Red Hat JBoss Enterprise Application Platform 6.4

管理および設定ガイド

Red Hat JBoss Enterprise Application Platform 6 向け

Last Updated: 2023-02-21

Red Hat JBoss Enterprise Application Platform 6.4 管理および設定ガイド

Red Hat JBoss Enterprise Application Platform 6 向け

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

法律上の通知

概要

本書は、Red Hat JBoss Enterprise Application Platform 6 およびそのパッチリリースに関する管理および設定ガイドです。

目次

第1章 はじめに

1.1. RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6

Red Hat JBoss Enterprise Application Platform 6(JBoss EAP 6)は、オープン標準に構築されたミドルウェアプラットフォームで、Java Enterprise Edition 6 仕様に準拠します。JBoss Application Server 7 と高可用性クラスタリング、メッセージング、分散キャッシングなどのテクノロジーを統合します。

JBoss EAP 6 には、必要な場合にのみサービスを有効にできる新しいモジュラー構造が含まれ、起動速度が改善されます。

管理コンソールと管理コマンドラインインターフェースにより、XML 設定ファイルの編集が不必要になり、タスクをスクリプト化および自動化する機能が追加されました。

また、JBoss EAP 6 には、セキュアでスケーラブルな Java EE アプリケーションの迅速な開発を可能にする API と開発フレームワークが含まれます。

バグの報告

1.2. JBOSS EAP 6 の機能

表1.1 JBoss EAP 6 の機能

機能	説明
Java 証明書	認定された Java Enterprise Edition 6 の Full Profile と Web Profile。
管理対象ドメイン	• 複数のサーバーインスタンスおよび物理ホストを一元管理し、スタンドアロンサーバーで単一のサーバーインスタンスを使用することを可能にします。 • 設定、デプロイメント、ソケットバインディング、モジュール、拡張機能、およびシステムプロパティーをサーバーグループごとに管理します。 • アプリケーションのセキュリティー (セキュリティードメインを含む) の管理を一元化および簡略化します。
管理コンソールおよび管理 CLI	新しいドメインまたはスタンドアロンサーバー管理インターフェースです。XML 設定ファイルの編集は不要になりました。管理 CLI には、管理タスクをスクリプト化および自動化できるバッチモードも含まれています。

機能	説明
簡素化されたディレクトリーのレイアウト	**modules** ディレクトリーにすべてのアプリケーションサーバーモジュールが含まれるようになりました。共通ディレクトリーおよびサーバー固有の **lib** ディレクトリーは非推奨になりました。**ドメイン** ディレクトリーと **スタンドアロン** ディレクトリーには、それぞれドメインデプロイメントとスタンドアロンデプロイメントのアーティファクトおよび設定ファイルが含まれます。
モジュラークラスローディングメカニズム	モジュールは必要に応じてロードおよびアンロードされます。これにより、パフォーマンスが向上します。
簡略化されたデータソース管理	データベースドライバーは他のサービスと同様にデプロイされます。さらに、データソースは管理コンソールまたは管理 CLI で直接作成および管理されます。
リソース使用の削減と効率化	JBoss EAP 6 はより少ないシステムリソースを使用し、以前のバージョンよりも効率的に使用します。その他の利点の他にも、JBoss EAP 6 は JBoss EAP 5 よりも起動および停止します。

バグの報告

1.3. JBOSS EAP 6 の操作モード

JBoss EAP 6 は、JBoss EAP 6 インスタンスに対してスタンドアローンサーバーと管理対象ドメインの 2 つの操作モードを提供します。

2 つのモードはサーバーの管理方法が異なりますが、エンドユーザーの要求に対応する能力ではありません。高可用性(HA)クラスター機能は、どちらの操作モードでも利用できることに注意してください。スタンドアロンサーバーのグループは、HA クラスターを形成するように設定できます。

バグの報告

1.4. スタンドアロンサーバー

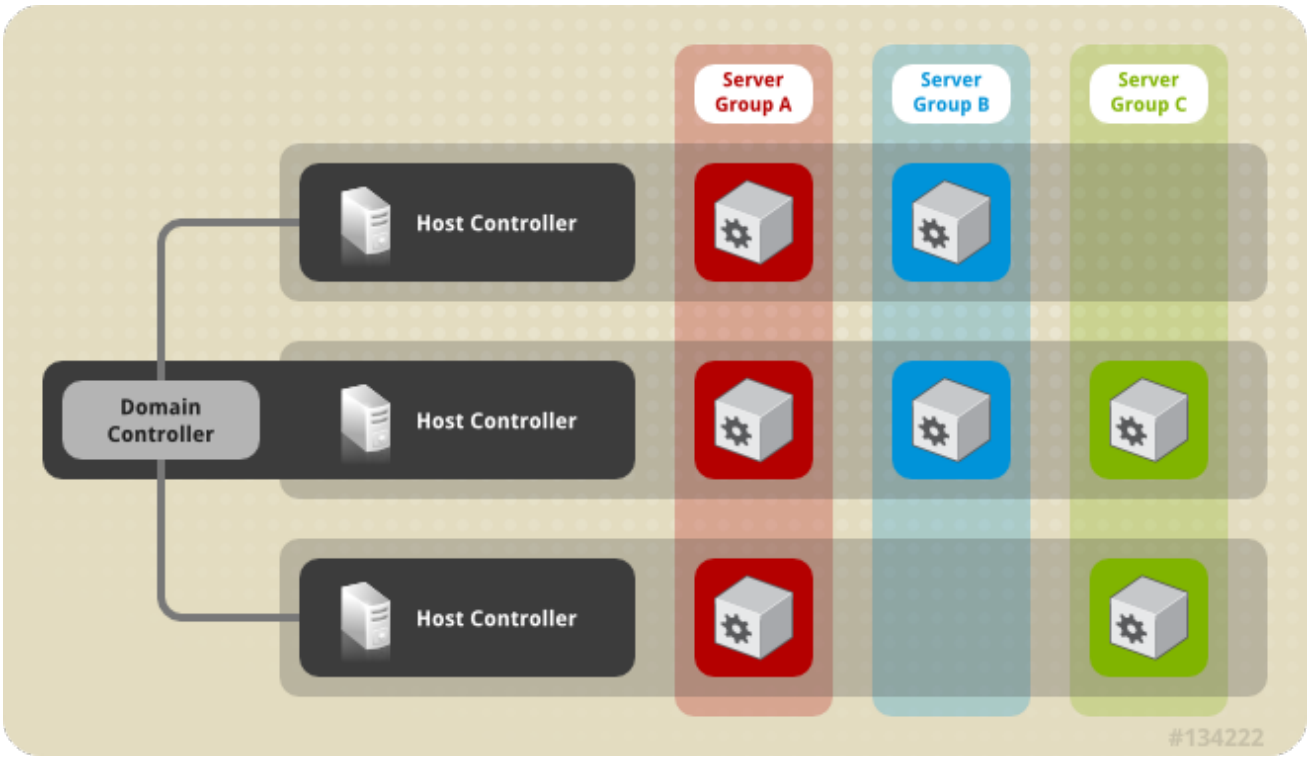
スタンドアロンサーバーモードは独立したプロセスで、以前のバージョンの JBoss EAP で唯一存在した実行モードと似ています。

スタンドアロンサーバーとして実行する JBoss EAP のインスタンスは単一インスタンスのみですが、オプションとしてクラスター化された設定で実行することも可能です。

バグの報告

1.5. 管理対象ドメイン

図1.1 管理対象ドメインを表す図



管理対象ドメイン操作モードでは、単一の制御点から複数の JBoss EAP 6 インスタンスを管理できます。

一元管理された JBoss EAP 6 サーバーは、ドメインのメンバーと呼ばれます。ドメインの JBoss EAP 6 インスタンスはすべて共通の管理ポリシーを共有します。

各ドメインは 1 つの<firstterm>ドメインコントローラー</firstterm>、1 つ以上の<firstterm>ホストコントローラー</firstterm>、およびホスト毎に 0 個以上のサーバーグループによって構成されます。

ドメインコントローラーは、ドメインが制御される中心点です。ドメインの管理ポリシーに従って各サーバーが設定されるようにします。ドメインコントローラーはホストコントローラーでもあります。

ホストコントローラーは、**domain.sh** または **domain.bat** スクリプトが実行される物理ホストまたは仮想ホストです。ホストコントローラーは、ドメイン管理タスクをドメインコントローラーに委譲するように設定されています。

各ホスト上のホストコントローラーはドメインコントローラーと対話し、ホスト上で実行されているアプリケーションサーバーインスタンスのライフサイクルを制御し、ドメインコントローラーがそれらを管理できるようにします。各ホストに複数のサーバーグループが含まれるようにすることが可能です。

サーバーグループとは、JBoss EAP 6 が JBoss EAP 6 がインストールされ、1つとして管理および設定されるサーバーインスタンスのセットです。ドメインコントローラーはサーバーグループにデプロイされたアプリケーションとその設定を管理します。そのため、サーバーグループの各サーバーは同じ設定とデプロイメントを共有します。

同じ物理システム上の同じ JBoss EAP 6 インスタンス内で、単一のドメインコントローラー、単一のホストコントローラー、および複数のサーバーを実行できます。

ホストコントローラーは特定の物理（または仮想）ホストに割り当てられます。異なる設定を使用する場合は、同じハードウェア上で複数のホストコントローラーを実行でき、ポートとその他のリソースが競合しないようにします。

バグの報告

1.6. ドメインコントローラー

ドメインコントローラーは、ドメインの集中管理点として動作する JBoss EAP 6 サーバーインスタンスです。1つのホストコントローラーインスタンスがドメインコントローラーとして動作するよう設定されます。

ドメインコントローラーの主な役割は次のとおりです。

- ドメインの集中管理ポリシーを維持する。
- すべてのホストコントローラーが現在のコンテンツを認識するようにする。
- 実行中のすべての JBoss EAP 6 インスタンスがこのポリシーに従って設定されるよう、ホストコントローラーをサポートする。

デフォルトでは、集中管理ポリシーは **domain/configuration/domain.xml** ファイルに保存されます。このファイルは、ドメインコントローラーのホストのファイルシステムに展開した JBoss EAP 6 インストールファイルにあります。

domain.xml ファイルは、ドメインコントローラーとして実行するよう設定されたホストコントローラーの **domain/configuration/** ディレクトリーに配置する必要があります。このファイルは、ドメインコントローラーとして実行されないホストコントローラーへのインストールには必要ありません。このようなサーバーに **domain.xml** ファイルが存在すると、害はありません。

domain.xml ファイルには、ドメインのサーバーインスタンスで実行できるプロファイル設定が含まれています。プロファイル設定には、プロファイルを構成するさまざまなサブシステムの詳細設定が含まれます。ドメイン設定には、ソケットグループの定義とサーバーグループの定義も含まれます。

バグの報告

1.7. ドメインコントローラーの検索およびフェールオーバー

管理対象ドメインの設定時、ドメインコントローラーに接続するために必要な情報を使用して各ホストコントローラーを設定する必要があります。JBoss EAP 6 では、ドメインコントローラーを検索する複数のオプションを使用して各ホストコントローラーを設定できます。ホストコントローラーは、適切なオプションが見つかるまでオプションのリストを繰り返し処理します。

これにより、バックアップドメインコントローラーのコンタクト情報とともにホストコントローラーを事前設定できます。プライマリードメインコントローラーに問題がある場合は、バックアップホストコントローラーをマスターに昇格でき、昇格後にホストコントローラーを新規マスターに自動的にフェイルオーバーできます。

以下は、ドメインコントローラー検索の複数のオプションを持つホストコントローラーを設定する方法の例になります。

例1.1 複数のドメインコントローラーオプションで設定されたホストコントローラー

```
<domain-controller>
    <remote security-realm="ManagementRealm">
        <discovery-options>
            <static-discovery name="primary" host="172.16.81.100" port="9999"/>
            <static-discovery name="backup" host="172.16.81.101" port="9999"/>
        </discovery-options>
    </remote>
</domain-controller>
```

静的検出オプションには、以下の必須属性が含まれます。

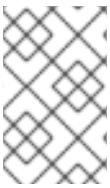
name
: このドメインコントローラー検索オプションの名前。

host
: リモートドメインコントローラーのホスト名。

port
: リモートドメインコントローラーのポート。

上記の例では、最初の検出オプションが指定されることが予想されます。2 つ目のオプションはフフェイルオーバーの状況で使用される可能性があります。

プライマリードメインコントローラーで問題が発生した場合、**--backup** オプションで開始されたホストコントローラーを昇格してドメインコントローラーとして動作することができます。



注記

--backup オプションを指定してホストコントローラーを起動すると、そのコントローラーはドメイン設定のローカルコピーを維持します。この設定は、ホストコントローラーがドメインコントローラーとして動作するよう再設定された場合に使用されます。

手順1.1 ホストコントローラーを昇格してドメインコントローラーにする

1. 元のドメインコントローラーが停止した状態であることを確認します。
2. 管理 CLI を使用して、新しいドメインコントローラーとなるホストコントローラーへ接続します。
3. 以下のコマンドを実行してホストコントローラーを設定し、新しいドメインコントローラーとして動作するようにします。

```
/host=HOST_NAME:write-local-domain-controller
```

4. 以下のコマンドを実行し、ホストコントローラーをリロードします。

```
reload --host=HOST_NAME
```

2. で選択したホストコントローラーがドメインコントローラーとして動作するようになります。

バグの報告

1.8. ホストコントローラー

ホストコントローラーは、**domain.sh** または **domain.bat** スクリプトをホストで実行する際に起動します。

ホストコントローラーの主な役割はサーバーを管理することです。ドメイン管理タスクを委譲し、ホスト上で実行される個別のアプリケーションサーバープロセスを開始および停止します。

ホストコントローラーはドメインコントローラーと対話し、サーバーとドメインコントローラー間の通信を管理できるようにします。ドメインの複数のホストコントローラーは 1 つのドメインコントローラーのみと対話できます。そのため、単一のドメインモード上で実行されるホストコントローラーおよびサーバーインスタンスはすべて単一のドメインコントローラーを持ち、同じドメインに属する必要があります。

デフォルトでは、各ホストコントローラーは、ホストのファイルシステムの未展開の JBoss EAP 6 インストールファイルにある domain/configuration/host.xml ファイルから設定を読み取ります。host.xml ファイルには、特定のホストに固有する以下の設定情報が含まれます。

- このインストールから実行される JBoss EAP 6 インスタンスの名前。
- 次の設定のいずれか。
  - ホストコントローラーがドメインコントローラーへ通知して、ホストコントローラー自体を登録し、ドメイン設定へアクセスする方法。
  - リモートドメインコントローラーの検索および通知方法。
  - ホストコントローラーはドメインコントローラーとして動作する。
- ローカルの物理インストールに固有する設定。たとえば、domain.xml で宣言された名前付きインターフェース定義は、host.xml の実際のマシン固有の IP アドレスにマップできます。また、domain.xml の抽象パス名を host.xml の実際のファイルシステムパスにマッピングできます。

バグの報告

1.9. サーバーグループ

サーバーグループとは、1 つのグループとして管理および設定される複数のサーバーインスタンスのことです。管理対象ドメインでは、各アプリケーションサーバーインスタンスは唯一のメンバーである場合でも 1 つのサーバーグループに属します。グループのサーバーインスタンスは同じプロファイル設定とデプロイされたコンテンツを共有します。

ドメインコントローラーとホストコントローラーは、ドメインにある各サーバーグループのすべてのインスタンスに標準設定を強制します。

ドメインを複数のサーバーグループで構成できます。異なるサーバーグループを異なるプロファイルやデプロイメントで設定できます。ドメインは、異なるサービスを提供する異なるサーバー層で設定できます。

異なるサーバーグループが同じプロファイルやデプロイメントを持つこともできます。これにより、最初のサーバーグループでアプリケーションがアップグレードされた後に 2 つ目のサーバーグループでアプリケーションがアップデートされるアプリケーションのローリングアップグレードが可能になり、サービスの完全停止を防ぎます。

以下はサーバーグループ定義の例になります。

例1.2 サーバーグループ定義

```
<server-group name="main-server-group" profile="default">
 <socket-binding-group ref="standard-sockets"/>
 <deployments>
  <deployment name="foo.war_v1" runtime-name="foo.war"/>
```

```
    <deployment name="bar.ear" runtime-name="bar.ear"/>
  </deployments>
</server-group>
```

サーバーグループに含まれる必須の属性は次のとおりです。

- name: サーバーグループの名前。
- profile: サーバーグループのプロファイル名。
- socket-binding-group: グループのサーバーに使用されるデフォルトのソケットバインディンググループ。この名前は、**host.xml** のサーバーごとに上書きできます。ただし、これはすべてのサーバーグループに必須要素であり、存在しない場合はドメインを開始できません。

サーバーグループに含まれる任意の属性は次のとおりです。

- deployments: グループのサーバー上にデプロイするデプロイメントコンテンツ。
- system-properties: グループのサーバーに設定するシステムプロパティー。
- jvm: グループのすべてのサーバーに対するデフォルトの JVM 設定。ホストコントローラーはこれらの設定を **host.xml** の他の設定とマージし、サーバーの JVM を起動するために使用される設定を作成します。
- socket-binding-port-offset: ソケットバインディンググループによって提供されたポート値に追加するデフォルトのオフセット。
- management-subsystem-endpoint: **true** に設定され、Remoting サブシステムからエンドポイントを使用してサーバーグループに属するサーバーがホストコントローラーに接続し直します（Remoting サブシステムが機能するためにはサブシステムが存在する必要があります）。

バグの報告

1.10. JBOSS EAP 6 プロファイル

以前のバージョンの JBoss EAP で使用されたプロファイルの概念は使用されなくなりました。JBoss EAP 6 は、少数の設定ファイルを使用してその設定に関する情報をすべて保持するようになりました。

モジュールとドライバーが必要に応じて読み込まれるようになりました。そのため、以前のバージョンの JBoss EAP 6 で使用されていたデフォルトのプロファイルの概念により、サーバーをより効率的に起動するのではなく、適用されません。

デプロイメント時に、サーバーまたはドメインコントローラーによってモジュールの依存関係が決定、順序付け、解決され、正しい順序でロードされます。モジュールは、デプロイメントが必要なくなったときにアンロードされます。

設定からサブシステムを削除して、モジュールを無効にしたり、ドライバーやその他のサービスを手作業でアンロードしたりできます。ただし、ほとんどの場合、これは必要ありません。どのアプリケーションもモジュールを使用しない場合はロードされません。

バグの報告

1.11. 異なるバージョンのサーバーの管理

注記

異なるバージョンの JBoss EAP サーバーを管理するには、JBoss EAP の最新リリースがドメインコントローラーとして動作する必要があります。

- JBoss EAP スキーマは異なるバージョンを使用します。そのため、新しいバージョンの JBoss EAP ドメインコントローラーには、下位バージョンの JBoss EAP ホストを制御する問題は必要ありませんが、**domain.xml** は使用中のすべてのバージョンが 最も古いもの である必要があります。
- クラスターがある場合、クラスターのすべてのメンバーサーバーは同じバージョンの JBoss EAP に属する必要があります。
- ドメインのすべてのホストには、Process Controller、Host Controller、および managed server などの Java プロセスが複数存在します。これらの Java プロセスは JBoss EAP の同じインストールから起動する必要があるため、同じバージョンが使用されます。

警告

しかし、JBoss EAP 6.3 のドメインコントローラーが JBoss EAP 6.2 以上のスレーブを管理する場合は、若干非互換性があります。この問題を修正する必要があります。[named-formatter] 属性はターゲットモデルバージョンで認識されず、古い属性に置き換える必要があります。詳細は以下を参照してください。
https://access.redhat.com/solutions/1238073

バグの報告

第2章 アプリケーションサーバー管理

2.1. JBOSS EAP ドキュメントの規則

本ガイドのすべてのインスタンスは、使用するインストール方法によって異なる JBoss EAP ルートインストールディレクトリーを参照します。

zip インストール方法

EAP_HOME は、JBoss EAP ZIP ファイルが抽出されたディレクトリーを参照します。

インストーラーメソッド

EAP_HOME は、JBoss EAP のインストールを選択したディレクトリーを参照します。

RPM インストール方法

EAP_HOME は、/usr/share/jbossas ディレクトリーを参照します。

注記

この表記法 は、JBoss EAP で説明されている規則と同じ規則に従って、JBoss Warehouse のインストール場所を参照するために使用されます。

バグの報告

2.2. JBOSS EAP 6 の起動および停止

2.2.1. JBoss EAP 6 の起動

JBoss EAP は、スタンドアロンサーバーまたは管理対象ドメイン2 つのモードのいずれかで実行され、Red Hat Enterprise Linux と Microsoft Windows Server の2 つのプラットフォームでサポートされます。JBoss EAP を起動するコマンドは、基礎となるプラットフォームと目的のモードによって異なります。

表2.1 JBoss EAP を起動するコマンド

Operating System	スタンドアロンサーバー	管理対象ドメイン
Red Hat Enterprise Linux	***EAP_HOME*/bin/standalone.sh**	***EAP_HOME*/bin/domain.sh**
Microsoft Windows Server	***EAP_HOME*\bin\standalone.bat**	***EAP_HOME*\bin\domain.bat**

JBoss EAP 6 の起動方法に関する詳細は、「JBoss EAP 6 をスタンドアロンサーバーとして起動」および「JBoss EAP 6 を管理対象ドメインとして起動」を参照してください。

バグの報告

2.2.2. JBoss EAP 6 をスタンドアロンサーバーとして起動

概要

ここでは、JBoss EAP 6 をスタンドアロンサーバーとして起動する手順を説明します。

手順2.1 プラットフォームサービスをスタンドアロンサーバーとして起動

1. *Red Hat Enterprise Linux の場合*
   ***EAP_HOME/bin/standalone.sh**コマンドを実行します。*
2. *Microsoft Windows Server の場合*
   ***EAP_HOME\bin\standalone.bat**コマンドを実行します。*
3. *他のパラメーターを指定する (任意)*
   *起動スクリプトで利用可能なパラメーターの一覧を表示するには、**-h** パラメーターを使用します。*

結果

JBoss EAP 6 スタンドアロンサーバーインスタンスが起動します。

バグの報告

2.2.3. 1 台のマシンで複数の JBoss EAP スタンドアロンサーバーを実行

概要

このトピックでは、1 台のマシンで複数の JBoss EAP スタンドアロンサーバーを実行する手順を説明します。

手順2.2 1 台のマシンで JBoss EAP スタンドアロンサーバーの複数のインスタンスを実行

1. *各スタンドアロンサーバーの **EAP_HOME/standalone/** ディレクトリーを直接EAP_HOME/ の下に作成します。たとえば、スタンドアロンサーバーの **node1** および **node2** のディレクトリーを作成するには、以下のコマンドを入力します。*

```
$ cd EAP_HOME
$ cp -a ./standalone ./node1
$ cp -a ./standalone ./node2
```

