI 内で以下のコマンドを実行します。

```
/subsystem=teiid:write-attribute(name=SETTING_NAME, value=VALUE)
```

注記

トランスレーターおよびトランスポート(SSL を含む) の設定を変更する方法について、「管理 CLI を使用したトランスポートおよび SSL の設定の管理」 と 「管理 CLI を使用したトランスレーター設定の管理」 を参照してください。

例:

```
/subsystem=teiid:write-attribute(name=max-active-plans, value=50)
```

管理 CLI の詳細については、Red Hat JBoss Enterprise Application Platform 『管理および設定ガイド』を参照してください。

注記

これらの設定を変更したあとで、サーバーをリロードすることが求められます。これらの設定の変更は、サーバーが再起動するまで反映されません。サーバーは、管理 CLI 内から **reload** コマンドを実行することによりリロードできます。サーバーのリロード後は、管理 CLI 内で作業を続行するために **connect** コマンドを実行して再接続する必要がある場合があります。

11.4. 管理 CLI を使用したトランスポートおよび SSL の設定の管理

JBoss Data Virtualization トランスポート設定を管理するには、基本的な JBoss Data Virtualization 設定に使用されたものと同じコマンドを使用してコマンドで特定のトランスポートを指定します。例を以下に示します。

```
/subsystem=teiid/transport=TRANSPORT_NAME:read-resource
```

現在の JBoss Data Virtualization 設定を出力するために以下のコマンドを実行すると、利用可能なトランスポート名が **transport** 下にリストされます。

```
/subsystem=teiid:read-resource
```

11.5. 管理 CLI を使用したトランスレーター設定の管理

JBoss Data Virtualization トランスレーター設定を管理するには、基本的な JBoss Data Virtualization 設定に使用されたものと同じコマンドを使用してコマンドで特定のトランスレーターを指定します。例を以下に示します。

```
/subsystem=teiid/translator=TRANSLATOR_NAME:read-resource
```

現在の JBoss Data Virtualization 設定を出力するために以下のコマンドを実行すると、利用可能なトランスレーター名が **translator** 下にリストされます。

```
/subsystem=teiid:read-resource
```

11.6. トランスポートセキュリティー認証モード

以下の認証モードが利用可能です。

anonymous

証明書は交換されません。キーストアおよびトラストストアプロパティーの設定は必要ありません。クライアントでは匿名サーバーに接続するために **org.teiid.ssl.allowAnon** を true (デフォルト値) に設定する必要があります。通信は TLS_DH_anon_WITH_AES_128_CBC_SHA SSL 暗号スイートを使用して暗号化されます。これはほとんどのセキュアイントラネットに適切です。

1-way

クライアントに対してサーバーを認証します。サーバーは、サーバーのキーストアに格納されたプライベートキーで署名された証明書を提供します。対応するサーバーのパブリックキーはクライアントのトラストストアに存在する必要があります。

2-way

クライアントとサーバーの相互認証。サーバーは、サーバーのキーストアに格納されたプライベートキーで署名された証明書を提供します。対応するサーバーのパブリックキーはクライアントのトラストストアに存在する必要があります。また、クライアントは、クライアントのキーストアに格納されたプライベートキーで署名された証明書を提供します。対応するクライアントのパブリックキーは、サーバーのトラストストアに存在する必要があります。

注記

キーツールを使用して暗号化キーを生成できますが、最初にパブリックキー暗号化を管理するためにローカルの要件を検討する必要があります。

11.7. JBoss 管理コンソールを使用したコア設定の管理

1. JBoss 管理コンソールにログインします。
2. **Profile** タブを選択します。
3. ナビゲーションツリーから、**Subsystems** → **Teiid** を選択します。

4. 以下のオプションのいずれかを選択します。
   - Query Engine - ここから、コアクエリーエンジンプロパティーを参照および設定できます。
   - Translators - ここから、トランスレーターを参照、追加、および削除できます。
   - Transports - ここから、トランスポートを参照、追加、および削除できます。
5. 必要に応じて設定を参照および変更できます。