2. *ノード名、IP アドレス、サーバーディレクトリー、オプションのサーバー設定ファイル、および任意のポートオフセットを指定して、各 JBoss EAP スタンドアロンインスタンスを起動します。このコマンドは、以下の構文を使用します。*

```
$ ./bin/standalone.sh -Djboss.node.name=UNIQUE_NODENAME -
Djboss.server.base.dir=EAP_HOME/NODE_DIRECTORY -b IP_ADDRESS -
bmanagement MGMT_IP_ADDRESS --server-config=SERVER_CONFIGURATION_FILE
-Djboss.socket.binding.port-offset=PORT_OFFSET
```

   a. *この例では、**node1**を起動します。*

```
$ cd EAP_HOME
$ ./bin/standalone.sh -Djboss.node.name=node1 -
Djboss.server.base.dir=EAP_HOME/node1 -b 10.10.10.10 -bmanagement 127.0.0.1
```

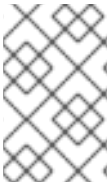
b. ***node2*** *を起動する例は、マシンが複数の* ***IP*** *アドレスをサポートするかどうかによって異なります。*

- *マシンが複数の* ***IP*** *アドレスをサポートする場合、以下のコマンドを使用します。*

```
$ cd EAP_HOME
$ ./bin/standalone.sh -Djboss.node.name=node2 -
Djboss.server.base.dir=EAP_HOME/node2 -b 10.10.10.40 -bmanagement
127.0.0.40
```

- *マシンが複数の* ***IP*** *アドレスをサポートしない場合は、* ***jboss.socket.binding.port-offset*** *プロパティーを指定してポートの競合を避ける必要があります。*

```
$ cd EAP_HOME
$ ./bin/standalone.sh -Djboss.node.name=node2 -
Djboss.server.base.dir=EAP_HOME/node2 -b 10.10.10.10 -bmanagement
127.0.0.1 -Djboss.socket.binding.port-offset=100
```



注記

2 つのノードを同時に管理する場合、または同じ設定を持つ 2 つのノードを管理する場合は、スタンドアロンサーバーを実行するのではなく、管理対象ドメインで実行することが推奨されます。

バグの報告

2.2.4. JBoss EAP 6 を管理対象ドメインとして起動

操作の順序

ドメイン内のサーバーグループのスレーブサーバーを起動する前にドメインコントローラーを起動する必要があります。最初に、最初に、関連するホストコントローラーと、ドメインに関連付けられたその他のホストで使用します。

手順2.3 プラットフォームサービスを管理対象ドメインとして起動

1. *Red Hat Enterprise Linux の場合*
   EAP_HOME/bin/domain.sh*コマンドを実行します。*

2. *Microsoft Windows Server の場合*
   EAP_HOME\bin\domain.bat*コマンドを実行します。*

3. *他のパラメーターを起動スクリプトに渡す (任意)*
   起動スクリプトで利用可能なパラメーターの一覧を表示するには、 ***-h*** *パラメーターを使用します。*

結果

JBoss EAP 6 管理対象ドメインインスタンスが起動します。

バグの報告

2.2.5. 管理対象ドメインのホストの名前設定

概要

管理対象ドメインで実行されている各ホストには一意な名前を付ける必要があります。管理を容易にし、複数のホストで同じホスト設定ファイルを使用できるようにするために、以下の優先順位を用いてホスト名が決定されます。

1. *設定された場合、* ***host.xml*** *設定ファイルのホスト要素 名 属性。*
2. ***jboss.host.name*** *システムプロパティーの値。*
3. ***jboss.qualified.host.name*** *システムプロパティーの最後のピリオド(.)の後に続く値。最後のピリオドがない場合は全体の値。*
4. ***POSIX*** *ベースのオペレーティングシステムでは、* ***HOSTNAME*** *環境変数のピリオド(.)の後に続く値。Microsoft Windows の* ***COMPUTERNAME*** *環境変数、最後のピリオドがない場合は全体の値。*

環境変数の設定方法は、お使いのオペレーティングシステムのドキュメントを参照してください。システムプロパティーの設定方法は、「管理 CLI を使用したシステムプロパティーの設定」を参照してください。

本トピックでは、システムプロパティーまたはハードコードされた名前を使用して、設定ファイルのホストの名前を設定する方法について説明します。

手順2.4 システムプロパティーを用いたホスト名の設定

1. *編集するホスト設定ファイル（例：* ***host.xml*** *）を開きます。*
2. *以下のように、ファイルで* ***host*** *要素を見つけます。*

```
<host name="master" xmlns="urn:jboss:domain:1.6">
```

3. *これが存在する場合は、* ***name="HOST_NAME"*** *属性宣言を削除します。* ***host*** *要素は以下のようになります。*

```
<host xmlns="urn:jboss:domain:1.6">
```

4. *以下のように、* ***-Djboss.host.name*** *引数を渡すサーバーを起動します。*

```
-Djboss.host.name=HOST_NAME
```

手順2.5 特定の名前を用いたホスト名の設定

1. *以下の構文を使用して、* ***JBoss EAP*** *スレーブホストを起動します。*

```
bin/domain.sh --host-config=HOST_FILE_NAME
```

以下に例を示します。

```
bin/domain.sh --host-config=host-slave01.xml
```

2. *管理* ***CLI*** *を起動します。*
3. *以下の構文を使用してホスト名を置き換えます。*

```
/host=EXISTING_HOST_NAME:write-
attribute(name="name",value=UNIQUE_HOST_NAME)
```

以下に例を示します。

```
/host=master:write-attribute(name="name",value="host-slave01")
```

以下の結果が表示されるはずです。

```
"outcome" => "success"
```

これにより、host **-slave01.xml** ファイルのホスト名属性が以下のように変更されます。

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

4. この処理を完了するには、変更前のホスト名を使用してサーバー設定をリロードする必要があります。

```
reload --host=EXISTING_HOST_NAME
```

以下に例を示します。

```
reload --host=master
```

バグの報告

2.2.6. 2 台のマシンでの管理対象ドメインの作成

注記

この例の実行には、ファイアウォールの設定が必要になることがあります。

1 台のマシンがドメインコントローラーで、別のマシンがホストである 2 台のマシンで管理対象ドメインを作成できます。詳細は、「ドメインコントローラー」を参照してください。

- IP1 = ドメインコントローラーの IP アドレス (マシン 1)
- IP2 = ホストの IP アドレス (マシン 2)

手順2.6 2 台のマシンで管理対象ドメインを作成

1. マシン 1 での作業
   a. **add-user.sh** スクリプトを使用して管理ユーザーを追加します。たとえば、ホストがドメインコントローラーを認証できるため、**slave01** です。**add-user** 出力の **SECRET_VALUE** をメモします。
   b. 専用のドメインコントローラー向けに事前設定された **host-master.xml** 設定ファイルを使用してドメインを起動します。
   c. **-bmanagement=$IP1** を使用して、ドメインコントローラーを他のマシンが見えるようにします。
      ■

```
EAP_HOME/bin/domain.sh --host-config=host-master.xml -bmanagement=$IP1
```

2. マシン2 での作業

   a. **EAP_HOME/domain/configuration/host-slave.xml** ファイルをユーザークレデンシャルで更新します。

```
<?xml version='1.0' encoding='UTF-8'?>
      <host xmlns="urn:jboss:domain:1.6" name="slave01">
      <!-- add user name here -->
       <management>
         <security-realms>
           <security-realm name="ManagementRealm">
             <server-identities>
               <secret value="$SECRET_VALUE" />
               <!-- use secret value from add-user.sh output-->
             </server-identities>
             ...
```

   b. ホストを起動します。

```
EAP_HOME/bin/domain.sh --host-config=host-slave.xml  -
Djboss.domain.master.address=$IP1 -b=$IP2
```

3. ドメインを管理します。
   CLI を使用する場合:

```
EAP_HOME/bin/jboss-cli.sh -c --controller=$IP1
```

   Web コンソールを使用する場合:

```
http://$IP1:9990
```

   サーバーのインデックスページへアクセスします。

```
http://$IP2:8080/
http://$IP2:8230/
```

バグの報告

2.2.7. 1 台のマシンで管理対象ドメインを作成

jboss.domain.base.dir プロパティーを使用すると、複数のホストコントローラーを 1 台のマシンで実行できます。

重要

複数の **JBoss EAP** ホストコントローラーを 1 台のマシン上でシステムサービスとして設定することはサポートされません。

手順2.7 1 台のマシンで複数のホストコントローラーを実行する

1. ドメインコントローラーの **EAP_HOME/domain** ディレクトリーをコピーします。

```
cp -r EAP_HOME/domain /path/to/domain1
```

2. ホストコントローラーの **EAP_HOME/domain** ディレクトリーをコピーします。

```
cp -r EAP_HOME/domain /path/to/host1
```

3. **/path/to/domain1** を使用してドメインコントローラーを起動します。

```
EAP_HOME/bin/domain.sh --host-config=host-master.xml -
Djboss.domain.base.dir=/path/to/domain1
```

4. **/path/to/host1** を使用してホストコントローラーを起動します。

```
EAP_HOME/bin/domain.sh --host-config=host-slave.xml -
Djboss.domain.base.dir=/path/to/host1 -Djboss.domain.master.address=IP_ADDRESS
-Djboss.management.native.port=PORT
```

結果

このように起動された各インスタンスは、ベースインストールディレクトリー（**EAP_HOME**/モジュール/）の残りのリソースを共有しますが、**jboss.domain.base.dir** で指定されたディレクトリーからドメイン設定を使用します。

バグの報告

2.2.8. 代替設定を用いた JBoss EAP 6 の起動

設定ファイルを指定しない場合、サーバーはデフォルトファイルで起動します。

設定を手動で指定することもできます。このプロセスは、管理対象ドメインまたはスタンドアロンサーバーとオペレーティングシステムのどちらを使用しているかによって異なります。

前提条件

- 代替の設定ファイルを使用する前に、デフォルト設定をテンプレートとして使用して準備します。
- 管理対象ドメインの場合、代替の設定ファイルは **EAP_HOME/domain/configuration/** ディレクトリーに保存されます。
- スタンドアロンサーバーの場合、代替の設定ファイルは **EAP_HOME/standalone/configuration/** ディレクトリーに保存されます。

設定例

設定例は **EAP_HOME/docs/examples/configs/** ディレクトリーに含まれています。これらの例を使用して、クラスタリングやトランザクション **XTS API** などの機能を有効にします。

代替設定でインスタンスの起動

スタンドアロンサーバー

*スタンドアロンサーバーの場合は、**--server-config** スイッチを使用して設定ファイルを指定します。設定ファイルは **EAP_HOME/standalone/configuration/** ディレクトリーにあり、このディレクトリーに対する相対パスを指定する必要があります。*

> *例2.1 Red Hat Enterprise Linux のスタンドアロンサーバーに別の設定ファイルを使用*
>
> ```
> [user@host bin]$ ./standalone.sh --server-config=standalone-alternate.xml
> ```
>
> *この例では、**EAP_HOME/standalone/configuration/standalone-alternate.xml** 設定ファイルを使用します。*

> *例2.2 Microsoft Windows Server のスタンドアロンサーバーに別の設定ファイルを使用*
>
> ```
> C:\EAP_HOME\bin> standalone.bat --server-config=standalone-alternate.xml
> ```
>
> *この例では、**EAP_HOME\standalone\configuration\standalone-alternative.xml** 設定ファイルを使用します。*

管理対象ドメイン

*管理対象ドメインの場合は、**--domain-config** スイッチを使用して設定ファイルを指定します。設定ファイルは **EAP_HOME/domain/configuration/** ディレクトリーにあり、そのディレクトリーに対する相対パスを指定する必要があります。*

> *例2.3 Red Hat Enterprise Linux の管理対象ドメインに別の設定ファイルを使用*
>
> ```
> [user@host bin]$ ./domain.sh --domain-config=domain-alternate.xml
> ```
>
> *この例では、**EAP_HOME/domain/configuration/domain-alternate.xml** 設定ファイルを使用します。*

> *例2.4 Microsoft Windows Server の管理対象ドメインに別の設定ファイルを使用*
>
> ```
> C:\EAP_HOME\bin> domain.bat --domain-config=domain-alternate.xml
> ```
>
> *この例では、**EAP_HOME\domain\configuration\domain-alternate.xml** 設定ファイルを使用します。*

バグの報告

2.2.9. JBoss EAP 6 の停止

JBoss EAP 6 を停止する方法は、起動方法によって異なります。このタスクでは、対話的に起動したインスタンスを停止し、サービスが開始したインスタンスを停止して、スクリプトによりバックグラウンドにフォークされたインスタンスを停止します。

注記

管理対象ドメインでサーバーまたはサーバーグループを停止する方法は、「管理コンソールを使用したサーバーの停止」 を参照してください。管理 CLI を使用してサーバーを停止する方法は、「管理 CLI を使用したサーバーの起動および停止」 を参照してください。

手順2.8 JBoss EAP 6 のインスタンスの停止

- *コマンドラインプロンプトから対話的に起動したインスタンスの停止*
  *JBoss EAP 6 が稼働しているターミナルで **Ctrl-C** を押します。*

手順2.9 オペレーティングシステムサービスとして起動されたインスタンスの停止

オペレーティングシステムに応じて、以下のいずれかの手順を実行します。

- - *Red Hat Enterprise Linux*
    *Red Hat Enterprise Linux の場合は、サービススクリプトを記述している場合は、**stop** 機能を使用します。これはスクリプトに記述する必要があります。次に、**service scriptname stop** を使用することができます。scriptname はスクリプト名に置き換えます。*

  - *Microsoft Windows Server*
    *Microsoft Windows では、**net service** コマンドを使用するか、コントロールパネルの **Services** アプレットからサービスを停止します。*

手順2.10 バックグラウンドで実行されているインスタンスの停止 (Red Hat Enterprise Linux)

1. *プロセスのプロセス ID(PID) を取得します。*

   - *単一のインスタンスのみが実行されている場合 (スタンドアロンモード)*
     以下のコマンドのいずれかが、JBoss EAP 6 の単一インスタンスの PID を返します。

     - ***pidof java***

     - ***jps***

       *（**jps** コマンドは、**jboss-modules.jar** 用と jps 自体の 2 つのプロセスの ID を返します。**jboss-modules.jar** の ID を使用して EAP インスタンスを停止します。*

   - *複数の EAP インスタンスが実行されている場合 (ドメインモード)*
     複数の EAP インスタンスが実行されている場合、正しいプロセスを識別するには、さらに包括的なコマンドを使用する必要があります。

     - ***jps** コマンドは冗長モードで使用して、見つかった java プロセスに関する詳細情報を提供できます。*

       *以下は、PID およびロールによって実行されている EAP プロセスを特定する、詳細な **jps** コマンドからのブリッジ出力です。*

       ```
       $ jps -v
       12155 jboss-modules.jar -D[Server:server-one] -XX:PermSize=256m -
       XX:MaxPermSize=256m -Xms1303m
       ...

       12196 jboss-modules.jar -D[Server:server-two] -XX:PermSize=256m -
       ```

```
XX:MaxPermSize=256m -Xms1303m
...

12096 jboss-modules.jar -D[Host Controller] -Xms64m -Xmx512m -
XX:MaxPermSize=256m
...

11872 Main -Xms128m -Xmx750m -XX:MaxPermSize=350m -
XX:ReservedCodeCacheSize=96m -XX:+UseCodeCacheFlushing
...

11248 jboss-modules.jar -D[Standalone] -XX:+UseCompressedOops -
verbose:gc
...

12892 Jps
...

12080 jboss-modules.jar -D[Process Controller] -Xms64m -Xmx512m -
XX:MaxPermSize=256m
...
```

- *ps aux コマンドを使用して、複数の EAP インスタンスに関する情報を返すこともできます。*

  以下は、PID およびロールによって実行されている EAP プロセスを特定する、冗長ps aux コマンドの出力です。

```
$ ps aux | grep java
username 12080  0.1  0.9 3606588 36772 pts/0   Sl+  10:09   0:01 /path/to/java -
D[Process Controller] -server -Xms128m -Xmx128m -XX:MaxPermSize=256m
...

username 12096  1.0  4.1 3741304 158452 pts/0  Sl+  10:09   0:13 /path/to/java -
D[Host Controller] -Xms128m -Xmx128m -XX:MaxPermSize=256m
...

username 12155  1.7  8.9 4741800 344224 pts/0  Sl+  10:09   0:22 /path/to/java -
D[Server:server-one] -XX:PermSize=256m -XX:MaxPermSize=256m -Xms1000m -
Xmx1000m -server -
...

username 12196  1.8  9.4 4739612 364436 pts/0  Sl+  10:09   0:22 /path/to/java -
D[Server:server-two] -XX:PermSize=256m -XX:MaxPermSize=256m -Xms1000m -
Xmx1000m -server
...
```

上記の例では、ドメイン全体を停止するにはProcess Controller プロセスを停止します。

grep ユーティリティーは、以下のいずれかのコマンドを使用してProcess Controller を特定できます。

```
jps -v | grep "Process Controller"
```

```
ps aux | grep "Process Controller"
```

-

2. *kill PID を実行してプロセス TERM シグナルを送信します。ここで、PID は上記のコマンドの 1 つで識別されるプロセス ID です。*

結果

JBoss EAP 6 がクリーンにシャットダウンされ、データは失われません。

バグの報告

2.2.10. Server Runtime で渡されるスイッチおよび引数の参照

*アプリケーションサーバーの起動スクリプトは実行時に引数とスイッチを受け入れます。これにより、**standalone.xml**、**domain.xml**、および **host.xml** 設定ファイルに定義される設定の代替設定でサーバーを起動できます。*

他の設定には、ソケットバインディングの代替セットを持つサーバーの起動や 2 次設定が含まれていることがあります。

*起動時に help スイッチ **-h** または **--help** を渡すと、利用可能なパラメーター一覧にアクセスできます。*

表2.2 ランタイムスイッチおよび引数

引数またはスイッチ	モード	説明
--admin-only	Standalone	サーバーの実行タイプを **ADMIN_ONLY** に設定します。これにより管理インターフェースが開かれ、管理リクエストが許可されますが、他のランタイムサービスは起動されず、エンドユーザーのリクエストは許可されません。
--admin-only	Domain	ホストコントローラーの実行タイプを **ADMIN_ONLY** に設定します。これにより管理インターフェースが開かれ、管理要求が許可されますが、サーバーは起動しません。または、このホストコントローラーがドメインのマスターである場合はスレーブホストコントローラーからの受信接続が許可されます。
-b=<value>, **-b <value>**	Standalone、Domain	パブリックインターフェースのバインドアドレスを設定するために使用される **jboss.bind.address** システムプロパティーを設定します。値の指定がない場合はデフォルトで 127.0.0.1 が指定されます。他のインターフェースにバインドアドレスを設定するには **-b<interface>=<value>** エントリーを確認します。
-b<interface>=<value>	Standalone、Domain	システムプロパティー **jboss.bind.address.<interface>** を指定の値に設定します。例：**-bmanagement=*IP_ADDRESS***
--backup	Domain	このホストがドメインコントローラーではない場合でも永続ドメイン設定のコピーを保持します。

引数またはスイッチ	モード	説明
-c=<config>, -c <config>	Standalone	使用するサーバー設定ファイルの名前。デフォルトは **standalone.xml** です。
-c=<config>, -c <config>	Domain	使用するサーバー設定ファイルの名前。デフォルトは **domain.xml** です。
--cached-dc	Domain	ホストがドメインコントローラーではなく、起動時にドメインコントローラーに接続できない場合、ローカルでキャッシュされたドメイン設定のコピーを使用してブートします。
--debug [<port>]	Standalone	オプションの引数を用いてデバッグモードを有効にし、ポートを指定します。起動スクリプトがサポートする場合のみ動作します。
-D<name>[=<value>]	Standalone m、Domain	システムプロパティーを設定します。
--domain-config=<config>	Domain	使用するサーバー設定ファイルの名前。デフォルトは **domain.xml** です。
-h, --help	Standalone m、Domain	ヘルプメッセージを表示し、終了します。
--host-config=<config>	Domain	使用するホスト設定ファイルの名前。デフォルトは **host.xml** です。
--interprocess-hc-address=<address>	Domain	ホストコントローラーがプロセスコントローラーからの通信をリッスンしなければならないアドレス。
--interprocess-hc-port=<port>	Domain	ホストコントローラーがプロセスコントローラーからの通信をリッスンしなければならないポート。
--master-address=<address>	Domain	システムプロパティー **jboss.domain.master.address** を指定の値に設定します。デフォルトのスレーブホストコントローラー設定では、マスターホストコントローラーのアドレスを設定するために使用されます。
--master-port=<port>	Domain	システムプロパティー **jboss.domain.master.port** を指定の値に設定します。デフォルトのスレーブホストコントローラー設定では、マスターホストコントローラーによるネイティブ管理の通信で使用されるポートを設定するために使用されます。
--read-only-server-config=<config>	Standalone	使用するサーバー設定ファイルの名前。元のファイルは上書きされないため、***--server-config*** および ***-c*** とは異なります。

引数またはスイッチ	モード	説明
--read-only-domain-config=<config>	Domain	使用するドメイン設定ファイルの名前。最初のファイルは上書きされないため、***--domain-config*** および ***-c*** とは異なります。
--read-only-host-config=<config>	Domain	使用するホスト設定ファイルの名前。最初のファイルは上書きされないため、***--host-config*** とは異なります。
-P=<url>, -P <url>, --properties=<url>	Standalone m、Domain	該当する URL からシステムプロパティーをロードします。
--pc-address=<address>	Domain	プロセスコントローラーが制御するプロセスからの通信をリッスンするアドレス。
--pc-port=<port>	Domain	プロセスコントローラーが制御するプロセスからの通信をリッスンするポート。
-S<name>[=<value>]	Standalone	セキュリティープロパティーを設定します。
--server-config=<config>	Standalone	使用するサーバー設定ファイルの名前。デフォルトは **standalone.xml** です。
-u=<value>, -u <value>	Standalone m、Domain	設定ファイルの socket-binding 要素のマルチキャストアドレスを設定するために使用される **jboss.default.multicast.address** システムプロパティーを設定します。値の指定がない場合はデフォルトで 230.0.0.4 が指定されます。
-v,-V,--version	Standalone m、Domain	アプリケーションサーバーのバージョンを表示し、終了します。

警告

*JBoss EAP 6 に同梱される設定ファイルは、スイッチ（**-b**、**-u**）を処理するよう設定されます。スイッチによって制御されるシステムプロパティーを使用しないよう設定ファイルを変更した場合は、実行するコマンドにスイッチを追加しても効果はありません。*

バグの報告

2.3. サーバーの起動と停止

2.3.1. 管理 CLI を使用したサーバーの起動および停止

前提条件

- *「管理 CLI の起動」*

サーバーは、管理 CLI または管理コンソールを使用して起動および停止できます。どちらのツールでも、単一のスタンドアロンサーバーインスタンスを制御し、管理対象ドメインのデプロイメント全体で複数のサーバーを管理できます。

管理コンソールを使用する場合は、「管理コンソールを使用したサーバーの起動」 を参照してください。管理 CLI を使用する場合、スタンドアロンサーバーおよび管理対象ドメインインスタンスでプロセスは異なります。

管理 CLI を用いたスタンドアロンサーバーの停止

*スクリプトまたはシェルプロンプトで手動で起動するスタンドアロンサーバーは、**shutdown** コマンドを使用して管理 CLI からシャットダウンできます。*

> *例2.5 管理 CLI よりスタンドアロンサーバーインスタンスを停止する*
>
> ```
> [standalone@localhost:9999 /] shutdown
> ```

JBoss EAP 6 スタンドアロンサーバーインスタンスを再起動するには、「JBoss EAP 6 をスタンドアロンサーバーとして起動」 の説明に従ってインスタンスの起動スクリプトを実行するか、手動で起動します。

管理 CLI を用いた管理対象ドメインの起動および停止

管理コンソールは、ドメイン内の特定サーバーを選択的に起動または停止できます。これには、ドメイン全体にわたるサーバーグループや、ホスト上の特定サーバーインスタンスが含まれます。

> *例2.6 管理 CLI より管理対象ドメインのサーバーホストを停止する*
>
> *スタンドアロンサーバーインスタンスと同様に、**shutdown** コマンドを使用して、宣言された管理対象ドメインホストをシャットダウンします。この例では、シャットダウン操作を呼び出す前にインスタンス名を宣言して master という名前のサーバーホストを停止します。*
>
> ```
> [domain@localhost:9999 /] shutdown --host=master
> ```

> *例2.7 管理 CLI より管理対象ドメインのサーバーホストを起動および停止する*
>
> *この例では、**start** および **stop** 操作を呼び出す前にグループを宣言することで、**main-server-group** という名前のデフォルトのサーバーグループを起動します。*
>
> ```
> [domain@localhost:9999 /] /server-group=main-server-group:start-servers
> ```
>
> ```
> [domain@localhost:9999 /] /server-group=main-server-group:stop-servers
> ```

> *例2.8 管理 CLI より管理対象ドメインのサーバーインスタンスを起動および停止する*
>
> *この例では、起動および停止操作を呼び出す前にホストとサーバー設定を宣言することで、マスター ホストの **server-one** という名前のサーバーインスタンスを起動 して停止 します。*

```
[domain@localhost:9999 /] /host=master/server-config=server-one:start

[domain@localhost:9999 /] /host=master/server-config=server-one:stop
```

注記

タブ キーを使用して文字列を補完し、利用可能なホストおよびサーバー設定の値などの可視変数を表示します。

バグの報告

2.3.2. 管理コンソールを使用したサーバーの起動

前提条件

- *「JBoss EAP 6 を管理対象ドメインとして起動」*
- *「管理コンソールへのログイン」*

手順2.11 管理対象ドメイン向けのサーバーの起動

1. *コンソール上部の ドメイン タブを選択し、**TOPOLOGY** タブを選択します。左側のナビゲーションバーの**Domain** で**Overview** を選択します。*
2. ***Server Instances** リストから、起動するサーバーを選択します。実行中のサーバーはチェックマークで表示されます。*

   このリストにあるインスタンスにカーソルを合わせ、サーバーの詳細の下に青色でオプションを表示します。
3. *インスタンスを起動するには、表示された時に**Start Server** テキストをクリックします。確認ダイアログボックスが開きます。確認 をクリックしてサーバーを起動します。*

結果

選択したサーバーが起動し、稼働状態になります。

バグの報告

2.3.3. 管理コンソールを使用したサーバーの停止

前提条件

- *「JBoss EAP 6 を管理対象ドメインとして起動」*
- *「管理コンソールへのログイン」*

手順2.12 管理コンソールを使用した管理対象ドメインのサーバーの停止

1. *コンソール上部の ドメイン タブを選択し、**TOPOLOGY** タブを選択します。左側のナビゲーションバーの**Domain** で**Overview** を選択します。*

2. *利用可能なサーバーインスタンスの一覧は、ホスト、グループ、およびサーバーインスタンスの表に表示されます。実行中のサーバーはチェックマークで表示されます。*

3. *選択したサーバーにカーソルを合わせます。表示される **Stop Server** テキストをクリックします。確認ダイアログウインドウが表示されます。*

4. *確認 をクリックしてサーバーを停止します。*

結果

選択されたサーバーが停止します。

バグの報告

2.4. 設定ファイル

2.4.1. JBoss EAP 6 の設定ファイル

***JBoss EAP 6** の設定は以前のバージョンから大幅に変更されました。最も明確な違いの1 つは、以下のファイルが1 つ以上含まれる簡素化された設定ファイル構造を使用することです。*

表2.3 設定ファイルの場所

サーバーモード	場所	目的
domain.xml	***EAP_HOME*/domain/configuration/domain.xml**	これは、管理対象ドメインの主要設定ファイルです。ドメインマスターのみがこのファイルを読み取ります。他のドメインメンバーでは削除できます。
host.xml	***EAP_HOME*/domain/configuration/host.xml**	このファイルには、管理対象ドメインの物理ホスト固有の設定情報が含まれています (ネットワークインターフェース、ソケットバインディング、ホスト名、その他のホスト固有の詳細など)。**host.xml** ファイルには、**host-master.xml** と **host-slave.xml** の両方の機能がすべて含まれています。これについては以下で説明します。このファイルはスタンドアロンサーバーには存在しません。
host-master.xml	***EAP_HOME*/domain/configuration/host-master.xml**	このファイルには、サーバーを管理対象ドメインのマスターサーバーとして実行するために必要な設定情報のみが含まれています。このファイルはスタンドアロンサーバーには存在しません。

サーバーモード	場所	目的
host-slave.xml	***EAP_HOME*/domain/configuration/host-slave.xml**	このファイルには、サーバーを管理対象ドメインのスレーブサーバーとして実行するために必要な設定情報のみが含まれています。このファイルはスタンドアロンサーバーには存在しません。
standalone.xml	***EAP_HOME*/standalone/configuration/standalone.xml**	スタンドアロンサーバーのデフォルト設定ファイルです。これには、サブシステム、ネットワーキング、デプロイメント、ソケットバインディング、およびその他の設定可能な詳細など、スタンドアロンサーバーに関するすべての情報が含まれます。この設定は、スタンドアロンサーバーの起動時に自動的に使用されます。
standalone-full.xml	***EAP_HOME*/standalone/configuration/standalone-full.xml**	スタンドアロンサーバーの設定例になります。これには、高可用性に必要なサブシステムを除くすべてのサブシステムのサポートが含まれます。これを使用するには、サーバーを停止し、次のコマンドを使用して再起動します。 ***EAP_HOME*/bin/standalone.sh -c standalone-full.xml**
standalone-ha.xml	***EAP_HOME*/standalone/configuration/standalone-ha.xml**	この設定ファイルの例では、デフォルトのサブシステムをすべて有効にし、高可用性または負荷分散クラスターに参加できるようにスタンドアロンサーバーの **mod_cluster** および JGroups サブシステムを追加します。このファイルは管理対象ドメインには適用されません。この設定を使用するには、以下のコマンドを実行してサーバーを停止し、再起動します。 ***EAP_HOME*/bin/standalone.sh -c standalone-ha.xml**

サーバーモード	場所	目的
standalone-full-ha.xml	***EAP_HOME*/standalone/configuration/standalone-full-ha.xml**	スタンドアロンサーバーの設定例になります。これには、高可用性に必要なサブシステムを含むすべてのサブシステムのサポートが含まれます。これを使用するには、サーバーを停止し、次のコマンドを使用して再起動します。 ***EAP_HOME*/bin/standalone.sh -c standalone-full-ha.xml**

これらはデフォルトの場所です。ランタイム時に別の設定ファイルを指定できます。

注記

JBoss EAP 6 の設定データのバックアップに関する詳細は、「JBoss EAP 設定データのバックアップ」 を参照してください。

バグの報告

2.4.2. JBoss EAP 設定データのバックアップ

概要

このトピックでは、後で JBoss EAP サーバー設定を復元するためにバックアップする必要のあるファイルについて説明します。

手順2.13 設定データのバックアップ

1. *ユーザーおよびプロファイルデータ、ドメイン、ホスト、スレーブ、およびロギング設定を保持するには、以下のディレクトリーの内容全体をバックアップします。*
   - *EAP_HOME/standalone/configuration/*
   - *EAP_HOME/domain/configuration*
2. ***EAP_HOME/modules/system/layers/base/** ディレクトリーに作成されたカスタムモジュールをバックアップします。*
3. ***EAP_HOME/welcome-content/** ディレクトリーの Welcome コンテンツをバックアップします。*
4. ***EAP_HOME/bin/** ディレクトリーに作成されたカスタムスクリプトをバックアップします。*

バグの報告

2.4.3. 記述子ベースのプロパティー置換

アプリケーションの設定（データソース接続パラメーターなど）は、通常は開発、テスト、および実稼働デプロイメントによって異なります。Java EE 仕様にはこれらの設定を外部化するメソッドが含まれていないため、このような違いはビルドシステムスクリプトで対応することがあります。JBoss EAP 6 では、記述子ベースのプロパティー置換 を使用して設定を外部で管理できます。

記述子ベースのプロパティー置換は、記述子を基にプロパティーを置き換えるため、アプリケーションやビルドチェーンから環境に関する仮定を除外できます。環境固有の設定は、アノテーションやビルドシステムスクリプトでなく、デプロイメント記述子に指定できます。設定はファイルに指定したり、パラメーターとしてコマンドラインで提供したりできます。

*記述子ベースのプロパティー置換は、**standalone.xml** または **domain.xml** を介してグローバルに有効になります。*

> *例2.9 記述子ベースのプロパティー置換*
>
> ```
> <subsystem xmlns="urn:jboss:domain:ee:1.2">
>   <spec-descriptor-property-replacement>
>     true
>   </spec-descriptor-property-replacement>
>   <jboss-descriptor-property-replacement>
>     true
>   </jboss-descriptor-property-replacement>
> </subsystem>
> ```

*Java EE 記述子置換はデフォルトで **無効** になっています。有効にすると、**ejb-jar.xml** および **persistence.xml** 設定ファイルで記述子を置き換えることができます。*

*JBoss 固有の記述子置換はデフォルトで **有効** になっています。有効にすると、以下の設定ファイルで記述子を置換できます。*

- ***jboss-ejb3.xml***
- ***jboss-app.xml***
- ***jboss-web.xml***
- ***-jms.xml***
- ***-ds.xml***

たとえば、以下のアノテーションを持つ Bean があるとします。

> *例2.10 アノテーションの例*
>
> ```
> @ActivationConfigProperty(propertyName = "connectionParameters", propertyValue =
> "host=192.168.1.1;port=5445")
> ```

*記述子ベースのプロパティー置換を有効にすると、以下のようにコマンドラインから **connectionParameters** を指定できます。*

```
./standalone.sh -DconnectionParameters='host=10.10.64.1;port=5445'
```

*システムプロパティーを使用して同じことを実現するには、literal の値の代わりに 式 を使用します。式は **${**パラメーター**:default}** の形式を取ります。式が設定で使用されると、そのパラメーターの値がその意味になります。パラメーターが存在しない場合は、代わりに指定されたデフォルト値が使用されます。*

例2.11 記述子で式の使用

```
<activation-config>
  <activation-config-property>
    <activation-config-property-name>
      connectionParameters
      </activation-config-property-name>
    <activation-config-property-value>
      ${jms.connection.parameters:'host=10.10.64.1;port=5445'}
    </activation-config-property-value>
  </activation-config-property>
</activation-config>
```

式 **${jms.connection.parameters:'host=10.10.64.1;port=5445'}** により、デフォルト値を提供しながらコマンドラインが提供するパラメーターで接続パラメーターを上書きできます。

バグの報告

2.4.4. 記述子ベースのプロパティー置換の有効化または無効化

概要

記述子プロパティー置換の **finite** コントロールが **jboss-as-ee_1_1.xsd** に導入されました。ここでは、記述子ベースのプロパティー置換を設定するのに必要な手順を説明します。

前提条件

- 「JBoss EAP 6 の起動」
- 「管理 CLI の起動」

記述子ベースのプロパティー置換フラグはブール値を持ちます。

- **true** に設定すると、プロパティー置換が有効になります。
- **false** に設定すると、プロパティー置換が無効になります。

手順2.14 **jboss-descriptor-property-replacement**

jboss-descriptor-property-replacement 以下の記述子でプロパティー置換を有効または無効にするために使用されます。

- ***jboss-ejb3.xml***
- ***jboss-app.xml***
- ***jboss-web.xml***
- ***-jms.xml***
- ***-ds.xml***

jboss-descriptor-property-replacement のデフォルト値は **true** です。

1. 管理 CLI で以下のコマンドを実行し、**jboss-descriptor-property-replacement** の値を確認します。

```
/subsystem=ee:read-attribute(name="jboss-descriptor-property-replacement")
```

2. 次のコマンドを実行して動作を設定します。

```
/subsystem=ee:write-attribute(name="jboss-descriptor-property-replacement",value=VALUE)
```

手順2.15 **spec-descriptor-property-replacement**

spec-descriptor-property-replacement 以下の記述子でプロパティー置換を有効または無効にするために使用されます。

- ***ejb-jar.xml***
- ***persistence.xml***
- ***application.xml***
- ***web.xml***

spec-descriptor-property-replacement のデフォルト値は **false** です。

1. 管理 CLI で以下のコマンドを実行し、**spec-descriptor-property-replacement** の値を確認します。

```
/subsystem=ee:read-attribute(name="spec-descriptor-property-replacement")
```

2. 次のコマンドを実行して動作を設定します。

```
/subsystem=ee:write-attribute(name="spec-descriptor-property-replacement",value=VALUE)
```

結果

記述子ベースのプロパティー置換が正常に設定されます。

バグの報告

2.4.5. ネストされた式

式はネスト化できるため、固定値の代わりにさらに高度な式を使用できます。ネストされた式の書式は、通常の式の場合と同様ですが、ある式が別の式に組み込まれます。 例を以下に示します。

例2.12 ネストされた式

```
${system_value_1${system_value_2}}
```

ネストされた式は、再帰的に評価されるため、最初に**内部**の式が評価され、次に**外部**の式が評価されます。ネストされた式は式が許可された場所であればどこでも許可されます（ただし、管理 CLI コマンドを除く）。

通常式と同様に、ネストされた式の解決でサポートされるソースはシステムプロパティー、環境変数、

*および Vault になります。デプロイメントのみの場合、デプロイメントアーカイブの **META-INF/jboss.properties** ファイルにリストされたプロパティーをソースとすることができます。サブデプロイメントをサポートする EAR またはその他のデプロイメントタイプでは、**META-INF/jboss.properties** が外部デプロイメント（EAR など）にある場合、解決はすべてのサブデプロイメントにスコープが設定され、**META-INF/jboss.properties** が EAR 内のサブデプロイメントアーカイブ（例： EAR 内の WAR）にある場合はサブデプロイメントに対してスコープ指定されます。*

> *例2.13 設定ファイルでのネストされた式の使用*
>
> *ネストされた式の実際のライフサイクルアプリケーションはデータソース定義にあります。データソース定義で使用されるパスワードがマスクされている場合は、データソース定義で生成される行は以下のようになります。*
>
> ```
> <password>${VAULT::ds_ExampleDS::password::1}</password>
> ```
>
> *ネストされた式を使用すると、**ds_ExampleDS** の値をシステムプロパティーに置き換えることができます。システムプロパティー **datasource_name** に **ds_ExampleDS** の値が割り当てられている場合、データソース定義の行は以下のようになります。*
>
> ```
> <password>${VAULT::${datasource_name}::password::1}</password>
> ```
>
> *JBoss EAP は最初に式 **${datasource_name}** を評価し、次にこれを大きな式に入力し、結果となる式を評価します。この設定の利点は、データソースの名前が固定された設定から抽象化されることです。*

式が再帰的になれば、式が式に解決され、その後解決されます。ネストされた式と再帰式は間接的な形です。管理 CLI コマンドでは再帰式が許可されないことに注意してください。

> *例2.14 再帰式*
>
> *前の例を続行すると、式 **${ VAULT::ds_ExampleDS::password::1}** に解決される **${ foo}** 式が使用され、Vault: シークレット に含まれる値に解決されます。*

バグの報告

2.4.6. ファイルの履歴設定

*アプリケーションサーバーの設定ファイルには、**standalone.xml**、**domain.xml** ファイル、および **host.xml** ファイルが含まれます。これらのファイルは直接編集して変更できますが、管理 CLI や管理コンソールなど、利用可能な管理操作でアプリケーションサーバーモデルを設定する方法が推奨されます。*

サーバーインスタンスの保守および管理を容易にするために、アプリケーションサーバーは起動時に元の設定ファイルのタイムスタンプバージョンを作成します。管理操作によってその他の設定変更が行われると、元のファイルが自動的にバックアップされ、インスタンスの作業用コピーが参照およびロールバック用に保持されます。このアーカイブ機能では、サーバー設定のスナップショットの保存、読み込み、および削除が拡張され、再利用およびロールバックシナリオが可能になります。

- *「以前の設定でのサーバーの起動」*
- *「管理 CLI を使用した設定スナップショットの保存」*
- *「管理 CLI を使用した設定スナップショットのロード」*

- 「管理 CLI を使用した設定スナップショットの削除」
- 「管理 CLI を使用したすべての設定スナップショットのリスト」

バグの報告

2.4.7. 以前の設定でのサーバーの起動

以下の例は、**standalone.xml** を使用してスタンドアロンサーバーの以前の設定でアプリケーションサーバーを起動する方法を示しています。同じ概念が **domain.xml** と **host.xml** の管理対象ドメインにそれぞれ適用されます。

この例では、管理操作によってサーバーモデルが変更される場合にアプリケーションサーバーにより自動的に保存される以前の設定が呼び出されます。

例2.15 保存した設定でサーバーを起動します。

1. 起動するバックアップバージョンを特定します。この例では、起動に成功すると最初の変更の前にサーバーモデルのインスタンスが呼び出されます。

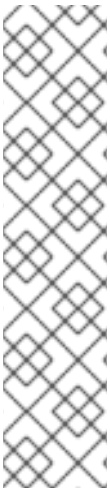
```
EAP_HOME/standalone/configuration/standalone_xml_history/current/standalone.v1.xml
```

2. **jboss.server.config.dir** 下で相対ファイル名を渡し、バックアップモデルのこの設定を用いてサーバーを起動します。

```
EAP_HOME/bin/standalone.sh --server-config=standalone_xml_history/current/standalone.v1.xml
```

結果

アプリケーションサーバーが、選択された設定で起動されます。



注記

ドメイン設定履歴は **EAP_HOME/domain/configuration/domain_xml_history/current/domain.v1.xml** にあります。

jboss.domain.config.dir 下で相対ファイル名を渡し、バックアップモデルのこの設定を用いてサーバーを起動します。

この設定を用いてドメインを起動するには、以下を実行します。

```
EAP_HOME/bin/domain.sh --domain-config=domain_xml_history/current/domain.v1.xml
```

バグの報告

2.4.8. 管理 CLI を使用した設定スナップショットの保存

概要

設定スナップショットは、現在のサーバー設定の特定の時点のコピーです。これらのコピーは管理者が保存およびロードできます。

*以下の例では、**standalone.xml** 設定ファイルを使用しますが、同じプロセスが**domain.xml** および **host.xml** 設定ファイルにも適用されます。*

前提条件

- *「管理 CLI の起動」*

手順2.16 設定スナップショットの撮影および保存

- *スナップショットの保存*
  ***take-snapshot** 操作を実行し、現在のサーバー設定のコピーを取得します。*

```
[standalone@localhost:9999 /] :take-snapshot
{
    "outcome" => "success",
    "result" =>
"/home/User/EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/
20110630-172258657standalone.xml"
```

結果

現在のサーバー設定のスナップショットが保存されます。

バグの報告

2.4.9. 管理 CLI を使用した設定スナップショットのロード

設定スナップショットは、現在のサーバー設定の特定の時点のコピーです。これらのコピーは管理者が保存およびロードできます。スナップショットを読み込むプロセスは、スナップショットの作成、一覧表示、および削除に使用される管理 CLI インターフェースではなく、コマンドラインから使用する方法と似ています。「以前の設定でのサーバーの起動」

*以下の例は **standalone.xml** ファイルを使用しますが、同じプロセスが**domain.xml** および**host.xml** ファイルに適用されます。*

手順2.17 設定スナップショットのロード

1. *ロードするスナップショットを特定します。この例では、スナップショットディレクトリーから以下のファイルを呼び出します。スナップショットファイルのデフォルトのパスは以下のとおりです。*

   ```
   EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110812-
   191301472standalone.xml
   ```

   スナップショットは相対パスにより表されます。たとえば、上記の例は次のように記述できます。

   ```
   jboss.server.config.dir/standalone_xml_history/snapshot/20110812-
   191301472standalone.xml
   ```

2. *ファイル名を渡し、選択したスナップショットを用いてサーバーを起動します。*

```
EAP_HOME/bin/standalone.sh --server-
config=standalone_xml_history/snapshot/20110913-164449522standalone.xml
```

結果

ロードしたスナップショットに選択された設定を用いてサーバーが再起動します。

バグの報告

2.4.10. 管理 CLI を使用した設定スナップショットの削除

前提条件

- 「管理 CLI の起動」

設定スナップショットは、現在のサーバー設定の特定の時点のコピーです。これらのコピーは管理者が保存およびロードできます。

以下の例は **standalone.xml** ファイルを使用しますが、同じプロセスが **domain.xml** および **host.xml** ファイルに適用されます。

手順2.18 特定のスナップショットの削除

1. 削除するスナップショットを特定します。この例では、スナップショットディレクトリーから以下のファイルを削除します。

```
EAP_HOME/standalone/configuration/standalone_xml_history/snapshot/20110630-
165714239standalone.xml
```

2. 以下の例のようにスナップショットの名前を指定して、**:delete-snapshot** コマンドを実行して特定のスナップショットを削除します。

```
[standalone@localhost:9999 /] :delete-snapshot(name="20110630-
165714239standalone.xml")
{"outcome" => "success"}
```

結果

スナップショットが削除されます。

手順2.19 スナップショットすべてを削除

- 以下の例のように、**:delete-snapshot(name="all")** コマンドを実行して、すべてのスナップショットを削除します。

```
[standalone@localhost:9999 /] :delete-snapshot(name="all")
{"outcome" => "success"}
```

結果

スナップショットがすべて削除されます。

バグの報告

2.4.11. 管理 CLI を使用したすべての設定スナップショットのリスト

前提条件

- *「管理 CLI の起動」*

設定スナップショットは、現在のサーバー設定の特定の時点のコピーです。これらのコピーは管理者が保存およびロードできます。

*以下の例は **standalone.xml** ファイルを使用しますが、同じプロセスが **domain.xml** および **host.xml** ファイルに適用されます。*

手順2.20 すべての設定スナップショットのリスト

- *すべてのスナップショットのリスト*
  ***:list-snapshots** コマンドを実行して、保存されたすべてのスナップショットを一覧表示します。*

```
[standalone@localhost:9999 /] :list-snapshots
{
    "outcome" => "success",
    "result" => {
        "directory" =>
"/home/hostname/EAP_HOME/standalone/configuration/standalone_xml_history/snapshot",
        "names" => [
            "20110818-133719699standalone.xml",
            "20110809-141225039standalone.xml",
            "20110802-152010683standalone.xml",
            "20110808-161118457standalone.xml",
            "20110912-151949212standalone.xml",
            "20110804-162951670standalone.xml"
        ]
    }
}
```