注記

ヘルプが必要な場合は、**Need Help?** リンクを選択します。

警告

一部のプロパティーでは、有効にする前にサーバーを再起動する必要があります。

11.8. Red Hat JBoss Data Virtualization で使用されるポート

Red Hat JBoss Data Virtualization は、JBoss Enterprise Application Platform で使用されたポートを継承します。完全なリストについては、https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6/html/Administration_and_Configuration_Guide/Network_Ports_Used_By_JBoss_Enterprise_Application_Platform_62.html を参照してください。

また、ポート 31000 および 35432 もオープンにする必要があります。

11.9. 管理コンソールを使用したデフォルトの JDBC ポートの変更

1. 管理コンソールへログインします。
2. **管理コンソールでの `Socket Binding` パネルへのナビゲート**
   a. A. **スタンドアロンモード**

      コンソールの右上より **`Profile`** タブを選択します。

      B. **ドメインモード**

      i. コンソールの右上より **`Profiles`** タブを選択します。

      ii. 左上のドロップダウンボックスより該当するプロファイルを選択します。

      iii. コンソールの左側にある **Subsystems** メニューを展開します。

   b. コンソールの左側のメニューから **General Configuration** → **Socket Binding** を選択します。
3. **ポート番号の変更**
   a. **`teiid-jdbc`** 設定を選択します。

b. **Edit** ボタンを選択します。

c. **Port** に新しいポート番号を設定します。

d. **Save** を選択します。

11.10. システムプロパティー

一部の動作は、設定ファイルではなくシステムプロパティーで設定できます。JBoss Data Virtualization のシステムプロパティーを設定する通常の場所は *EAP_HOME*/bin/*standalone*.conf です。プロパティー設定の形式は **-D*property*=*value*** です。

表11.1 システムプロパティー

設定	説明	デフォルト値
org.teiid.allowNanInfinity	数値関数が NaN (Not A Number) および +-Infinity を返すことを許可する場合は、true に設定します。これらの値は SQL の仕様に準拠しないことに注意してください。	デフォルトでは false に設定されます。
org.teiid.useValueCache	正規の値キャッシュを有効にする場合は true に設定します。値キャッシュは、同一の値を再使用するためにバッファーメモリーの空き容量が小さくなり、JBoss Data Virtualization が消費するメモリーが削減されるときに動的に使用されます。ただし、キャッシュルックアップに関連する計算コストがある場合、類似しない大量のデータを処理するインストールに対してこの設定を有効にすることは適切ではありません。	デフォルトでは false に設定されます。
org.teiid.ansiQuotedIdentifiers	先頭の識別子部分がない、二重引用符に囲まれた値を文字列リテラルとして処理する以前の動作をエミュレートする場合は false に設定します。この動作は SQL の仕様で期待されません。	デフォルトでは true に設定されます。
org.teiid.subqueryUnnestDefault	サブクエリーの IN 述語および EXISTS 述語を積極的にネスト解除する場合は true に設定します。可能な場合は、述語が従来の結合にネスト解除され、依存結合計画に使用できます。従来の結合が可能でない場合 (NOT IN を使用する場合など) は、利用可能なコスト情報に基いて、セミ結合またはアンチ結合のマージ結合バージョンが考慮されます。	デフォルトでは false に設定されます。