結果

スナップショットがリストされます。

バグの報告

2.5. ファイルシステムパス

***JBoss EAP 6** は、ファイルシステムパスに論理名を使用します。**domain.xml**、**host.xml**、および **standalone.xml** 設定ファイルには、パスを宣言するセクションが含まれます。*

各ファイルの他のセクションは論理名を使用してパスを参照できます。そのため、各インスタンスに絶対パスを使用する必要がなく、特定のホスト設定が汎用論理名に解決できます。

*たとえば、デフォルトの logging サブシステム設定は **jboss.server.log.dir** をサーバーのログディレクトリーの論理名として宣言します。*

例2.16 ロギングディレクトリーの相対パス例

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

JBoss EAP 6 では、複数の標準的なパスが自動的に提供されるため、ユーザーが設定ファイルでこれらのパスを設定する必要はありません。

表2.4 標準的なパス

Value	説明
java.ext.dirs	Java Development Kit 拡張ディレクトリーパス。
jboss.home.dir	JBoss EAP 6 ディストリビューションのルートディレクトリー。
user.home	ユーザーのホームディレクトリー。
user.dir	ユーザーのカレントワーキングディレクトリー。
java.home	Java インストールディレクトリー。
jboss.server.base.dir	各サーバーインスタンスのルートディレクトリー。
jboss.server.data.dir	サーバーが永続データファイルストレージに使用するディレクトリー。
jboss.server.config.dir	サーバー設定が含まれるディレクトリー。
jboss.server.log.dir	サーバーがファイルストレージに使用するディレクトリー。
jboss.server.temp.dir	サーバーが一時ファイルストレージに使用するディレクトリー。
jboss.server.deploy.dir	サーバーがデプロイされたコンテンツの格納に使用するディレクトリー。
jboss.controller.temp.dir	ホストコントローラーが一時的なファイルストレージとして使用するディレクトリー。
jboss.domain.base.dir	ドメインコンテンツのベースディレクトリー。
jboss.domain.config.dir	ドメイン設定が含まれるディレクトリー。
jboss.domain.data.dir	ドメインが永続データファイルの格納に使用するディレクトリー。
jboss.domain.log.dir	ドメインが永続ログファイルの格納に使用するディレクトリー。

Value	説明
jboss.domain.temp.dir	ドメインが一時ファイルの格納に使用するディレクトリー。
jboss.domain.deployment.dir	ドメインがデプロイ済みコンテンツの格納に使用するディレクトリー。
jboss.domain.servers.dir	ドメインが管理対象ドメインインスタンスの出力を格納するために使用するディレクトリー。

パスのオーバーライド

*スタンドアロンサーバーを実行している場合は、2 つの方法のいずれかですべての **jboss.server.*** パスを上書きできます。*

- *サーバーの起動時にコマンドライン引数を渡すことができます。以下に例を示します。*

  ```
  bin/standalone.sh -Djboss.server.log.dir=/var/log
  ```

- *サーバー設定ファイルで **JAVA_OPTS** 変数を変更できます。**EAP_HOME/bin/standalone.conf** ファイルを開き、 ファイルの最後に以下の行を追加します。*

  ```
  JAVA_OPTS="$JAVA_OPTS -Djboss.server.log.dir=/var/log"
  ```

*パスのオーバーライドは、管理対象ドメインで実行しているサーバーでサポートされます。たとえば、**jboss.domain.servers.dir** を使用して管理対象ドメインのサーバーのベースディレクトリーを変更できます。*

カスタムパスの追加

*また、独自のカスタムパスを作成することもできます。たとえば、ロギングに使用する相対パスを定義できます。次に、ログハンドラーを変更して **my.relative.path** を使用することができます。*

例2.17 カスタムロギングパス

```
my.relative.path=/var/log
```

バグの報告

2.5.1. ディレクトリーのグループ化

*ドメインモードでは、各サーバーのファイルは **EAP_HOME/domain/** ディレクトリーに保存されます。サブディレクトリーの名前は、server または file タイプのいずれかで **directory-grouping** 属性に従って付けられます。*

サーバーを基にしたディレクトリーのグループ化

*デフォルトのディレクトリーグルーピングはサーバー です。管理作業が**サーバー中心** である場合はこの設定が推奨されます。たとえば、サーバーインスタンスごとにバックアップやログファイルの処理を設定することができます。*

例2.18 サーバーを基にしたディレクトリーのグループ化

Zip メソッドを使用して JBoss EAP がインストールされ、すべてのデフォルトオプションが適用されると、ドメインモードのディレクトリー構造は以下のようになります。

```
EAP_HOME/domain
        └─ servers
            ├── server-one
            │   ├── data
            │   ├── tmp
            │   └── log
            └── server-two
                ├── data
                ├── tmp
                └── log
```

directory-grouping 属性をデフォルトから変更し、リセットする場合は、以下の管理 CLI コマンドを入力します。

```
/host=master:write-attribute(name="directory-grouping",value="by-server")
```

これにより、コントローラーの **host.xml** 設定ファイルが更新されます。

```
<servers directory-grouping="by-server">
  <server name="server-one" group="main-server-group" >
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
  </server>
</servers>
```

タイプを基にしたディレクトリーのグループ化

各サーバーのディレクトリーをサーバーごとにグループ化するのではなく、ファイルタイプでグループ化することもできます。管理作業が**ファイルタイプ中心**である場合は、この設定が推奨されます。たとえば、データファイルのみを含める場合は、バックアップ設定が簡単になります。

タイプ 別にドメインデータディレクトリーをグループ化するには、以下の管理 CLI コマンドを入力します。

```
/host=master:write-attribute(name="directory-grouping",value="by-type")
```

これにより、コントローラーの **host.xml** 設定ファイルが更新されます。

```
<servers directory-grouping="by-type">
  <server name="server-one" group="main-server-group" >
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
  </server>
</servers>
```

例2.19 タイプを基にしたディレクトリーのグループ化

Zip メソッドを使用して JBoss EAP がインストールされ、ドメインのファイルがタイプを基にグループ化された場合、ドメインモードのディレクトリー構造は以下のようになります。

```
EAP_HOME/domain
        ├── data
        │   └── servers
        │       ├── server-one
        │       └── server-two
        ├── log
        │   └── servers
        │       ├── server-one
        │       └── server-two
        └── tmp
            └── servers
                ├── server-one
                └── server-two
```

バグの報告

2.5.2. ユースケース： ディレクトリーの上書き

この例の目的は、/opt/jboss_eap/data/domain_data ディレクトリーにドメインファイルを格納し、各最上位ディレクトリーにカスタム名を付けることです。使用されるディレクトリーのグループ化は、デフォルトの by-server です。

- *サブディレクトリー all_logs に保存されているログファイル*
- *サブディレクトリー all_data に保存されているデータファイル*
- *all_temp サブディレクトリーに格納されている一時ファイル*
- *all_servers サブディレクトリーに格納されているサーバーのファイル*

この設定を行うには、JBoss EAP の起動時に複数のシステムプロパティーを上書きします。

```
./domain.sh \
    -Djboss.domain.temp.dir=/opt/jboss_eap/data/domain_data/all_temp \
    -Djboss.domain.log.dir=/opt/jboss_eap/data/domain_data/all_logs \
    -Djboss.domain.data.dir=/opt/jboss_eap/data/domain_data/all_data\
    -Djboss.domain.servers.dir=/opt/jboss_eap/data/domain_data/all_servers
```

この結果、パス構造は次のようになります。

```
/opt/jboss_eap/data/domain_data/
├── all_data
│   └── content
├── all_logs
│   ├── host-controller.log
│   └── process-controller.log
├── all_servers
│   ├── server-one
│   │   ├── data
│   │   │   ├── content
│   │   │   ├── logging.properties
```

```
    │   │   ├── log
    │   │   │   └── server.log
    │   │   └── tmp
    │   │       ├── vfs
    │   │       │   └── temp
    │   │       └── work
    │   │           └── jboss.web
    │   │               └── default-host
    │   └── server-two
    │       ├── data
    │       │   ├── content
    │       │   ├── logging.properties
    │       ├── log
    │       │   └── server.log
    │       └── tmp
    │           ├── vfs
    │           │   └── temp
    │           └── work
    │               └── jboss.web
    │                   └── default-host
    └── all_temp
        └── auth
            ...
```

バグの報告

第3章 管理インターフェース

3.1. アプリケーションサーバーの管理

JBoss EAP 6 は、サーバー設定に XML ファイルを使用し、JBoss EAP 6 サーバーの設定および管理の 3 つの方法（Web インターフェース、コマンドラインクライアント、および XML 設定ファイルの直接編集）を提供します。

設定ファイルを編集するのに推奨される方法は、管理 CLI と新しい Web ベースの管理コンソールです。XML 設定ファイルに加えた編集は引き続き可能ですが、管理コンソールと管理 CLI を使用して設定および管理することで、サーバーインスタンスの管理に追加の検証および高度な機能が提供されます。

3 つの方法のいずれかでサーバー設定に加えた編集は、異なるビュー全体で同期されます。

注記

ただし、サーバーインスタンスの実行中に XML 設定ファイルに加えた編集は、サーバーモデルによって上書きされます。サーバーインスタンスの実行中に XML 設定ファイルに追加されたコメントもすべて削除されます。

Web ブラウザーでグラフィカルユーザーインターフェースを使用してサーバーを管理するには、管理コンソールを使用します。コマンドラインインターフェースを使用してサーバーを管理するには、管理 CLI を使用します。

管理コンソールと管理 CLI の推奨管理ツールとして、上級ユーザーが希望する場合に独自のツールを開発できるようにする基礎となる管理 API の例も提供されています。

バグの報告

3.2. 管理アプリケーションプログラミングインターフェース (API)

HTTP API

管理コンソールは、Google Web Toolkit(GWT) で構築された Web インターフェースです。HTTP 管理インターフェースを使用してサーバーと通信します。

HTTP API エンドポイントは、管理レイヤーと統合するために HTTP プロトコルに依存する管理クライアントのエントリーポイントです。

HTTP プロトコルに依存する管理クライアントは、JSON でエンコードされたプロトコルとデタイプを使用しない RPC スタイルの API を使用して、管理対象ドメインまたはスタンドアロンサーバーに対して管理操作を記述および実行します。

HTTP API は管理コンソールによって使用されますが、他のクライアントの統合機能も提供します。

HTTP API エンドポイントはドメインコントローラーまたはスタンドアロンサーバーインスタンスと同じ場所に置かれます。管理操作を実行するコンテキストと、Web インターフェースにアクセスするためのコンテキストが 2 つあります。デフォルトでは、HTTP API エンドポイントはポート 9990 で実行されます。

例3.1 HTTP API 設定ファイルの例

```
<management-interfaces>
```

```
    [...]
    <http-interface security-realm="ManagementRealm">
      <socket-binding http="management-http"/>
    </http-interface>
  </management-interfaces>
```

管理コンソールは、HTTP 管理 API と同じポートで提供されます。デフォルトの localhost でアクセスされる管理コンソール、特定のホストとポートの組み合わせによってリモートでアクセスされる管理コンソール、および公開されるドメイン API を区別することが重要です。

表3.1 管理コンソールまたは公開される HTTP API へのアクセスに使用する URL

URL	説明
http://localhost:9990/console	ローカルホストでアクセスされる管理コンソール (管理対象ドメイン設定を制御します)。
http://*hostname*:9990/console	リモートでアクセスされる管理コンソール (ホストを指定し、管理対象ドメイン設定を制御します)。
http://*hostname*:9990/management	HTTP 管理 API は管理コンソールと同じポートで実行され、API に公開された raw 属性と値を表示します。

例3.2 HTTP API を使用した属性値の取得

*以下の URL は HTTP Web コネクター属性値を取得します（デフォルトの操作は **read-resource** です）。*

```
http://hostname:9990/management/subsystem/web/connector/http
```

例3.3 HTTP API を使用した単一の属性値の取得

*以下の URL は、ExampleDS データソースの **enabled** 属性を取得します。*

```
http://hostname:9990/management/subsystem/datasources/data-source/ExampleDS?
operation=attribute&name=enabled
```

HTTP API を使用してアプリケーションをデプロイする方法は、「HTTP API を使用したアプリケーションのデプロイ」 を参照してください。

ネイティブ API

管理 CLI はネイティブ API ツールです。これは管理対象ドメインまたはスタンドアロンサーバーインスタンスで利用でき、管理者はドメインコントローラーまたはスタンドアロンサーバーインスタンスに接続し、データイプな管理モデルを介して利用可能な管理操作を実行できます。

ネイティブ API エンドポイントは、ネイティブプロトコルに依存して管理レイヤーと統合する管理クライアントのエントリーポイントです。管理操作の記述および実行には、非常に少数の Java タイプに基

づいた、オープンバイナリープロトコルと RPC スタイルの API を使用します。これは、管理 CLI 管理ツールによって使用されますが、他のクライアントの統合機能も提供します。

ネイティブ API エンドポイントはホストコントローラーまたはスタンドアロンサーバーと共存します。管理 CLI を使用するには、これを有効にする必要があります。デフォルトでポート 9999 で実行されます。

例3.4 ネイティブ API 設定ファイルの例

```
<management-interfaces>
  <native-interface security-realm="ManagementRealm">
    <socket-binding native="management-native"/>
  </native-interface>
  [...]
</management-interfaces>
```

バグの報告

3.3. 管理コンソール

3.3.1. 管理コンソール

管理コンソールは JBoss EAP 6 の Web ベースの管理ツールです。

管理コンソールを使用して、サーバーの起動および停止、アプリケーションのデプロイおよびアンデプロイ、システム設定の調整、サーバー設定への永続的な変更を行います。管理コンソールは管理タスクも実行でき、変更があればサーバーインスタンスの再起動またはリロードが必要な場合はライブ通知も行います。

管理対象ドメインでは、同じドメイン内のサーバーインスタンスやサーバーグループをドメインコントローラーの管理コンソールから集中管理できます。

バグの報告

3.3.2. 管理コンソールへのログイン

前提条件

- JBoss EAP 6 が稼働している必要があります。
- コンソールにアクセスするためのパーミッションを持つユーザーがすでに作成されている必要があります。

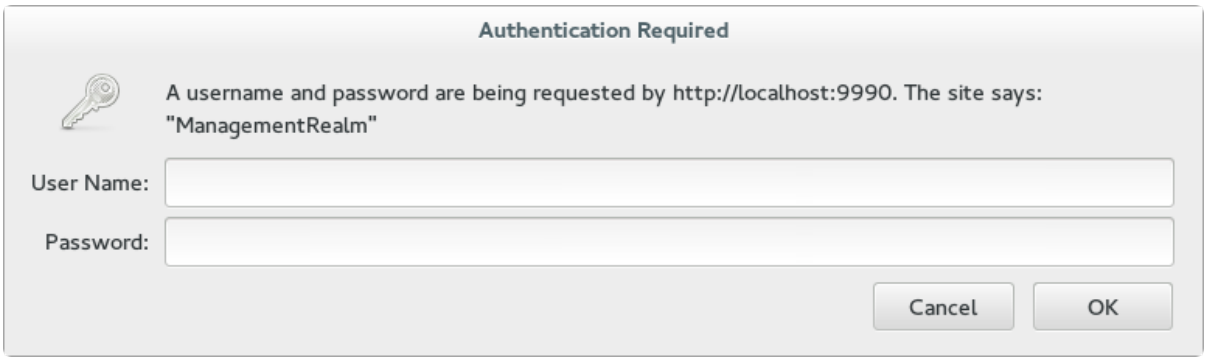
1. Web ブラウザーを起動し、以下のアドレスに移動します。
   http://localhost:9990/console/App.html

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. 管理コンソールにログインするためのユーザー名とパスワードを入力します。

図3.1 管理コンソールのログイン画面



結果

ログインすると、以下のアドレスにリダイレクトされ、管理コンソールのランディングページが表示されます。 http://localhost:9990/console/App.html#home

バグの報告

3.3.3. 管理コンソールの言語の変更

Web ベースの管理コンソールの言語設定では、デフォルトで英語が使用されます。以下の言語のいずれかを選択できます。

サポートされている言語

- *ドイツ語 (de)*
- *簡体中国語 (zh-Hans)*
- *ブラジルポルトガル語 (pt-BR)*
- *フランス語 (fr)*
- *スペイン語 (es)*
- *日本語 (ja)*

手順3.1 Web ベース管理コンソールの言語の変更

1. *管理コンソールへのログイン*
   Web ベースの管理コンソールにログインします。

2. *Settings ダイアログを開きます。*
   *画面の右下付近には **Settings** ラベルがあります。クリックして管理コンソールの設定を開きます。*

3. *必要な言語を選択します。*
   ***Locale** 選択ボックスから希望の言語を選択します。**Save** を選択します。確認ボックスに、アプリケーションのリロードが必要であると表示されます。**Confirm** をクリックします。システムは、Web ブラウザーを自動的に更新し、新しいロケールを使用します。*

バグの報告

3.3.4. JBoss EAP コンソールの分析

Google Analytics とは

Google Analytics は、Web サイトに包括的な使用統計を提供する、無料の Web 分析サービスです。サイトへのアクセス、ページビュー、アクセスごとのページ、およびサイトに費やした平均時間など、サイトに関する重要なデータを提供します。Google Analytics を使用すると、Web サイトの存在と、その機能をより明確に把握することができます。

JBoss EAP 管理コンソールの Google Analytics

JBoss EAP 6 では、管理コンソールで Google Analytics を有効または無効にするオプションを利用できます。Google Analytics 機能は、お客様がコンソールをどのように使用するか、コンソールのどの部分がお客様の最も重要かについて、Red Hat EAP チームが理解できるようにしています。この情報は、チームがコンソールの設計、機能、およびコンテンツを顧客の即時のニーズに適応させるのに役立つものです。

備考

デフォルトでは、JBoss EAP 6 コンソールで Google Analytics が無効になり、その使用は任意です。

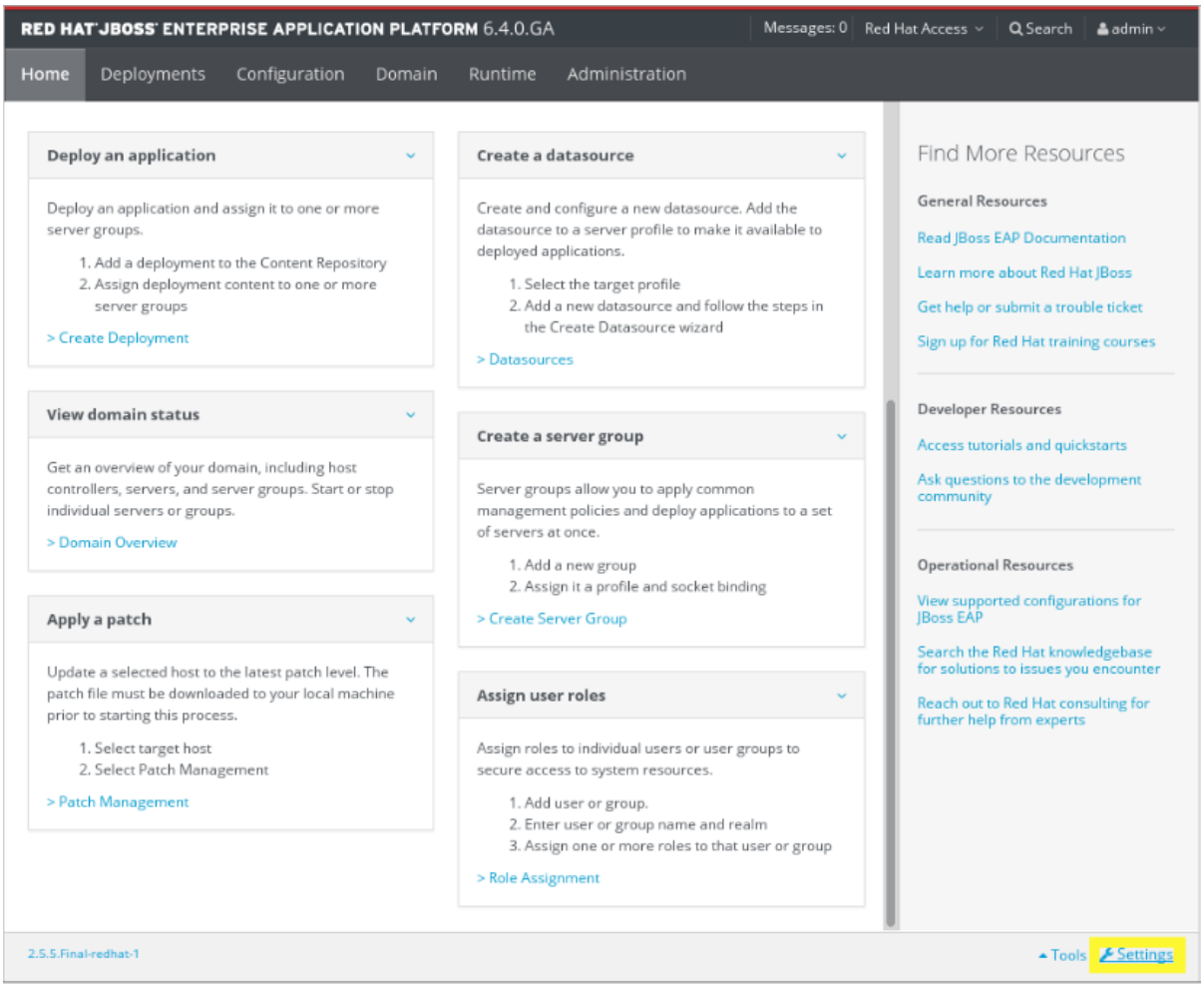
バグの報告

3.3.5. JBoss EAP コンソールでの Google Analytics の有効化

JBoss EAP 管理コンソールで Google Analytics を有効にするには、以下を行います。

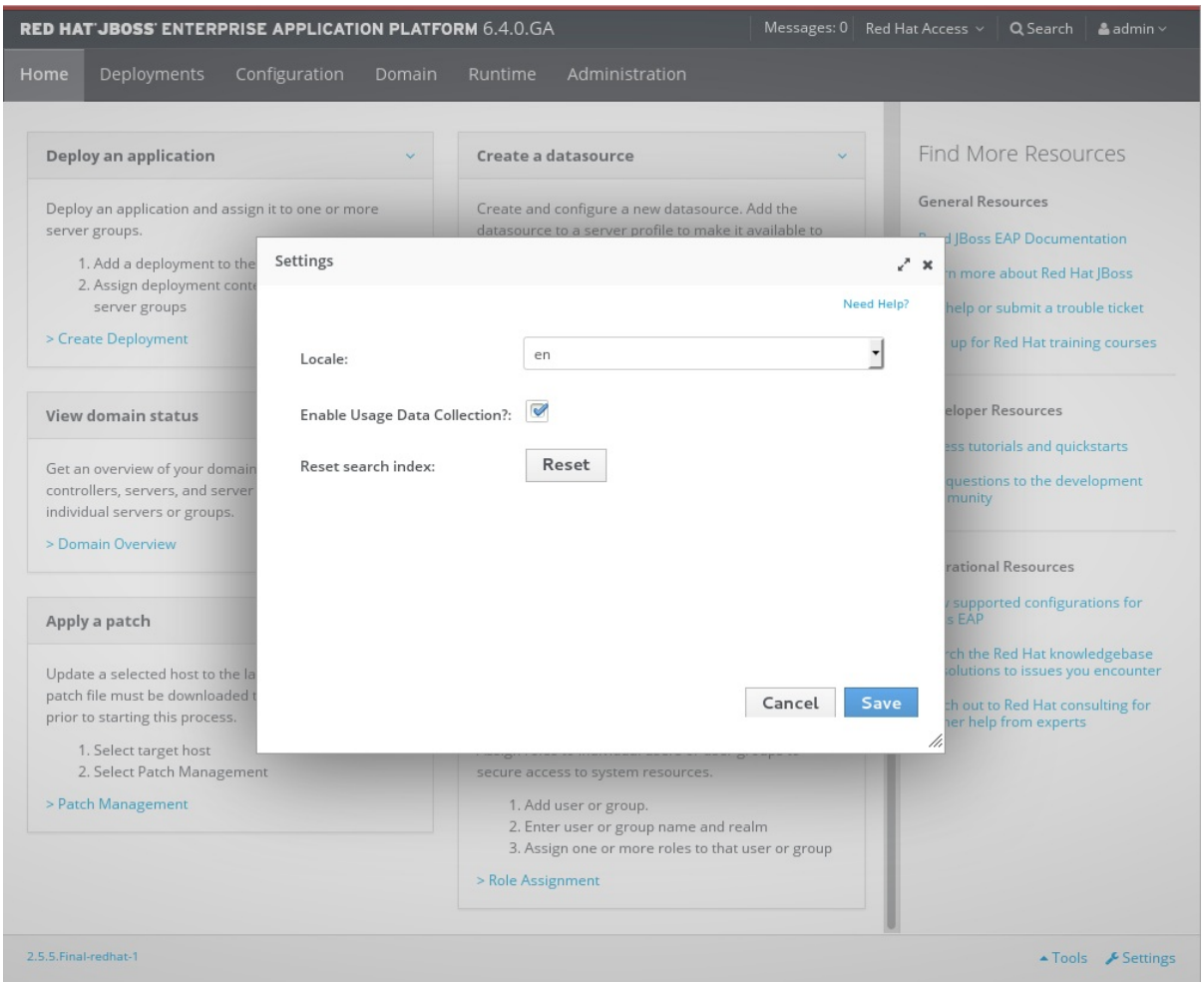
- 管理コンソールへのログイン
- 管理 コンソールの設定 をクリック

図3.2 管理コンソールのログイン画面



- ***Settings*** *ダイアログで* ***Enable Usage Data Collection*** *チェックボックスを選択し、* ***Save*** *ボタンをクリックします。アプリケーションのリロードを確認し、新しい設定を有効にします。*

図3.3 Settings ダイアログ（使用率データの収集を有効化）



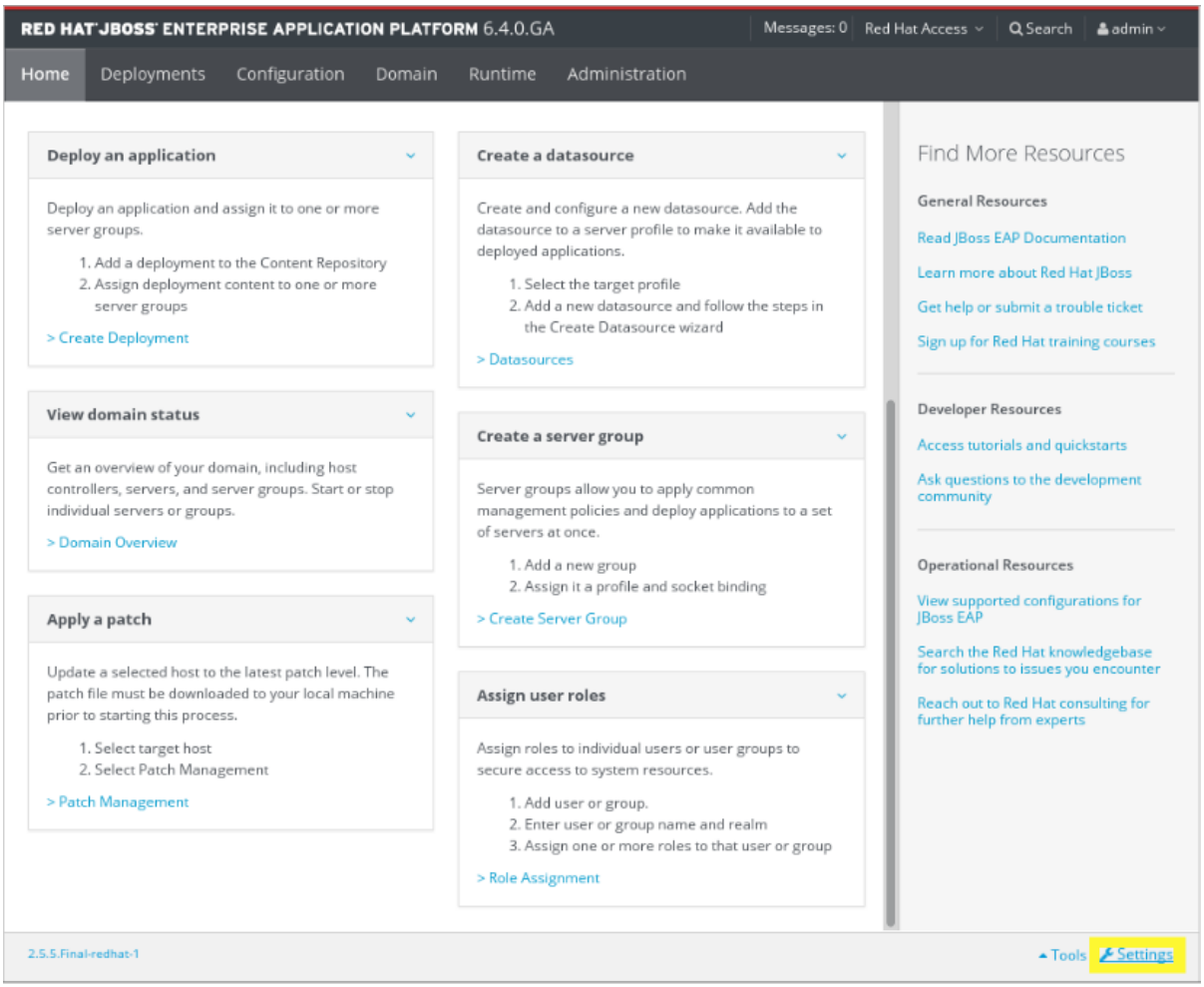
バグの報告

3.3.6. JBoss EAP コンソールでの Google Analytics の無効化

JBoss EAP 管理コンソールで Google Analytics を無効にするには、以下を行います。

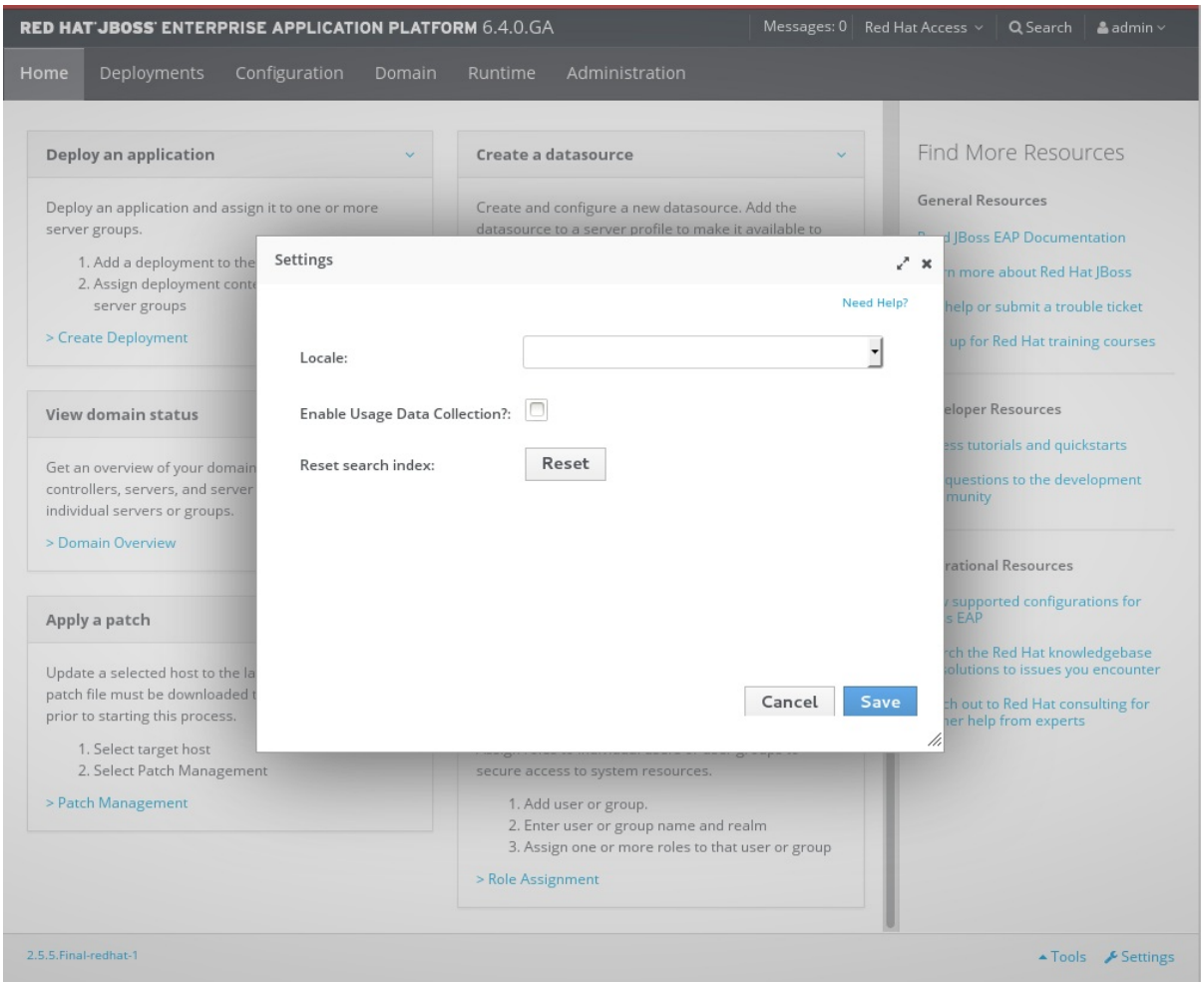
- *管理コンソールへのログイン*
- *管理 コンソールの設定 をクリック*

図3.4 管理コンソールのログイン画面



- ***Settings*** *ダイアログの* ***Enable Usage Data Collection*** *オプションの選択を解除して、選択を削除します。* ***Save*** *ボタンをクリックします。アプリケーションのリロードを確認し、新しい設定を有効にします。*

図3.5 Settings ダイアログ（使用率データの収集を無効化）



バグの報告

3.3.7. 管理コンソールを使用したサーバーの設定

前提条件

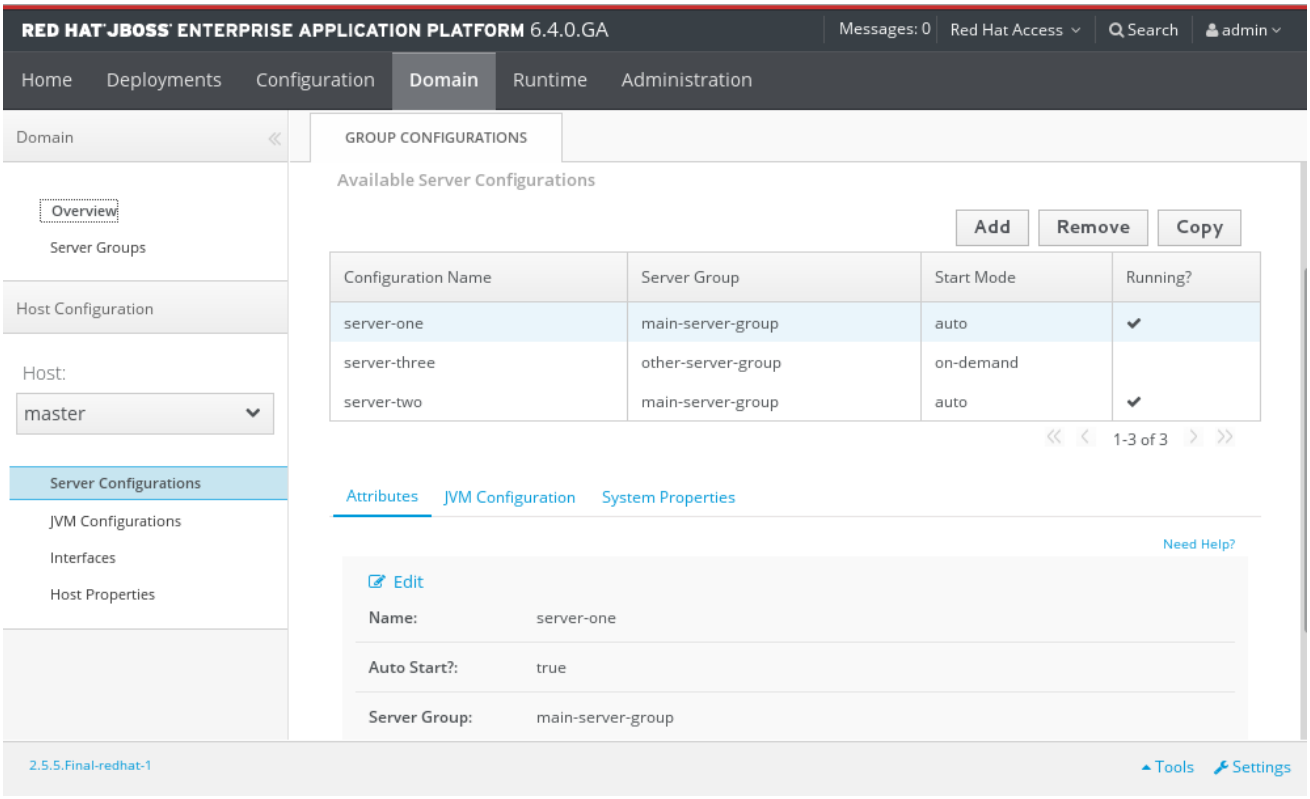
- *「JBoss EAP 6 を管理対象ドメインとして起動」*
- *「管理コンソールへのログイン」*

手順3.2 サーバーの設定

1. *コンソールの上部から ドメイン タブを選択します。利用可能なサーバーインスタンスが表に表示されます。*
2. ***Server Configurations** をクリックします。*

   *関連するホストの **Server Configurations** パネルが表示されます。*
3. ***Available Server Configurations** テーブルからサーバーインスタンスを選択します。*
4. *選択したサーバーの詳細の上に **Edit** をクリックします。*
5. *設定属性を変更します。*
6. ***Save** をクリックして終了します。*

図3.6 サーバー構成



結果

サーバー設定が変更され、サーバーが次回再起動したときに変更が反映されます。

バグの報告

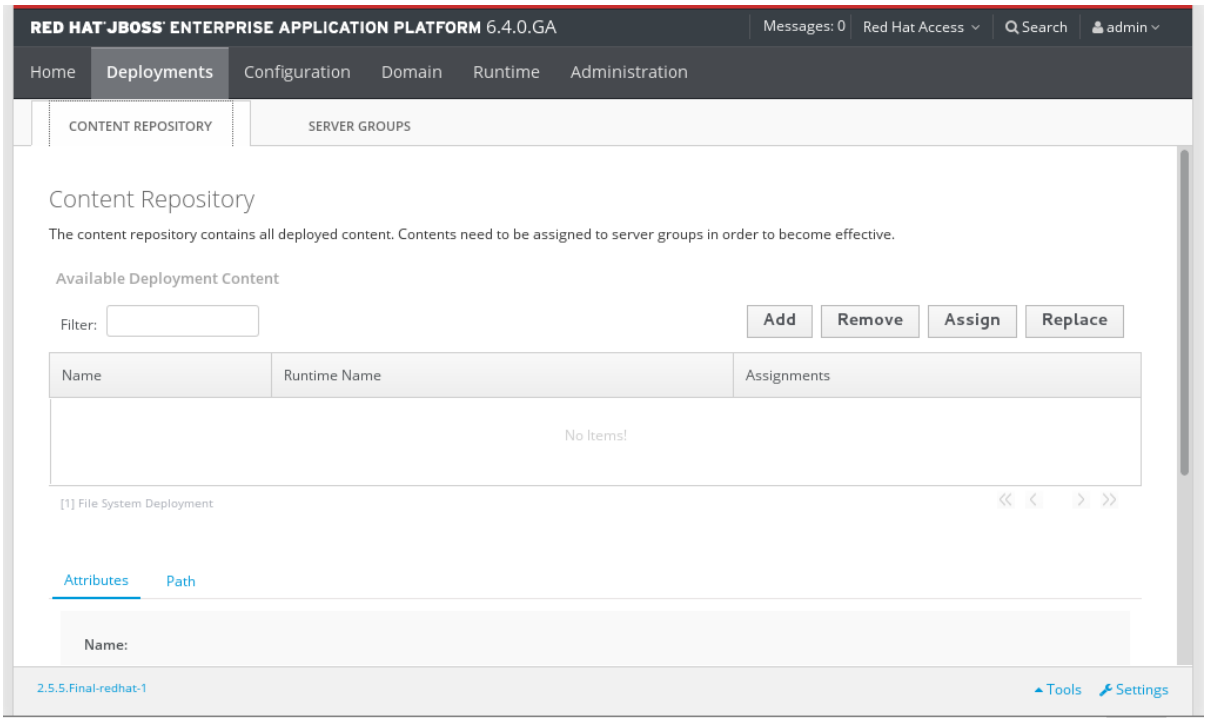
3.3.8. 管理コンソールでのデプロイメントの追加

前提条件

- *「管理コンソールへのログイン」*

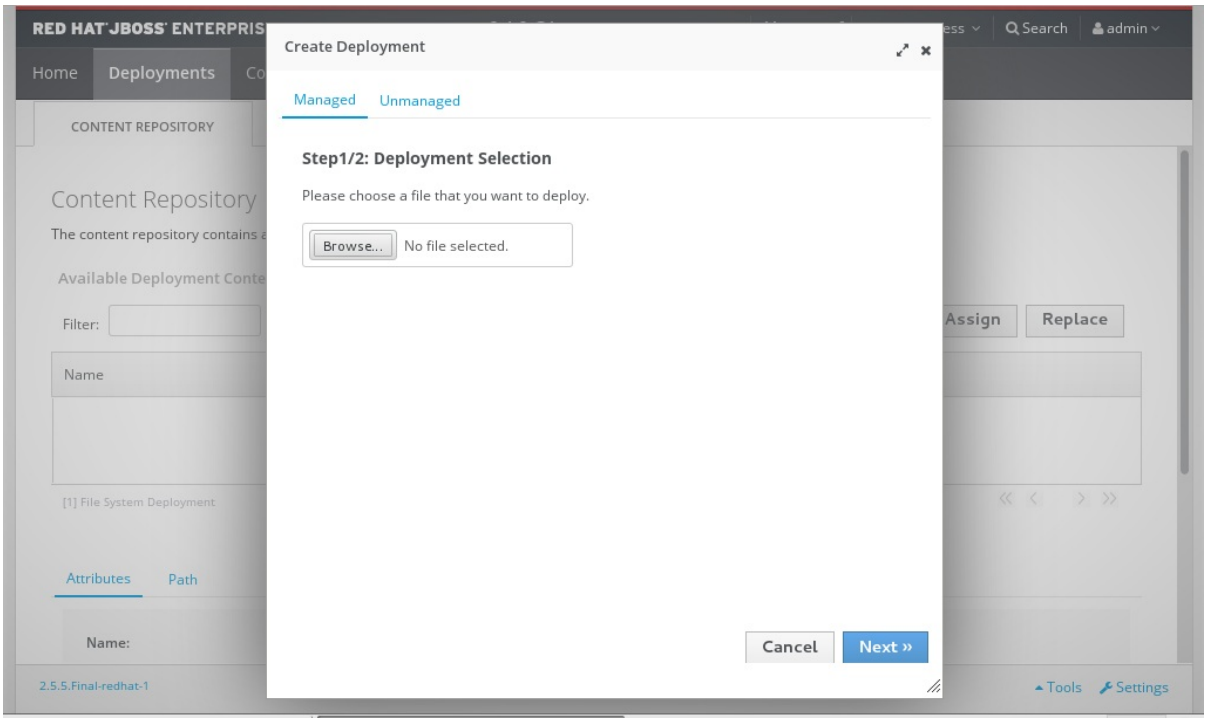
1. *コンソールの上部にある **Deployments** タブを選択します。*
2. *コンテンツリポジトリー タブで追加 を選択します。**Create Deployment** ダイアログボックスが表示されます。*

図3.7 スタンドアロンデプロイメントの管理



3. *ダイアログボックスで、**Browse** をクリックします。デプロイするファイルを参照し、これを選択してアップロードします。**Next** をクリックして先に進みます。*

図3.8 デプロイメントの選択



4. ***Create Deployments** ダイアログボックスで表示されるデプロイメント名とランタイム名を確認します。名前を確認したら、**Save** をクリックしてファイルをアップロードします。*