設定	説明	デフォルト値
org.teiid.ODBCPacketSize	ODBC 結果バッファーのターゲットサイズ (バイト単位)。これはハードな最大値ではなく、LOBS 行およびワイド行がさらに大きいバッファーを使用することがあります。	デフォルト値は 307200 です。
org.teiid.decimalAsDouble	固定小数点リテラル (たとえば 1.0) を 10 進数/BigDecimal 値ではなく double 値として解析し、以前のバージョンと同じように整数値用の AVG 関数から double 値を返す場合は true に設定されます。	デフォルトでは false に設定されます。
org.teiid.comparableLobs	JBoss Data Virtualization で BLOB と CLOB の列値を比較することを許可する場合は、true に設定します。ソースタイプメタデータにより、比較をプッシュダウンできるかどうかが決まります。	デフォルトでは false に設定されます。
org.teiid.comparableObject	JBoss Data Virtualization でオブジェクト列値を比較することを許可する場合は、true に設定します。ソースタイプメタデータにより、比較をプッシュダウンできるかどうかが決まります。オブジェクトインスタンスは、**`java.lang.Comparable.compareTo`** を適切に実装することが期待されます。インスタンスオブジェクトが **`Comparable`** でない場合は、**`ClassCastExceptions`** がスローされることがあります。	デフォルトでは false に設定されます。
org.teiid.padSpace	PAD SPACE 照合が使用される場合のように文字列を比較するときは true に設定します。つまり、文字列は比較のために同じ長さにパディングされます。このプロパティーが設定された場合は、**`trimStrings`** トランスレーターオプションを使用する必要がありません。	デフォルトでは false に設定されます。
org.teiid.collationLocale	Java ロケール文字列言語[_country[_varient]] に設定します。ここで、言語、国、および種類は 2 文字のコードです。有効なコードの詳細については、java.util.Locale を参照してください。**`org.teiid.comparableLobs`** が設定されている場合であっても、CLOB 値はロケールコレーターを使用して比較されません。	デフォルトでは設定されません。つまり、Java の通常の (UTF-16) 文字列比較が使用されます。

設定	説明	デフォルト値
org.teiid.clientVdbLoadTimeoutMillis	クライアント (現時点ではローカルクライアントのみ) がアクティブな VDB に接続する場合に例外をスローするまで待機するデフォルトの時間。クライアントは **`waitForLoad`** 接続プロパティーを使用してこの設定を上書きすることがあります。	デフォルト値は 5 分です。
org.teiid.enDateNames	Java デフォルトロケールから名前を返す代わりにシステム関数 **`dayName`** および **`monthName`** に対して英語の月名および曜日名を使用する場合は、true に設定します。このリリースより前は、**`dayName`** と **`monthName`** は常に英語の名前を返していました。	デフォルトでは false に設定されます。
org.teiid.pushdownDefaultNullOrder	ソースのデフォルトの null 順序が異なり、明示的な null 順序がサポートされるときに、null のデフォルトの null 順序を低下させる以前のリリースの動作を真似る場合は、true に設定します。	デフォルトでは false に設定されます。
org.teiid.implicitMultiSourceJoin	複数ソーステーブル間で暗黙的に結合を分割する以前のリリースの動作を無効にする場合は、false に設定します。false に設定された場合は、結合の結果を分割するために **`tbl1.source_name = tbl2.source_name`** などの明示的な述語が必要です。	デフォルトでは true に設定されます。
org.teiid.joinPrefetchBatches	ソースからストリームできる結合処理に対してプリフェッチまたはバッファできるバッチの数を設定します。スローな ALREADY_SORTED ソースが多数のバッチを返す非依存結合に参加する場合は、増加することを検討してください。 ただし、この値により、バッファリングを実行するためにディスクの使用率が増加することがあります。	デフォルト値は 10 です。

設定	説明	デフォルト値
org.teiid.maxStringLength	JBoss Data Virtualization で文字列の名目最大長を設定します。ほとんどの操作では、この値よりも大きい文字列の部分が切り捨てられます。この値を設定することにより、メモリに保持される LOB バイトの最大サイズを調整することもできます。ソースがサポートするよりも大きい文字列値を適切に処理できないことがあることに注意してください。 **警告** 文字列は全体がメモリに保持されます。メモリ不足のエラーが発生することがあるため、この値をあまり高く設定しないでください。	デフォルト値は 4000 です。
org.teiid.calendarTimestampDiff	以前のバージョンの timestampdiff の動作を使用する場合は、false に設定します。古い動作を使用すると、秒よりも大きい間隔に対する timestampdiff のプッシュバージョンと非プッシュバージョン間の結果が異なることがあることに注意してください。	デフォルトでは true に設定されます。

PostgreSQL との互換性を維持するために、以下のプロパティーが提供されます。

表11.2 PostgreSQL との互換性を維持するためのシステムプロパティー

設定	説明	デフォルト値
org.teiid.iso8601Week	ロケールに関係なく、週の計算に ISO 8601 ルールを使用する場合は、true に設定します。true に設定されると、**`dayOfWeek`** 関数が日曜日ではなく月曜日を 1 で開始します。週の関数では、 1 年のその週に 1 年の最初の木曜日が含まれている必要があります。	デフォルトでは false に設定されます。