結果

選択したコンテンツがサーバーにアップロードされ、デプロイメント可能になります。

バグの報告

3.3.9. 管理コンソールでのサーバーの新規作成

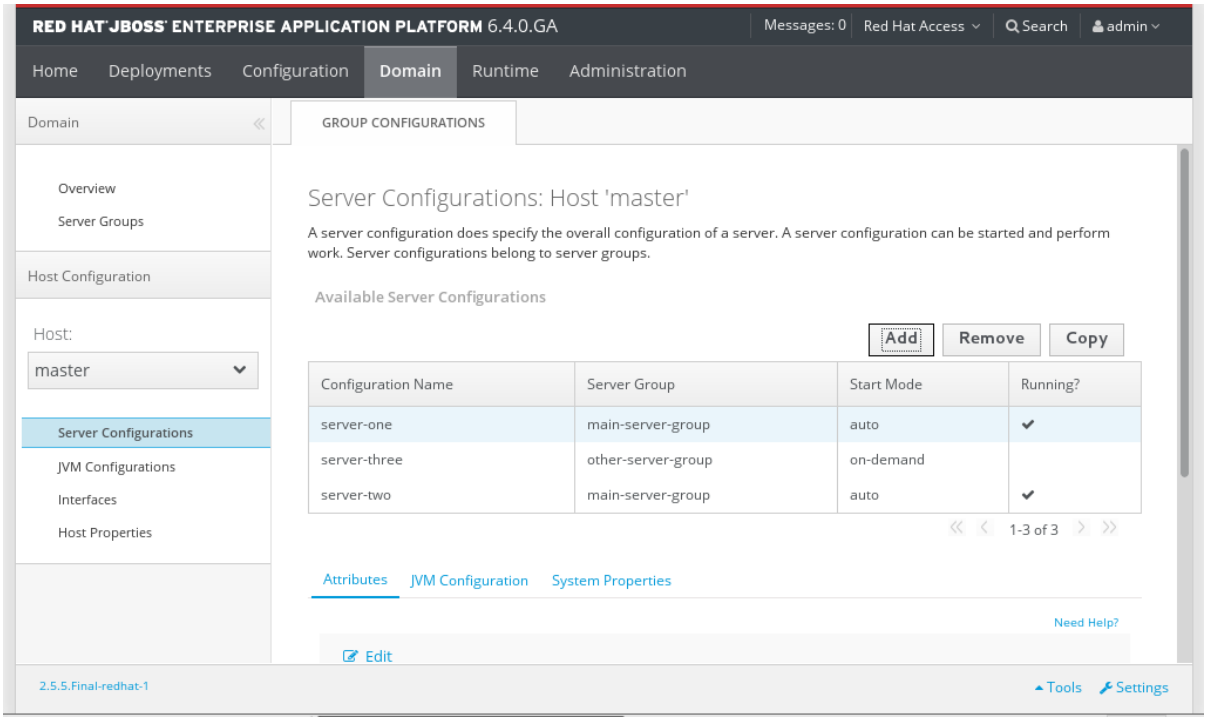
前提条件

- *「JBoss EAP 6 を管理対象ドメインとして起動」*
- *「管理コンソールへのログイン」*

手順3.3 サーバー設定の新規作成

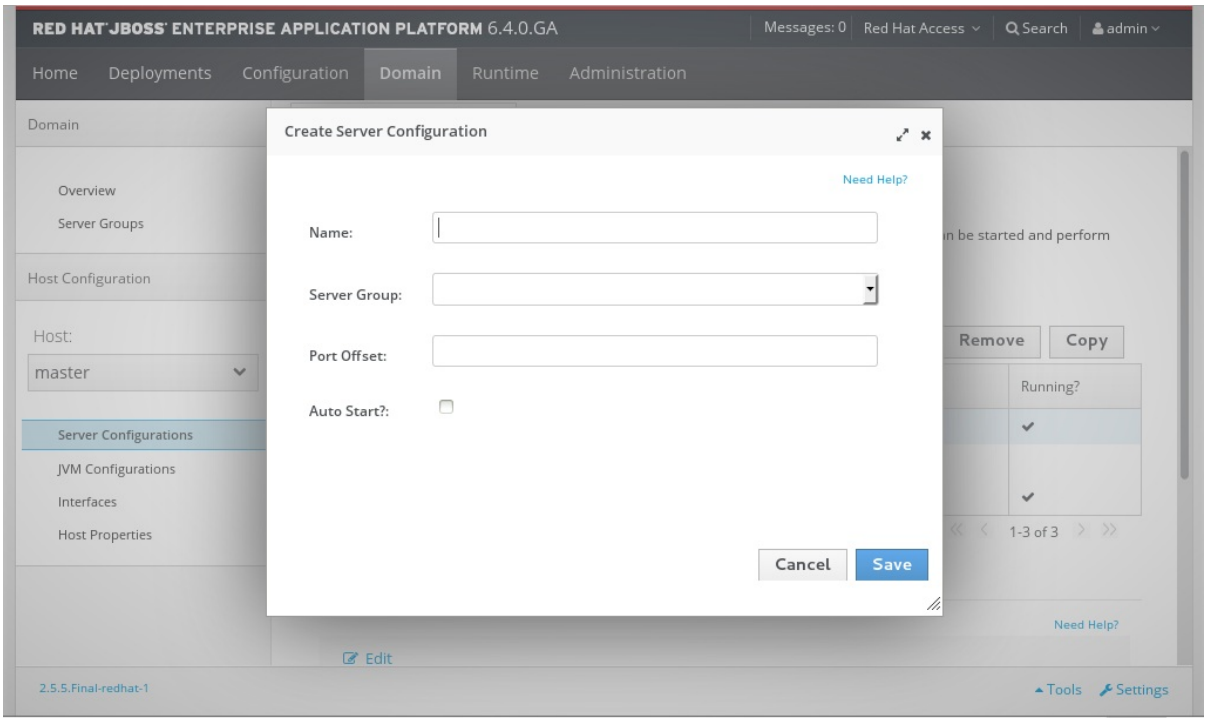
1. *管理コンソールの **Server Configurations** ページに移動します。*
   コンソールの上部から ドメイン タブを選択します。

   図3.9 サーバー設定



2. *左側のメニューの **Server Configurations** をクリックします。*
3. *設定の新規作成*
   a. ***Available Server Configurations** の表の上にある **Add** ボタンを選択します。*
   b. *「サーバー構成の 作成 」ダイアログに、基本的なサーバー設定を入力します。*
   c. ***Save** ボタンを選択して、新しいサーバー設定を保存します。*

図3.10 設定の新規作成



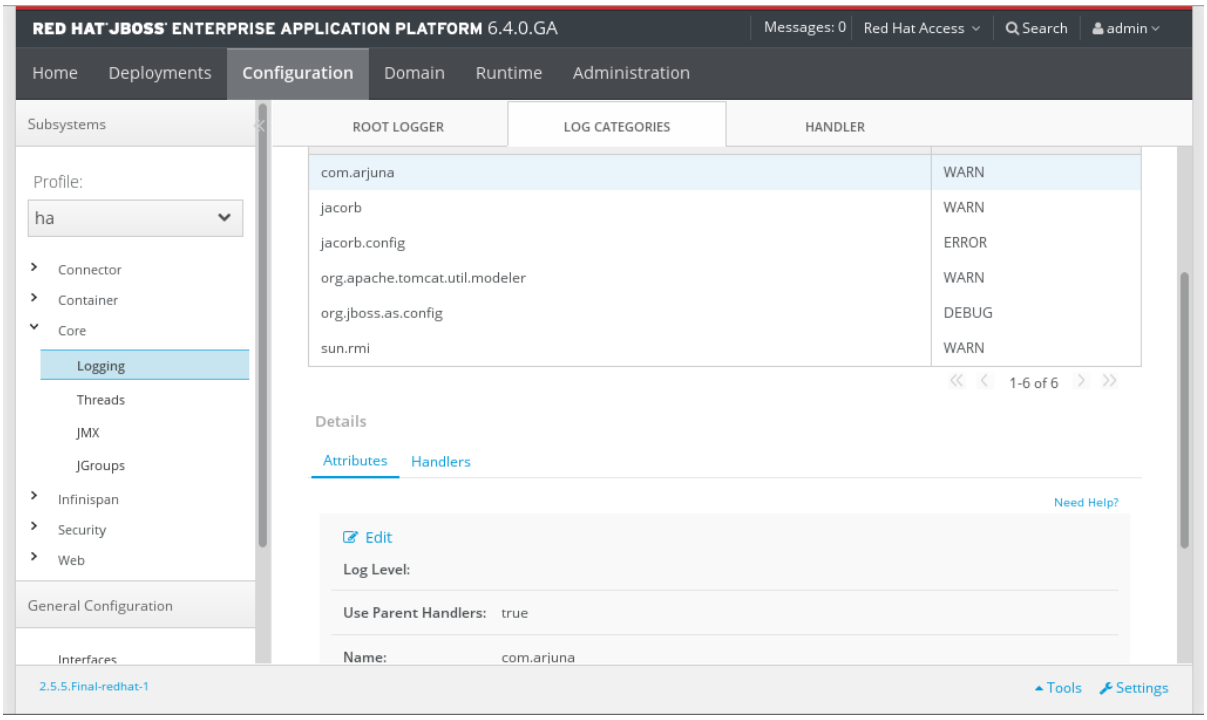
バグの報告

3.3.10. 管理コンソールを使用したデフォルトログレベルの変更

手順3.4 ロギングレベルの編集

1. 管理コンソールの **Logging** パネルに移動します。
   a. 管理対象ドメインを使用している場合は、コンソールの上部にある **Configuration** タブを選択し、コンソールの左側にあるドロップダウンリストから関連するプロファイルを選択します。
   b. 管理対象ドメインまたはスタンドアロンサーバーの場合は、コンソールの左側にある一覧から **Core** メニューを展開し、**Logging** エントリーをクリックします。
   c. コンソールの上部にある **Log Categories** タブをクリックします。

図3.11 ロギングパネル



2. *ロガー詳細の編集*
   ログカテゴリー テーブルのエントリーの詳細を編集します。

   a. ***Log Categories** テーブルのエントリーを選択し、以下の **Details** セクションで **Edit** をクリックします。*

   b. ***Log Level** ドロップダウンボックスでカテゴリーのログレベルを設定します。終了したら **Save** ボタンをクリックします。*

結果

該当するカテゴリーのログレベルが更新されます。

バグの報告

3.3.11. 管理コンソールでのサーバーグループの新規作成

前提条件

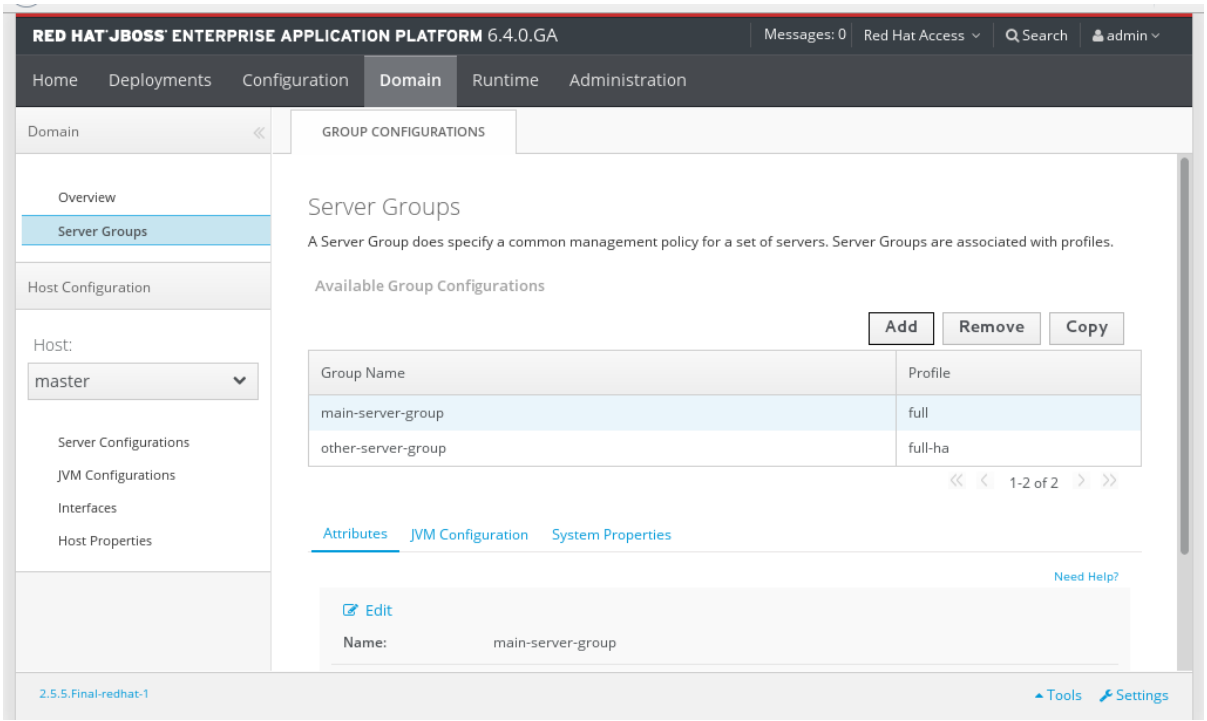
- *「管理コンソールへのログイン」*

手順3.5 サーバーグループの新規作成および追加

1. ***Server Groups** ビューに移動します。*
   コンソールの上部から ドメイン タブを選択します。

2. *左側の列で **Server Groups** を選択します。*

図3.12 サーバー グループ ビュー



3. *サーバーグループの追加*
   追加 ボタンをクリックして、新しいサーバーグループを追加します。

4. *サーバーグループの設定*

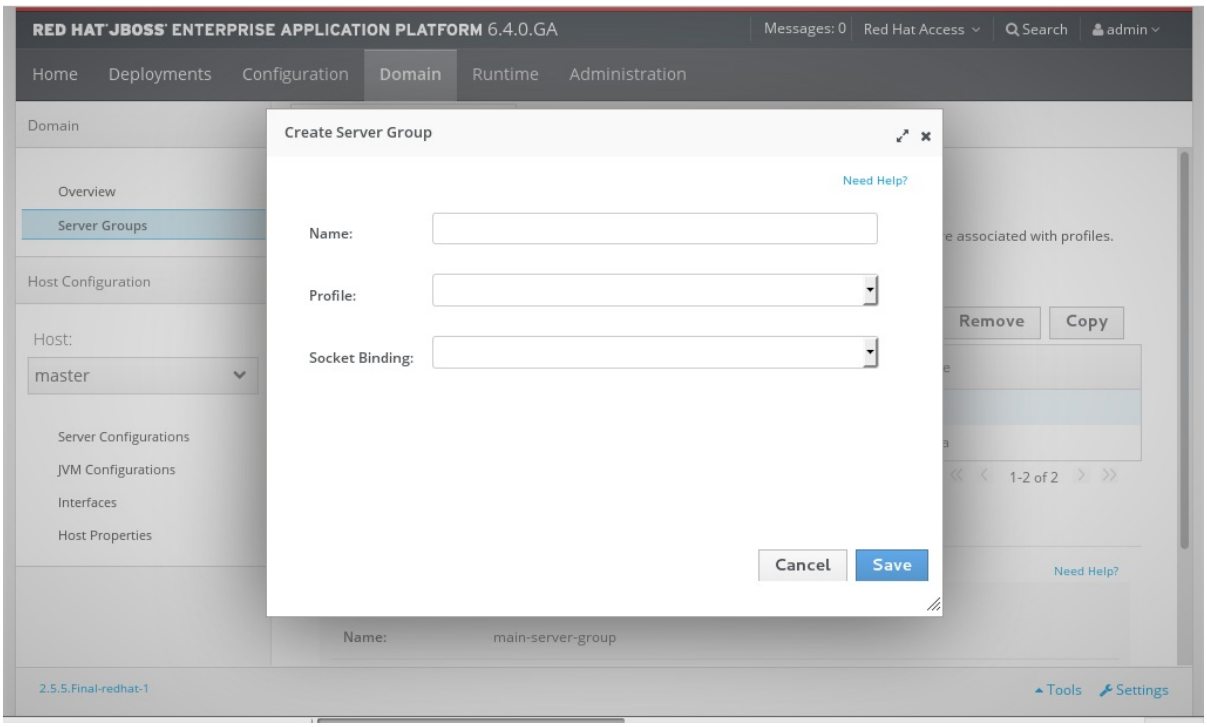
   a. *サーバーグループ名を入力します。*

   b. *サーバーグループのプロファイルを選択します。*

   c. *サーバーグループのソケットバインディングを選択します。*

   d. ***Save*** *ボタンをクリックして、新規グループを保存します。*

図3.13 サーバーグループの作成 ダイアログ



結果

新規サーバーグループが管理コンソールに表示されるようになります。

バグの報告

3.3.12. 管理コンソールでのログの表示

*JBoss EAP 6 管理コンソールでサーバーおよびアプリケーションログを表示して、エラー、パフォーマンスの問題、およびその他の問題の診断に役立ちます。管理コンソールログビューアーでログを表示できるようにするには、サーバーの **jboss.server.log.dir** ディレクトリーにある必要があります。JBoss EAP 6 の Log Viewer はユーザー RBAC ロールの割り当ても考慮するため、管理コンソールにログインしているユーザーはアクセスが許可されているログのみを表示できます。*

前提条件

- *「管理コンソールへのログイン」*

手順3.6 管理コンソールでの JBoss EAP 6 ログの表示

1. *管理コンソールの上部から **Runtime** タブを選択します。*
    a. *管理対象ドメインを使用している場合は、左側のメニューの **Change Server** ボタンを使用して、ログを表示する JBoss EAP 6 サーバーを選択します。*
2. *左側の **Platform** メニューを展開し、**Log Viewer** を選択します。*
3. *一覧からログファイルを選択し、表示 ボタンをクリックします。*

    ***Download** をクリックして、ログファイルをローカルマシンにダウンロードすることもできます。*

注記

管理コンソールログビューアーは、15MB を超えるログファイルを開こうとすると確認を表示します。

管理コンソールログビューアーは、非常に大きなログファイル(>100MB)を表示するテキストエディターの代わりとして使用するものではありません。管理コンソールログビューアーで非常に大きなログファイルを開くと Web ブラウザーがクラッシュする可能性があるため、大きなログファイルを常にダウンロードして、テキストエディターで開く必要があります。

4. *選択したログは、管理コンソールの新しいタブとして開きます。**LOG FILES** タブに戻り、直前の手順を繰り返すと、他のタブで複数のログファイルを開くことができます。*

バグの報告

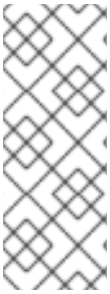
3.3.13. 管理コンソールでのカスタマーポータルの統合

access.redhat.com インターフェースを使用して、JBoss EAP インストールの管理コンソールを残さずに Red Hat カスタマーポータルのセクションを参照できます。

*管理コンソールのトップナビゲーションバーには、**Red Hat Access** のドロップダウンメニューが含まれます。このメニューをクリックすると、カスタマーポータルへのタスク固有の 3 つのリンクが表示されます。*

- *カスタマーポータルの検索*
- *ケースの作成*
- *ケースの修正*

これらの各リンクの機能は、以下で詳しく説明されています。



注記

これらのリンクのいずれかをクリックすると、カスタマーポータルにログインしていない場合は、ダイアログボックスが表示され、ログインが求められます。管理コンソールへのアクセスに使用するブラウザーセッションで、カスタマーポータルにログインする必要があります。あるブラウザーでカスタマーポータルにログインしていて、別のブラウザーを使用して管理コンソールにアクセスする場合は、ログインするよう要求されます。

カスタマーポータルの検索
***Search Customer Portal** をクリックすると、検索ボックスが含まれるページが表示されます。検索用語またはフレーズを入力して、ナレッジベースのアーティクルを見つけることができます。*

検索を実行したら、結果の一覧から項目を選択し、別のペインに表示される記事全体を表示できます。

ケースの作成
***Open Case** ページでは、新しいサポートケースを作成できます。*

新しいサポートケースを作成するには、完了するフォームが表示されます。推奨されるナレッジベースアーティクルの一覧がフォームのほかに提供されています。この一覧は、サポートケースに指定された詳細に基づいて更新されます。

ケースの修正
***Modify Case** ページでは、既存のサポートケースを表示および変更できます。*

結果を絞り込むには、検索を分類またはグループ化解除します。また、ケースの状態（オープン、クローズ、またはいずれか）で検索を限定することができます。

特定のサポートケースを選択したら、サポートケースの詳細を表示または更新し、コメントを追加できます。

バグの報告

3.4. 管理 CLI

3.4.1. 管理コマンドラインインターフェース (CLI)

管理コマンドラインインターフェース (CLI) は、JBoss EAP 6 のコマンドライン管理ツールです。

管理 CLI を使用して、サーバーの起動および停止、アプリケーションのデプロイおよびアンデプロイ、システム設定の構成、他の管理タスクの実行を行います。操作はバッチモードで実行可能で、複数のタスクをグループとして実行できます。

バグの報告

3.4.2. 管理 CLI の起動

要件:

- *「JBoss EAP 6 をスタンドアロンサーバーとして起動」*
- *「JBoss EAP 6 を管理対象ドメインとして起動」*

手順3.7 Linux または Microsoft Windows Server での CLI の起動

- *Linux での CLI の起動*
  ***EAP_HOME/bin/jboss-cli.sh** ファイルをコマンドラインで入力して実行します。*

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

- *Microsoft Windows Server での CLI の起動*
  ***EAP_HOME\bin\jboss-cli.bat** ファイルを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実行します。*

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

バグの報告

3.4.3. 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
```

バグの報告

3.4.4. 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3.8 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
```

  - 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に接続するには、**--connect** パラメーターを使用します。

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

  - **--connect** パラメーターを使用すると、サーバーのホストとポートを指定できます。ポート値 **9999** でアドレス **192.168.0.1** に接続するには、以下が適用されます。

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

バグの報告

3.4.5. 管理 CLI でのヘルプの取得

概要

CLI コマンドを学習したり、何ができるかわからない場合に、ガイダンスが必要になる場合があります。管理 CLI は、一般的なオプションおよびコンテキスト依存オプションに関するヘルプダイアログを特長としています。（操作コンテキストに依存する **help** コマンドには、スタンドアロンまたはドメインコントローラーへの確立された接続が必要になることに注意してください。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** extended 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、コマンドを入力してから **--help** を付けてください。

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

結果

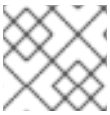
CLI ヘルプ情報が表示されます。

バグの報告

3.4.6. バッチモードでの管理 CLI の使用

概要

バッチ処理により、複数の操作リクエストをシーケンスにグループ化し、ユニットとして一緒に実行できます。シーケンスの操作リクエストのいずれかが失敗すると、操作のグループ全体がロールバックされます。



注記

バッチモードは条件付きステートメントをサポートしません。

前提条件

- 「管理 CLI の起動」
- 「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」

手順3.9 バッチモードのコマンドおよび操作

1. バッチモードの開始
   batch コマンドでバッチモードを 開始 します。

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

   バッチモードになると、プロンプトにハッシュ **(#)** が表示されます。

2. バッチへの操作要求の追加
   バッチモードでは、通常通りに操作リクエストを入力します。操作リクエストは、入力順にバッチに追加されます。

   操作リクエストのフォーマットに関する詳細は、「管理 CLI での操作およびコマンドの使用」を参照してください。

3. バッチの実行
   操作リクエストのシーケンス全体を入力したら、**run-batch** コマンドでバッチを実行します。

   ```
   [standalone@localhost:9999 / #] run-batch
   The batch executed successfully.
   ```

   バッチ処理で使用できるコマンドの完全リストは、「CLI のバッチモードコマンド」 を参照してください。

4. 外部ファイルに保存されたバッチコマンド

*頻繁に実行する **batch** コマンドは、外部テキストファイルに保存できます。**batch** コマンドの引数として完全パスをファイルに渡すか、**run- batch** コマンドの引数として直接実行することができます。*

*テキストエディターを使用して、**batch** コマンドファイルを作成できます。各コマンドは1行で記載する必要があり、**CLI** はこれにアクセスできる必要があります。*

*以下のコマンドは、**myscript.txt** ファイルをバッチモードで読み込みます。このファイルのすべてのコマンドを編集または削除できるようになりました。新しいコマンドを挿入することもできます。このバッチセッションでの変更は、**myscript.txt** ファイルに永続化されません。*

```
[standalone@localhost:9999 /] batch --file=myscript.txt
```

*次のコマンドは、**myscript.txt**ファイルに保存されている **batch** コマンドを即座に実行します。*

```
[standalone@localhost:9999 /] run-batch --file=myscript.txt
```

結果

入力された操作リクエストのシーケンスがバッチとして完了します。

バグの報告

3.4.7. CLI のバッチモードコマンド

*以下の表は、**JBoss EAP 6 CLI** で使用できる有効なバッチコマンドのリストを示しています。これらのコマンドはバッチ処理のみに使用できます。*

表3.2 CLI のバッチモードコマンド

コマンド名	説明
list-batch	現在のバッチのコマンドおよび操作のリスト
edit-batch-line line-number edited-command	編集する行番号と編集されたコマンドを提供して、現在のバッチの行を編集します。例： **edit-batch-line 2 data-source disable --name=ExampleDS**.
move-batch-line fromline toline	最初の引数としたい行番号と 2 つ目の引数としての新たポジションを指定して、バッチの行の順番を変えます。例： **move-batch-line 3 1**
remove-batch-line linenumber	指定の行で batch コマンドを削除します。例： **remove-batch-line 3**

コマンド名	説明
holdback-batch [batchname]	このコマンドを使用すると、現在のバッチを延期または保存できます。バッチ以外の CLI で何かを突然実行する場合は、これを使用します。この保留されたバッチに戻るには、CLI コマンドラインで再度 **batch** を入力します。 **holdback-batch** コマンドの使用中にバッチ名を指定すると、バッチはその名前の下に保存されます。名前付きのバッチに戻るには、コマンド **batchname** を使用します。batchname なしで **batch** コマンドを呼び出すと、新しい（名前のない）バッチが開始されます。名前のない保留されたバッチは 1 つのみ存在できます。 保留されたすべてのバッチの一覧を表示するには、**batch -l** コマンドを使用します。
discard-batch	現在アクティブなバッチを破棄します。

バグの報告

3.4.8. 管理 CLI での操作およびコマンドの使用

前提条件

- *「管理 CLI の起動」*
- *「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」*

手順3.10 要求の作成、設定、および実行

1. *操作要求の構築*
   操作リクエストを使用すると、管理モデルとの低レベルの対話が可能になります。サーバー設定を編集する制御方法を提供します。操作リクエストは 3 つの部分で構成されます。

   - *スラッシュ(/)で始まるアドレス。*
   - *コロン(:)で始まる操作名。*
   - *括弧（）内の任意のパラメーターセット。*

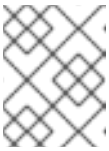
   a. *アドレスの特定*
   設定はアドレス指定可能なリソースの階層ツリーとして表されます。各リソースノードは異なる操作のセットを提供します。アドレスは、操作を実行するリソースノードを指定します。アドレスは以下の構文を使用します。

   ```
   /node-type=node-name
   ```

   - ***node-type** は、リソースノード種別です。これは、設定 XML の要素名にマッピングされます。*

- *node-name はリソースノード名です。これは、設定 XML の要素の **name** 属性にマッピングされます。*
- *リソースツリーの各レベルは、スラッシュ(/)で区切ります。*

*設定 XML ファイルを参照して、必要なアドレスを判断します。**EAP_HOME/standalone/configuration/standalone.xml** ファイルはスタンドアロンサーバーの設定を保持し、**EAP_HOME/domain/configuration/domain.xml** ファイルおよび **EAP_HOME/domain/configuration/host.xml** ファイルは管理対象ドメインの設定を保持します。*



注記

*ドメインモードで CLI コマンドを実行するには、ホストおよびサーバー仕様が必要です。例：**/host=master/server=server-one/subsystem=logging***

> *例3.5 操作アドレスの例*
>
> *ロギングサブシステムで操作を実行するには、操作要求に以下のアドレスを使用します。*
>
> ```
> /subsystem=logging
> ```
>
> *Java データソースに対して操作を実行するには、操作要求に以下のアドレスを使用します。*
>
> ```
> /subsystem=datasources/data-source=java
> ```

b. *操作の特定*
操作は異なるタイプのリソースノードによって異なります。操作では、以下の構文を使用します。

```
:operation-name
```

- *operation-name はリクエストする操作の名前です。*

*スタンドアロンサーバーのリソースアドレスで **read-operation-names** 操作を使用して、利用可能な操作を一覧表示します。*

> *例3.6 利用可能な操作*
>
> *ロギングサブシステムのすべての利用可能な操作をリストするために、スタンドアロンサーバーの以下の要求を入力します。*
>
> ```
> [standalone@localhost:9999 /] /subsystem=logging:read-operation-names
> {
>     "outcome" => "success",
>     "result" => [
>         "add",
>         "read-attribute",
>         "read-children-names",
>         "read-children-resources",
> ```

```
        "read-children-types",
        "read-operation-description",
        "read-operation-names",
        "read-resource",
        "read-resource-description",
        "remove",
        "undefine-attribute",
        "whoami",
        "write-attribute"
    ]
}
```

c. *パラメーターの決定*
各操作では異なるパラメーターが必要な場合があります。

パラメーターは以下の構文を使用します。

```
(parameter-name=parameter-value)
```

- ***parameter-name** は、パラメーターの名前です。*
- ***parameter-value** は、パラメーターの値です。*
- *複数のパラメーターはコンマ **(,)** で区切られます。*

*必要なパラメーターを確認するには、リソースノードで **read-operation-description** コマンドを実行し、操作名をパラメーターとして渡します。詳細は、例3.7「操作パラメーターの決定」 を参照してください。*

例3.7 操作パラメーターの決定

*ロギングサブシステムで **read-children-types** 操作に必要なパラメーターを決定するには、以下のように **read-operation-description** コマンドを入力します。*

```
[standalone@localhost:9999 /] /subsystem=logging:read-operation-
description(name=read-children-types)
{
    "outcome" => "success",
    "result" => {
        "operation-name" => "read-children-types",
        "description" => "Gets the type names of all the children under the selected
resource",
        "reply-properties" => {
            "type" => LIST,
            "description" => "The children types",
            "value-type" => STRING
        },
        "read-only" => true
    }
}
```

2. *完全操作要求の入力*

アドレス、操作、およびパラメーターが決定されたら、完全操作要求を入力します。

例3.8 操作要求の例

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-
resource(recursive=true)
```

結果

管理インターフェースは、サーバー設定の操作要求を実行します。

バグの報告

3.4.9. 管理 CLI で if-else 制御フローを使用

管理 CLI は、条件に基づいて実行するコマンドおよび操作のセットを選択できる **if - other** 制御フローをサポートします。**if** 条件は、**of** キーワードの後に指定された管理コマンドまたは操作の応答を評価するブール式です。

以下の項目をどれでも式に含めることができます。

- 条件演算子(**&&, ||**)
- 比較演算子(**>, >=, <=, ==, !=**)
- 式をグループ化および優先付けするかっこ

例3.9 管理 CLI コマンドでの**if** ステートメントの使用

この例では、システムプロパティー **test** の読み取りを試みます。**outcome** が **success** でない場合 (プロパティーが存在しないことを意味します)、システムプロパティーが追加され、**true** に設定されます。

```
if (outcome != success) of /system-property=test:read-resource
    /system-property=test:add(value=true)
end-if
```

上記の条件は、**outcome** を使用します。 これは、以下のように **of** キーワードの後の CLI コマンドが実行されると返されます。

```
[standalone@localhost:9999 /] /system-property=test:read-resource
{
    "outcome" => "failed",
    "failure-description" => "JBAS014807: Management resource '[(\"system-property\" => \"test\")]'
not found",
    "rolled-back" => true
}
```

例3.10 **if-else** statement with 管理 CLI コマンドの使用

この例では、サーバープロセスの起動タイプ（**STANDALONE** または **DOMAIN**）をチェックし、適切な CLI コマンドを実行して **ExampleDS** データソースを有効にします。

```
if (result == STANDALONE) of /:read-attribute(name=launch-type)
    /subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled, value=true)
else
    /profile=full/subsystem=datasources/data-source=ExampleDS:write-attribute(name=enabled,
value=true)
end-if
```

if-else control flow をファイルに指定すると（各行に 1 つずつ）管理 CLI コマンドを指定し、非対話的に実行するために **jboss-cli.sh** スクリプトに渡すことができます。

```
EAP_HOME/bin/jboss-cli.sh --connect --file=CLI_FILE
```

注記

入れ子の **if-else** ステートメントの使用はサポートされません。

バグの報告

3.4.10. 管理 CLI 設定オプション

管理 CLI 設定ファイル(**jboss-cli.xml**)は、CLI が起動されるたびにロードされます。**$EAP_HOME/bin** ディレクトリー、またはシステムプロパティー **jboss.cli.config** で指定されたディレクトリーのいずれかに存在する必要があります。

default-controller

connect コマンドがパラメーターなしで実行された場合に 接続 するコントローラーの設定。

default-controller パラメーター

host

コントローラーのホスト名。デフォルト： **localhost**

port

コントローラーに接続するポート番号。デフォルトは 9999 です。

validate-operation-requests

実行のためにリクエストがコントローラーに送信される前に、操作リクエストのパラメーターリストが検証されるかどうかを示します。Type: Boolean デフォルト： **true**

history

この要素には、コマンドおよび操作の履歴ログの設定が含まれます。

履歴 パラメーター

enabled

履歴 が有効になっているかどうかを示します。Type: Boolean デフォルト： **true**

file-name

履歴が保存されるファイルの名前。デフォルト = **.jboss-cli-history**.

file-dir

履歴が保存されるディレクトリー。Default = **$USER_HOME**

max-size

履歴ファイルの最大サイズ。デフォルトは 500 です。

resolve-parameter-values

操作リクエストをコントローラーに送信する前にコマンド引数（または操作パラメーター）の値として指定されたシステムプロパティーを解決するか、サーバー側で解決が発生するようにします。
Type: BooleanDefault = **false**.

connection-timeout

コントローラーとの接続を確立できる時間。タイプ： 整数デフォルトは 5000 秒です。

ssl

この要素には、SSL に使用されるキーストアとトラストストアの設定が含まれます。

警告

Red Hat は、影響を受けるすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSL を明示的に無効にすることを推奨します。

SSL パラメーター

vault

タイプ： **vaultType**

key-store

タイプ: 文字列

key-store-password

タイプ: 文字列

alias

タイプ: 文字列

key-password

タイプ: 文字列

trust-store

タイプ: 文字列

trust-store-password

タイプ: 文字列

modify-trust-store

true** に設定すると、認識されない証明書が受信されると、CLI はユーザーにプロンプトを表示し、トラストストアに保存できるようにします。Type: Boolean デフォルト： **true

vaultType

***code** および **module** の指定がない場合、デフォルトの実装が使用されます。**code** は指定され、**module** は指定されていない場合、Picketbox モジュールで指定されたクラスを探します。モジュール と コード が指定されている場合、「module」によって指定されたモジュールの コードによって指定されたクラスを探します。*

code

タイプ: 文字列

module

型: 文字列

silent

*情報およびエラーメッセージをターミナルに出力するかどうかを指定します。**false** が指定されている場合でも、設定が許可したり、出力ターゲットが> を使用してコマンドラインの一部として指定された場合、メッセージはロガーを使用してログに記録されます。デフォルトは **False** です。*

バグの報告

3.4.11. 管理 CLI コマンドのリファレンス

前提条件

- *「管理 CLI の起動」*

概要

トピック「管理 CLI でのヘルプの取得」では、一般的なオプションおよびコンテキスト機密オプションに関するヘルプダイアログなど、管理 CLI ヘルプ機能にアクセスする方法を説明します。help コマンドは操作コンテキストに依存し、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が確立されている必要があります。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。

表3.3

コマンド	説明

コマンド	説明
batch	新しいバッチを作成してバッチモードを開始するか、既存の保留中のバッチに応じてバッチモードを再度アクティブにします。保留されたバッチがない場合は、引数なしで呼び出されると新しいバッチが開始されます。名前のない保留されたバッチがある場合、このコマンドは再度アクティベートします。名前付きの保留されたバッチがある場合は、保留中のバッチ名を引数として指定することでアクティベートできます。
cd	現在のノードパスを引数に変更します。現在のノードパスは、アドレス部分を含まない操作要求のアドレスとして使用されます。操作要求にアドレスが含まれる場合、含まれるアドレスは現在のノードパスに対して相対的であると見なされます。現在のノードパスはノードタイプで終了します。この場合、logging:read-resource などの node-name を指定する操作を実行するには十分です。
明確な	画面を消去します。
command	既存の汎用タイプコマンドを追加、削除、および一覧表示できます。汎用型コマンドは、特定のノード型に割り当てられ、その型のインスタンスで実行できる操作の実行を可能にします。また、既存のインスタンスでその型によって公開されるプロパティーの編集も可能にします。
connect	指定されたホストおよびポートのコントローラに接続します。
connection-factory	接続ファクトリーを定義します。
data-source	データソースサブシステムで JDBC データソース設定を管理します。
deploy	ファイルパスで指定されたアプリケーションをデプロイするか、リポジトリーで無効にされている既存のアプリケーションを有効にします。引数を付けずに実行すると、既存のデプロイメントがすべて表示されます。
echo	JBoss EAP 6.4 からは、**echo** コマンドは指定されたテキストに出力されます。テキストはそのまま出力されるため、変数の使用は利用できません。 例: `echo *Phase one complete*`
help	ヘルプメッセージを表示します。***--commands*** 引数と共に使用して、指定のコマンドにコンテキストの機密結果を提供できます。
history	メモリーの CLI コマンド履歴を表示し、履歴の拡張が有効または無効であるかを表示します。引数と共に使用すると、必要に応じて履歴拡張を消去、無効化、および有効にできます。
jms-queue	メッセージングサブシステムで JMS キューを定義します。

コマンド	説明
jms-topic	メッセージングサブシステムで JMS トピックを定義します。
ls	ノードパスの内容を一覧表示します。デフォルトでは、ターミナルの幅全体を使用して結果が列に出力されます。***-l*** スイッチを使用すると、1 行に 1 つの名前で結果が出力されます。
pwd	現在の作業ノードの完全ノードパスを出力します。
quit	コマンドラインインターフェースを終了します。
read-attribute	値を表示し、引数によっては管理されたリソースの属性の詳細も表示します。
read-operation	指定された操作の詳細を表示します。 指定がない場合は使用できる操作をすべて表示します。
undeploy	目的のアプリケーションの名前を指定して実行する場合にアプリケーションをアンデプロイします。引数を指定して実行し、アプリケーションをリポジトリーから削除することもできます。アプリケーションの指定なしで実行された既存デプロイメントの一覧を出力します。
version	アプリケーションサーバーバージョンと環境情報を出力します。
xa-data-source	データソースサブシステムで JDBC XA データソース設定を管理します。

バグの報告

3.4.12. 管理 CLI 操作のリファレンス

管理 CLI の操作の公開

*管理 CLI の操作は、トピック「管理 CLI を使用した操作名の表示」で説明されている **read-operation-names** 操作を使用すると公開できます。操作の説明は、トピック「管理 CLI を使用した操作説明の表示」で説明されている **read-operation-descriptions** 操作を使用すると表示できます。*

表3.4 管理 CLI の操作

操作名	説明
add-namespace	namespaces 属性のマップに名前空間接頭辞のマッピングを追加します。
add-schema-location	schema-locations 属性のマップにスキーマロケーションマッピングを追加します。

操作名	説明
delete-snapshot	snapshots ディレクトリーからサーバー設定のスナップショットを削除します。
full-replace-deployment	以前にアップロードしたデプロイメントコンテンツを使用可能なコンテンツの一覧に追加し、ランタイムの同じ名前の既存コンテンツを置き換え、利用可能なコンテンツの一覧から置き換えたコンテンツを削除します。詳細は、リンクを参照してください。
list-snapshots	snapshots ディレクトリーに保存されているサーバー設定のスナップショットを一覧表示します。
read-attribute	選択したリソースの属性の値を表示します。
read-children-names	指定の型を持つ選択したリソースの配下にある子の名前をすべて表示します。
read-children-resources	指定のタイプであるすべての子リソースに関する情報を表示します。
read-children-types	選択したリソースの配下にある子すべての型名を表示します。
read-config-as-xml	現在の設定を読み込み、XML 形式で表示します。
read-operation-description	特定のリソースに対する操作の詳細を表示します。
read-operation-names	特定のリソースに対する全操作の名前を表示します。
read-resource	モデルリソースの属性値および任意の子リソースの基本情報もしくは詳細情報を表示します。
read-resource-description	リソースの属性、子リソースのタイプ、および操作についての詳細を表示します。
reload	すべてのサービスを終了し、再起動することでサーバーをリロードします。
remove-namespace	namespaces 属性マップから名前空間接頭辞のマッピングを削除します。
remove-schema-location	schema-locations 属性マップからスキーマロケーションマッピングを削除します。
replace-deployment	ランタイムの既存のコンテンツを新しいコンテンツに置き換えます。新しいコンテンツを事前にデプロイメントコンテンツリポジトリーにアップロードする必要があります。
resolve-expression	式を、式へ解析可能な入力または文字列として許可し、ローカルのシステムプロパティーおよび環境変数に対して解決する操作です。

操作名	説明
resolve-internet-address	インターフェース解決基準を取り、その基準と一致するローカルマシンの IP アドレスを見つけます。一致する IP アドレスが見つからない場合は失敗します。
server-set-restart-required	再起動が必要なモードにサーバーを設定します。
shutdown	**System.exit(0)への呼び出しにより、サーバーをシャットダウンします。**
start-servers	現在実行されていないすべての設定済みサーバーを管理対象ドメインで起動します。
stop-servers	管理対象ドメインで現在実行しているすべてのサーバーを停止します。
take-snapshot	サーバー設定のスナップショットを作成し、snapshots ディレクトリーに保存します。
upload-deployment-bytes	含まれたバイトアレイのデプロイメントコンテンツをデプロイメントコンテンツリポジトリーに追加すべきであることを示します。この操作は、コンテンツをランタイムにデプロイすべきかは指定していません。
upload-deployment-stream	対象入力ストリームインデックスで利用可能なデプロイメントコンテンツをデプロイメントコンテンツレポジトリーに追加すべきか指定します。この操作は、コンテンツをランタイムにデプロイすべきかは指定していません。
upload-deployment-url	対象の URL で利用可能なデプロイメントコンテンツをデプロイメントコンテンツリポジトリーに追加すべきかを指定します。この操作は、コンテンツをランタイムにデプロイすべきかは指定していません。
validate-address	操作のアドレスを検証します。
write-attribute	選択したリソースの属性値を設定します。

バグの報告

3.4.13. 管理 CLI におけるプロパティーの置換

JBoss EAP 6 は、管理インターフェースの Preset 要素およびプロパティー式の使用をサポートします。これらの式は、コマンドの実行中に定義された値に解決されます。

以下のプロパティーは式に置き換えることができます。

- *操作リクエストの操作アドレス部分（ノードタイプまたは名前として）。*
- *操作名。*
- *操作パラメーター名。*

- ヘッダー名および値。
- コマンド名。
- コマンド引数名。

デフォルトでは、CLI は引数またはパラメーターの値以外の各行に対してプロパティー置換を実行します。引数およびパラメーターの値は起動時にサーバーで解決されます。引数またはパラメーターの値のプロパティー置換を管理 CLI クライアントで行い、解決された値をサーバーに送信する必要がある場合は、以下の手順を実行します。

手順3.11 管理 CLI でのプロパティー置換の有効化

1. **EAP_HOME/bin/jboss-cli.xml** ファイルを開きます。
2. **resolve-parameter-values** パラメーターを見つけ、値を **true** に変更します（デフォルトは **false** です）。

```
<!-- whether to resolve system properties specified as command argument or
operation parameter values in the Management CLI VM before sending the operation
requests to the controller -->
    <resolve-parameter-values>true</resolve-parameter-values>
```

この要素は、操作リクエストのパラメーターの値とコマンド引数の値のみに影響します。他のコマンドラインには影響しません。これは、パラメーター/引数の値でない限り、**resolve-parameter-values** 要素の値に関係なく、コマンドラインに存在するシステムプロパティーが行の解析中に解決されることを意味します。

他の管理 CLI の設定オプションについては、「管理 CLI 設定オプション」を参照してください。

管理 CLI コマンドで使用されるシステム値は、すでに定義されている必要があることに注意してください。管理 CLI インスタンスの起動時に、**--properties=/path/to/file.properties** 引数または 1 つ以上の **-Dkey=VALUE** パラメーターを含める必要があります。プロパティーファイルは標準の key=value 構文を使用します。

プロパティーキーは、構文 **${MY_VAR}** を使用して管理 CLI コマンドに表示されます。

例3.11 例： 管理 CLI コマンドでのプロパティーの使用

```
/subsystem=datasources/data-source=${datasourcename}:add(connection-
url=jdbc:oracle:thin:@server:1521:ora1, jndi-name=java:/jboss/${name}, driver-
name=${drivername})
```

バグの報告

3.5. 管理 CLI 操作

3.5.1. 管理 CLI によるリソースの属性の表示

前提条件

- 「管理 CLI の起動」

概要

***read-attribute** 操作は、選択した属性の現在のランタイム値を読み取るために使用されるグローバル操作です。これは、ユーザーが設定した値のみを公開するために使用できます（デフォルト値または未定義の値を無視します）。要求プロパティーには、以下のパラメーターが含まれます。*

要求プロパティー

name

選択したリソース下で値を取得する属性の名前。

include-defaults

*ユーザーが設定した属性のみを表示したり、デフォルト値を無視するように操作結果を制限したりするために **false** に設定されるブール値パラメーターです。*

手順3.12 選択した属性の現在のランタイム値を表示

- ***read-attribute** 操作の実行*
  *管理 CLI で **read-attribute** 操作を使用してリソース属性の値を表示します。操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」 のトピックを参照してください。*

```
[standalone@localhost:9999 /]:read-attribute(name=name-of-attribute)
```

***read-attribute** 操作の利点は、特定の属性の現在のランタイム値を公開する機能です。**read-resource** 操作でも同様の結果が得られますが、**include-runtime** 要求プロパティーを追加した場合のみ、そのノードで利用可能な全リソース一覧の一部としてしか実行できません。**read-attribute** 操作は、以下の例のように、詳細な属性クエリーを行うことを目的としています。*

*例3.12 **read-attribute** 操作を実行してパブリックインターフェース IP を公開します。*

*公開する属性の名前が分かっている場合は、**read-attribute** を使用して現在のランタイムの正確な値を返すことができます。*

```
[standalone@localhost:9999 /] /interface=public:read-attribute(name=resolved-address)
{
    "outcome" => "success",
    "result" => "127.0.0.1"
}
```

***resolved-address** 属性はランタイムの値であるため、標準の **read-resource** 操作の結果には表示されません。*

```
[standalone@localhost:9999 /] /interface=public:read-resource
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
```

```
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}
```

resolved-address およびその他のランタイム値を表示するには、**include-runtime** リクエストプロパティーを使用する必要があります。

```
[standalone@localhost:9999 /] /interface=public:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "resolved-address" => "127.0.0.1",
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
        "virtual" => undefined
    }
}
```

結果

現在のランタイム属性値が表示されます。

バグの報告

3.5.2. 管理 CLI でのアクティブユーザーの表示

前提条件

- 「管理 CLI の起動」

概要

whoami 操作は、現在アクティブなユーザーの属性を識別するために使用されるグローバル操作です。この操作は、ユーザー名の ID と、割り当てられたレルムを公開します。**whoami** 操作は、管理者が複数のレルムで複数のユーザーアカウントを管理する場合や、複数のターミナルセッションおよびユーザーアカウントを持つドメインインスタンス全体でアクティブなユーザーを追跡するのに役立ちます。

手順3.13 **whoami** 操作を使用した管理 CLI でのアクティブユーザーの表示

- **whoami** 操作の実行
  管理 CLI で **whoami** 操作を使用してアクティブなユーザーアカウントを表示します。

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

  以下の例は、スタンドアロンサーバーインスタンスで **whoami** 操作を使用して、アクティブなユーザーが username で、ユーザーが **ManagementRealm** レルムに割り当てられていることを示しています。

  例3.13 スタンドアロンインスタンスでの **whoami** の使用