設定	説明	デフォルト値
org.teiid.backslashDefaultMatchEscape	エスケープが指定されないときに LIKE 述語および SIMILAR TO 述語のデフォルトのエスケープ文字として '\' を使用する場合は true に設定します。それ以外の場合、JBoss Data Virtualization では、ワイルドカード以外の各文字を完全一致文字として処理する SQL 仕様準拠の動作を前提とします。	デフォルトでは false に設定されます。
org.teiid.honorDeclareFetchTxn	false に設定すると、Red Hat JBoss Data Virtualization で処理を必要としなくなるため、UseDeclareFetch カーソルのラッピングの開始/コミットが無視されます。	デフォルトでは false に設定されます。

注記

PostgreSQL との互換性を維持するためのこれらのプロパティーにより、ODBC トランスポートだけでなく JBoss Data Virtualization 全体が影響を受けます。

11.11. Teiid 管理 CLI

AS CLI は、Teiid 用のコマンドラインベースの管理および監視ツールです。AdminShell は、Groovy スクリプト言語へのバインディングや、Teiid と対話するときに必要になることが多い高レベルのメソッドを提供します。多くの場合で、基礎の CLI コマンドを知っていると便利です。以下に、Teiid サーバーを管理するときに便利なコマンドを示します。

VDB の操作

```
deploy adminapi-test-vdb.xml
undeploy adminapi-test-vdb.xml

/subsystem=teiid:restart-vdb(vdb-name=AdminAPITestVDB, vdb-version=1, model-
names=TestModel)

/subsystem=teiid:list-vdbs()
/subsystem=teiid:get-vdb(vdb-name=AdminAPITestVDB,vdb-version=1)
/subsystem=teiid:change-vdb-connection-type(vdb-name=AdminAPITestVDB, vdb-
version=1,connection-type=ANY)

/subsystem=teiid:add-data-role(vdb-name=AdminAPITestVDB, vdb-version=1, data-
role=TestDataRole, mapped-role=test)
/subsystem=teiid:remove-data-role(vdb-name=AdminAPITestVDB, vdb-version=1,
data-role=TestDataRole, mapped-role=test)
```

ソースの操作

```
/subsystem=teiid:add-source(vdb-name=AdminAPITestVDB, vdb-version=1, source-
name=text-connector-test, translator-name=file, model-name=TestModel, ds-
name=java:/test-file)
```

```
/subsystem=teiid:remove-source(vdb-name=AdminAPITestVDB, vdb-version=1,
source-name=text-connector-test, model-name=TestModel)
/subsystem=teiid:update-source(vdb-name=AdminAPITestVDB, vdb-version=1,
source-name=text-connector-test, translator-name=file, ds-
name=java:/marketdata-file)
```

トランスレーターの操作

```
/subsystem=teiid:list-translators()
/subsystem=teiid:get-translator(translator-name=file)
/subsystem=teiid:read-translator-properties(translator-
name=file,type=OVERRIDE)
/subsystem=teiid:read-rar-description(rar-name=file)
```

ランタイムの操作

```
/subsystem=teiid:workerpool-statistics()

/subsystem=teiid:cache-types()
/subsystem=teiid:clear-cache(cache-type=PREPARED_PLAN_CACHE)
/subsystem=teiid:clear-cache(cache-type=QUERY_SERVICE_RESULT_SET_CACHE)
/subsystem=teiid:clear-cache(cache-type=PREPARED_PLAN_CACHE, vdb-
name=AdminAPITestVDB,vdb-version=1)
/subsystem=teiid:clear-cache(cache-type=QUERY_SERVICE_RESULT_SET_CACHE, vdb-
name=AdminAPITestVDB,vdb-version=1)
/subsystem=teiid:cache-statistics(cache-type=PREPARED_PLAN_CACHE)
/subsystem=teiid:cache-statistics(cache-type=QUERY_SERVICE_RESULT_SET_CACHE)

/subsystem=teiid:engine-statistics()

/subsystem=teiid:list-sessions()
/subsystem=teiid:terminate-session(session=sessionid)

/subsystem=teiid:list-requests()
/subsystem=teiid:cancel-request(session=sessionId, execution-id=1)
/subsystem=teiid:list-requests-per-session(session=sessionId)
/subsystem=teiid:list-transactions()

/subsystem=teiid:mark-datasource-available(ds-name=java:/accounts-ds)

/subsystem=teiid:get-query-plan(session=sessionid,execution-id=1)
```

第12章 ディレクトリー構造

12.1. ディレクトリー構造

以下は、Red Hat EAP インスタンス内の Red Hat JBoss Data Virtualization デプロイメントのコンテンツを示しています。

```
/bin
   /scripts
/docs
   /teiid
       /datsources
       /schema
       /examples
/domain
   /configuration
       application-users.properties
       application-roles.properties
/dataVirtualization
/modules
   /system
      /layers
         /dv
           /dv modules
         /base
           /eap modules
/standalone
   /configuration
          standalone.xml
          application-users.properties
          application-roles.properties
```

bin/scripts

このディレクトリーには、インストールおよびユーティリティー CLI スクリプトが含まれます。

docs/teiid

このディレクトリーには、ドキュメント、例、サンプルデータソース XML の一部、およびスキーマファイルが含まれます。

standalone/configuration

- **`standalone.xml`** は、Red Hat JBoss Data Virtualization 向けのマスター設定ファイルです。このファイルは、標準的な Red Hat JBoss EAP ウェブプロファイルサブシステムに加えて JBoss Data Virtualization サブシステムの設定を管理します。
- **`application-users.properties`** と **`application-roles.properties`** は、デフォルトのセキュリティードメインを使用して許可ユーザーとそのロールを定義します。ユーザーを追加するにはこれらのファイルを編集します。異なるセキュリティードメインを使用する場合は、メイン設定ファイルでセキュリティードメイン詳細を変更します。

domain/configuration

- **`application-users.properties`** と **`application-roles.properties`** は、デフォルトのセキュリティードメインを使用して Red Hat JBoss Data Virtualization で許可ユーザーと

そのロールを定義します。ユーザーを追加するにはこれらのファイルを編集します。異なるセキュリティードメインを使用する場合は、メイン設定ファイルでセキュリティードメイン詳細を変更します。

/modules/system/layers/dv/org/jboss/teiid/*

このディレクトリーでは、JBoss EAP 用の Red Hat JBoss Data Virtualization モジュールが定義されます。

/modules/system/layers/dv/org/jboss/teiid/client

このディレクトリーには、JBoss Data Virtualization クライアントライブラリーが含まれます。また、Red Hat JBoss Data Virtualization JDBC ドライバー JAR ファイルである **`teiid-[VERSION].Final-jdbc.jar`** と、JBoss Data Virtualization Hibernate ダイアレクトを含む **`teiid-hibernate-dialect-[VERSION].Final.jar`** が含まれます。

{standalone,domain}/tmp/teiid

このディレクトリーには、Red Hat JBoss Data Virtualization により作成された一時ファイルが含まれます。ほとんどの場合、これらのファイルはバッファーマネージャーにより作成されます。また、これらのファイルは VM の再起動では不要です。Red Hat JBoss Data Virtualization LOB 値を作成する場合 (たとえば、SQL/XML を使用)、通常、8KBのメモリーサイズの許容値が超えると、LOB ごとに 1 つのファイルが作成されます。使用量が大きい場合は、定期的に最適化するパーティションでバッファディレクトリーを参照することを検討してください。

{standalone,domain}/data/teiid-data

このディレクトリーには、キャッシュ済み VDB メタデータファイルが含まれます。これらのファイルは手動で編集しないでください。

dataVirtualization

このディレクトリーには、JDBC ドライバー、adminshell、ModeShape VDB および一部の WAR が含まれています。

付録A 改訂履歴

改訂 6.30-12.1	**Mon Jan 16 2017**	**Terry Chuang**

翻訳ファイルを XML ソースバージョン 6.30-12 と同期

改訂 6.30-12	**Wed Oct 12 2016**	**David Le Sage**

バージョン 6.3 用に更新。

改訂 6.2.0-15	**Wed Sep 2 2015**	**David Le Sage**

6.2 用に更新