  ```
  [standalone@localhost:9999 /]:whoami
  {
      "outcome" => "success",
      "result" => {"identity" => {
          "username" => "username",
          "realm" => "ManagementRealm"
      }}
  }
  ```

結果

現在アクティブなユーザーのアカウントが表示されます。

バグの報告

3.5.3. 管理 CLI でのシステムおよびサーバー情報の表示

前提条件

- 「管理 CLI の起動」

手順3.14 管理 CLI でのシステムおよびサーバー情報の表示

- **version** コマンドの実行
  管理 CLI で **version** コマンドを入力します。

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

結果

アプリケーションサーバーのバージョンと環境情報が表示されます。

バグの報告

3.5.4. 管理 CLI を使用した操作説明の表示

前提条件

- 「管理 CLI の起動」

手順3.15 管理 CLI でのコマンドの実行

- **read-operation-description** 操作を実行します。
  管理 CLI から **read-operation-description** を使用して、操作に関する情報を表示します。操作には、表示する操作を示すために、キーと値のペアの形式で追加のパラメーターが必要です。操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」のトピックを参照してください。

```
[standalone@localhost:9999 /]:read-operation-description(name=name-of-operation)
```

例3.14 list-snapshots 操作の説明表示

以下の例は、**list-snapshots** 操作を説明する方法を示しています。

```
[standalone@localhost:9999 /] :read-operation-description(name=list-snapshots)
{
    "outcome" => "success",
    "result" => {
        "operation-name" => "list-snapshots",
        "description" => "Lists the snapshots",
        "request-properties" => {},
        "reply-properties" => {
            "type" => OBJECT,
            "value-type" => {
                "directory" => {
                    "type" => STRING,
                    "description" => "The directory where the snapshots are stored",
                    "expressions-allowed" => false,
                    "required" => true,
                    "nillable" => false,
                    "min-length" => 1L,
                    "max-length" => 2147483647L
                },
                "names" => {
                    "type" => LIST,
                    "description" => "The names of the snapshots within the snapshots directory",
                    "expressions-allowed" => false,
                    "required" => true,
                    "nillable" => false,
                    "value-type" => STRING
                }
            }
        },
        "access-constraints" => {"sensitive" => {"snapshots" => {"type" => "core"}}},
```

```
        "read-only" => false
    }
}
```

結果

選択した操作に関する説明が表示されます。

バグの報告

3.5.5. 管理 CLI を使用した操作名の表示

前提条件

- *「管理 CLI の起動」*

手順3.16 管理 CLI でのコマンドの実行

- ***read-operation-names** 操作を実行します。*
  *管理 CLI から **read-operation-names** 操作を使用して、利用可能な操作の名前を表示します。操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」 のトピックを参照してください。*

  ```
  [standalone@localhost:9999 /]:read-operation-names
  ```

例3.15 管理 CLI を使用した操作名の表示

*以下の例は、**read-operation-names** 操作を説明する方法を示しています。*

```
[standalone@localhost:9999 /]:read-operation-names
{
    "outcome" => "success",
    "result" => [
        "add-namespace",
        "add-schema-location",
        "delete-snapshot",
        "full-replace-deployment",
        "list-snapshots",
        "read-attribute",
        "read-children-names",
        "read-children-resources",
        "read-children-types",
        "read-config-as-xml",
        "read-operation-description",
        "read-operation-names",
        "read-resource",
        "read-resource-description",
        "reload",
        "remove-namespace",
        "remove-schema-location",
        "replace-deployment",
        "resolve-expression",
        "resolve-internet-address",
```

```
        "server-set-restart-required",
        "shutdown",
        "take-snapshot",
        "undefine-attribute",
        "upload-deployment-bytes",
        "upload-deployment-stream",
        "upload-deployment-url",
        "validate-address",
        "validate-operation",
        "whoami",
        "write-attribute"
    ]
}
```

結果

利用可能な操作名が表示されます。

バグの報告

3.5.6. 管理 CLI を使用した利用可能なリソースの表示

前提条件

- *「管理 CLI の起動」*

概要

***read-resource** 操作は、リソース値を読み取るために使用されるグローバル操作です。これは、現在のノードまたは子ノードのリソースに関する基本情報または完全な情報のいずれかを、操作結果の範囲を拡張するか、または制限する要求プロパティーの範囲を公開するために使用できます。要求プロパティーには、以下のパラメーターが含まれます。*

要求プロパティー

recursive

子リソースに関する詳細情報を再帰的に含めるかどうか。

recursive-depth

含まれる子リソースの情報の深さ。

proxies

再帰クエリーにリモートリソースを含めるかどうか。たとえば、ドメインコントローラーのクエリーにスレーブホストコントローラーからのホストレベルのリソースを追加します。

include-runtime

*永続設定から取得されない属性値などの応答にランタイム属性を含めるかどうか。この要求プロパティーはデフォルトで **false** に設定されます。*

include-defaults

*デフォルト属性の読み取りを有効または無効にするブール値要求プロパティー。**false** に設定すると、ユーザーが設定した属性のみが返され、未定義のままの属性は無視されます。*

管理 CLI でのコマンドの実行

***read-resource** 操作を実行します。*

*管理 CLI で **read-resource** 操作を使用して利用可能なリソースを表示します。*

```
[standalone@localhost:9999 /]:read-resource
```

*以下の例は、スタンドアロンサーバーインスタンスで **read-resource** 操作を使用して一般的なリソース情報を公開する方法を示しています。結果は **standalone.xml** 設定ファイルに類似し、サーバーインスタンスにインストールまたは設定されたシステムリソース、拡張、インターフェース、およびサブシステムを表示します。これらはさらに直接クエリーできます。*

*例3.16 ルートレベルでの **read-resource** 操作の使用*

```
[standalone@localhost:9999 /]:read-resource
{
    "outcome" => "success",
    "result" => {
        "management-major-version" => 1,
        "management-micro-version" => 0,
        "management-minor-version" => 7,
        "name" => "localhost",
        "namespaces" => [],
        "product-name" => "EAP",
        "product-version" => "6.4.0.GA",
        "profile-name" => undefined,
        "release-codename" => "Janus",
        "release-version" => "7.5.0.Final-redhat-17",
        "schema-locations" => [],
        "core-service" => {
            "service-container" => undefined,
            "server-environment" => undefined,
            "module-loading" => undefined,
            "platform-mbean" => undefined,
            "management" => undefined,
            "patching" => undefined
        },
        "deployment" => undefined,
        "deployment-overlay" => undefined,
        "extension" => {
            "org.jboss.as.clustering.infinispan" => undefined,
            "org.jboss.as.connector" => undefined,
            "org.jboss.as.deployment-scanner" => undefined,
            "org.jboss.as.ee" => undefined,
            "org.jboss.as.ejb3" => undefined,
            "org.jboss.as.jaxrs" => undefined,
            "org.jboss.as.jdr" => undefined,
            "org.jboss.as.jmx" => undefined,
            "org.jboss.as.jpa" => undefined,
            "org.jboss.as.jsf" => undefined,
            "org.jboss.as.logging" => undefined,
            "org.jboss.as.mail" => undefined,
            "org.jboss.as.naming" => undefined,
            "org.jboss.as.pojo" => undefined,
            "org.jboss.as.remoting" => undefined,
```

```
            "org.jboss.as.sar" => undefined,
            "org.jboss.as.security" => undefined,
            "org.jboss.as.threads" => undefined,
            "org.jboss.as.transactions" => undefined,
            "org.jboss.as.web" => undefined,
            "org.jboss.as.webservices" => undefined,
            "org.jboss.as.weld" => undefined
        },
        "interface" => {
            "management" => undefined,
            "public" => undefined,
            "unsecure" => undefined
        },
        "path" => {
            "jboss.server.temp.dir" => undefined,
            "user.home" => undefined,
            "jboss.server.base.dir" => undefined,
            "java.home" => undefined,
            "user.dir" => undefined,
            "jboss.server.data.dir" => undefined,
            "jboss.home.dir" => undefined,
            "jboss.server.log.dir" => undefined,
            "jboss.server.config.dir" => undefined,
            "jboss.controller.temp.dir" => undefined
        },
        "socket-binding-group" => {"standard-sockets" => undefined},
        "subsystem" => {
            "jaxrs" => undefined,
            "jsf" => undefined,
            "jca" => undefined,
            "jmx" => undefined,
            "threads" => undefined,
            "webservices" => undefined,
            "sar" => undefined,
            "remoting" => undefined,
            "infinispan" => undefined,
            "weld" => undefined,
            "ejb3" => undefined,
            "transactions" => undefined,
            "datasources" => undefined,
            "deployment-scanner" => undefined,
            "logging" => undefined,
            "jdr" => undefined,
            "pojo" => undefined,
            "jpa" => undefined,
            "naming" => undefined,
            "ee" => undefined,
            "mail" => undefined,
            "web" => undefined,
            "resource-adapters" => undefined,
            "security" => undefined
        },
        "system-property" => undefined
    }
}
```

*子ノードに対する **read-resource** 操作を実行します。*

***read-resource** 操作を実行して、ルートから子ノードをクエリーできます。オペレーションの構造は最初に公開するノードを定義し、続いてこれに対して実行する操作を追加します。*

```
[standalone@localhost:9999 /]/subsystem=web/connector=http:read-resource
```

*以下の例では、特定の Web サブシステムノードに **read-resource** 操作を行うと、Web サブシステムコンポーネントに関する特定のリソース情報を公開できます。*

例3.17 ルートノードからの子ノードリソースの公開

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource
{
    "outcome" => "success",
    "result" => {
        "configuration" => undefined,
        "enable-lookups" => false,
        "enabled" => true,
        "executor" => undefined,
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "name" => "http",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

***cd** コマンドを使用して子ノードに移動し、**read-resource** 操作を直接実行しても、同じ結果が得られます。*

例3.18 ディレクトリーの変更による子ノードリソースの公開

```
[standalone@localhost:9999 /] cd subsystem=web
```

```
[standalone@localhost:9999 subsystem=web] cd connector=http
```

```
[standalone@localhost:9999 connector=http] :read-resource
{
    "outcome" => "success",
    "result" => {
        "configuration" => undefined,
        "enable-lookups" => false,
        "enabled" => true,
        "executor" => undefined,
```

```
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "name" => "http",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

再帰的なパラメーターを使用して結果にアクティブな値を含める

再帰的なパラメーターを使用すると、すべての属性の値 (永続的でない値、起動時に渡された値、ランタイムモデルでアクティブな他の属性など) を公開できます。

```
[standalone@localhost:9999 /]/interface=public:read-resource(include-runtime=true)
```

*以前の例と比較すると、**include-runtime** 要求プロパティーを含めると、HTTP コネクターによって送受信されたバイトなどの追加のアクティブな属性が公開されます。*

*例3.19 **include-runtime** パラメーターを使用して追加のアクティブな値を公開*

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource(include-
runtime=true)
{
    "outcome" => "success",
    "result" => {
        "any" => undefined,
        "any-address" => undefined,
        "any-ipv4-address" => undefined,
        "any-ipv6-address" => undefined,
        "inet-address" => expression "${jboss.bind.address:127.0.0.1}",
        "link-local-address" => undefined,
        "loopback" => undefined,
        "loopback-address" => undefined,
        "multicast" => undefined,
        "name" => "public",
        "nic" => undefined,
        "nic-match" => undefined,
        "not" => undefined,
        "point-to-point" => undefined,
        "public-address" => undefined,
        "resolved-address" => "127.0.0.1",
        "site-local-address" => undefined,
        "subnet-match" => undefined,
        "up" => undefined,
```

```
        "virtual" => undefined
    }
}
```

バグの報告

3.5.7. 管理 CLI を使用した利用可能なリソース説明の表示

前提条件

- 「管理 CLI の起動」

管理 CLI でのコマンドの実行

read-resource-description 操作を実行します。

管理 CLI から **read-resource-description** 操作を使用して、利用可能なリソースを読み取りおよび表示します。操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」のトピックを参照してください。

```
[standalone@localhost:9999 /]:read-resource-description
```

オプションパラメーターの使用

read-resource-description 操作では、追加のパラメーターを使用できます。

- **operations** パラメーターを使用して、リソースの操作の説明を追加します。

  ```
  [standalone@localhost:9999 /]:read-resource-description(operations=true)
  ```

- 継承された パラメーターを使用して、リソースの継承された操作の説明を追加または除外します。デフォルトの状態は **true** です。

  ```
  [standalone@localhost:9999 /]:read-resource-description(inherited=false)
  ```

- **recursive** パラメーターを使用して、子リソースの再帰的な記述を追加します。

  ```
  [standalone@localhost:9999 /]:read-resource-description(recursive=true)
  ```

- **locale** パラメーターを使用して、リソースの説明を取得します。**null** の場合、デフォルトのロケールが使用されます。

  ```
  [standalone@localhost:9999 /]:read-resource-description(locale=true)
  ```

- **access-control** パラメーターを使用して、このリソースの現在の呼び出し元が持つパーミッションに関する情報を取得します。

  ```
  [standalone@localhost:9999 /]:read-resource-description(access-control=none)
  ```

結果

利用可能なリソースの説明が表示されます。

バグの報告

3.5.8. 管理 CLI を使用したアプリケーションサーバーのリロード

前提条件

- 「管理 CLI の起動」

管理 CLI で **reload** 操作を使用してすべてのサービスをシャットダウンし、JBoss EAP インスタンスを再起動します。JVM 自体は再起動されないことに注意してください。リロード が完了すると、管理 CLI は自動的に再接続します。

操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」 を参照してください。

例3.20 アプリケーションのリロード

```
[standalone@localhost:9999 /]reload
{"outcome" => "success"}
```

バグの報告

3.5.9. 管理 CLI を使用したアプリケーションサーバーのシャットダウン

前提条件

- 「管理 CLI の起動」

手順3.17 アプリケーションサーバーのシャットダウン

- シャットダウン 操作の実行
  - 管理 CLI で シャットダウン 操作を使用し、**System.exit(0)** システムコールを介してサーバーをシャットダウンします。操作リクエストの詳細は、「管理 CLI での操作およびコマンドの使用」 のトピックを参照してください。
    - スタンドアロンモードでは、次のコマンドを使用します。

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

    - ドメインモードでは、適切なホスト名を指定して次のコマンドを使用します。

      ```
      [domain@localhost:9999 /]shutdown --host=master
      ```

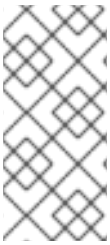
  - デタッチした CLI インスタンスへ接続し、サーバーをシャットダウンするには、次のコマンドを実行します。

    ```
    jboss-cli.sh --connect command=shutdown
    ```

- リモート CLI インスタンスへ接続し、サーバーをシャットダウンするには、次のコマンドを実行します。

```
[disconnected /] connect IP_ADDRESS
[standalone@IP_ADDRESS:9999 /] shutdown
```

IP_ADDRESS はインスタンスの IP アドレスに置き換えます。



注記

shutdown コマンドに **--restart=true** 引数を追加すると（以下に示されるように）、サーバーを再起動するよう要求されます。

```
[standalone@localhost:9999 /]shutdown --restart=true
```

結果

アプリケーションサーバーがシャットダウンしている。ランタイムが利用できないため、管理 CLI は切断されます。

バグの報告

3.5.10. 管理 CLI での属性の設定

前提条件

- 「管理 CLI の起動」

概要

write-attribute 操作は、選択したリソース属性の書き込みまたは変更に使用されるグローバル操作です。操作を使用して永続的な変更を行い、管理対象サーバーインスタンスの設定を変更できます。要求プロパティーには、以下のパラメーターが含まれます。

要求プロパティー

名前
: 選択されたリソース下で値を設定する属性の名前。

値
: 選択したリソース下の属性の必要な値。基礎となるモデルが null 値をサポートする場合は null にすることができます。

手順3.18 管理 CLI でのリソース属性の設定

- **write-attribute** 操作の実行
  管理 CLI で **write-attribute** 操作を使用してリソース属性の値を変更します。操作は、リソースの子ノードまたは完全なリソースパスが指定された管理 CLI のルートノードで実行できます。

例3.21 **write-attribute** 操作を使用したデプロイメントスキャナーの無効化

以下の例は、**write-attribute** 操作を使用してデプロイメントスキャナーを無効にします。タブ補完を使用して正しいリソースパスの設定を支援することにより、操作はルートノードから実行されます。

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:write-attribute(name=scan-enabled,value=false)
{"outcome" => "success"}
```

操作の結果は、**read-attribute** 操作を直接使用して確認できます。

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-attribute(name=scan-enabled)
{
    "outcome" => "success",
    "result" => false
}
```

結果は、**read-resource** 操作ですべてのノードで利用可能なリソース属性を一覧表示することによって確認することもできます。以下の例では、この特定の設定は **scan-enabled** 属性が **false** に設定されていることを示しています。

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=default:read-resource
{
    "outcome" => "success",
    "result" => {
        "auto-deploy-exploded" => false,
        "auto-deploy-xml" => true,
        "auto-deploy-zipped" => true,
        "deployment-timeout" => 600,
        "path" => "deployments",
        "relative-to" => "jboss.server.base.dir",
        "scan-enabled" => false,
        "scan-interval" => 5000
    }
}
```

結果

リソース属性が更新されます。

バグの報告

3.5.11. 管理 CLI を使用したシステムプロパティーの設定

手順3.19 管理 CLI を使用したシステムプロパティーの設定

1. JBoss EAP サーバーを起動します。
2. ご使用のオペレーティングシステム向けのコマンドを使用して、管理 CLI を起動します。

   Linux の場合

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

Windows の場合:

```
EAP_HOME\bin\jboss-cli.bat --connect
```

3. *システムプロパティーを追加します。*

   使用するコマンドは、スタンドアロンサーバーと管理対象ドメインのどちらを実行しているかによって異なります。管理対象ドメインを実行している場合は、そのドメインで実行しているすべてのサーバーへシステムプロパティーを追加できます。

   - *以下の構文を使用して、スタンドアロンサーバーにシステムプロパティーを追加します。*

     ```
     /system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
     ```

     例3.22 システムプロパティーをスタンドアロンサーバーへ追加

     ```
     [standalone@localhost:9999 /] /system-property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
     {"outcome" => "success"}
     ```

   - *以下の構文を使用して、管理対象ドメインのすべてのホストおよびサーバーにシステムプロパティーを追加します。*

     ```
     /system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
     ```

     例3.23 管理対象ドメインのすべてのサーバーへシステムプロパティーを追加

     ```
     [domain@localhost:9999 /] /system-property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
     {
         "outcome" => "success",
         "result" => undefined,
         "server-groups" => {"main-server-group" => {"host" => {"master" => {
             "server-one" => {"response" => {"outcome" => "success"}},
             "server-two" => {"response" => {"outcome" => "success"}}
         }}}}
     }
     ```

   - *以下の構文を使用して、管理対象ドメインのホストおよびそのサーバーインスタンスにシステムプロパティーを追加します。*

     ```
     /host=master/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
     ```

     例3.24 システムプロパティーをドメインのホストおよびそのサーバーへ追加

     ```
     [domain@localhost:9999 /] /host=master/system-property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
     ```

```
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
```

- 以下の構文を使用して、管理対象ドメインのサーバーインスタンスにシステムプロパティーを追加します。

```
/host=master/server-config=server-one/system-property=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

例3.25 システムプロパティーを管理対象ドメインのサーバーインスタンスへ追加

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-property=property.mybean.queue:add(value=java:/queue/MyBeanQueue)
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {"server-one" => {"response" => {"outcome" => "success"}}}}}}
}
```

4. システムプロパティーを読み取ります。

使用するコマンドは、スタンドアロンサーバーと管理対象ドメインのどちらを実行しているかによって異なります。

- 以下の構文を使用して、スタンドアロンサーバーからシステムプロパティーを読み取ります。

```
/system-property=PROPERTY_NAME:read-resource
```

例3.26 スタンドアロンサーバーからシステムプロパティーの読み取り

```
[standalone@localhost:9999 /] /system-property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {"value" => "java:/queue/MyBeanQueue"}
}
```

- 以下の構文を使用して、管理対象ドメインのすべてのホストおよびサーバーからシステムプロパティーを読み取ります。

```
/system-property=PROPERTY_NAME:read-resource
```

例3.27 管理対象ドメインのすべてのサーバーからシステムプロパティーを読み取り

```
[domain@localhost:9999 /] /system-property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}
```

- 以下の構文を使用して、管理対象ドメインのホストおよびそのサーバーインスタンスからシステムプロパティーを読み取ります。

```
/host=master/system-property=PROPERTY_NAME:read-resource
```

例3.28 ドメインのホストおよびそのサーバーからシステムプロパティーを読み取り

```
[domain@localhost:9999 /] /host=master/system-property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}
```

- 以下の構文を使用して、管理対象ドメインのサーバーインスタンスからシステムプロパティーを読み取ります。

```
/host=master/server-config=server-one/system-property=PROPERTY_NAME:read-resource
```

例3.29 管理対象ドメインのサーバーインスタンスからシステムプロパティーを読み取り

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-property=property.mybean.queue:read-resource
{
    "outcome" => "success",
    "result" => {
        "boot-time" => true,
        "value" => "java:/queue/MyBeanQueue"
    }
}
```

5. システムプロパティーを削除します。

使用するコマンドは、スタンドアロンサーバーと管理対象ドメインのどちらを実行しているかによって異なります。

- 以下の構文を使用して、スタンドアロンサーバーからシステムプロパティーを削除します。

```
/system-property=PROPERTY_NAME:remove
```

例3.30 スタンドアロンサーバーからシステムプロパティーを削除

```
[standalone@localhost:9999 /] /system-
property=property.mybean.queue:remove
{"outcome" => "success"}
```

- 以下の構文を使用して、管理対象ドメインのすべてのホストおよびサーバーからシステムプロパティーを削除します。

```
/system-property=PROPERTY_NAME:remove
```

例3.31 ドメインのホストおよびそのサーバーからシステムプロパティーを削除

```
[domain@localhost:9999 /] /system-property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
```

- 以下の構文を使用して、管理対象ドメインのホストおよびそのサーバーインスタンスからシステムプロパティーを削除します。

```
/host=master/system-property=PROPERTY_NAME:remove
```

例3.32 ドメインのホストおよびそのインスタンスからシステムプロパティーを削除

```
[domain@localhost:9999 /] /host=master/system-
property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {"outcome" => "success"}},
        "server-two" => {"response" => {"outcome" => "success"}}
    }}}}
}
```

- 以下の構文を使用して、管理対象ドメインのサーバーインスタンスからシステムプロパティーを削除します。

```
/host=master/server-config=server-one/system-property=PROPERTY_NAME:remove
```

例3.33 管理対象ドメインのサーバーからシステムプロパティーを削除

```
[domain@localhost:9999 /] /host=master/server-config=server-one/system-property=property.mybean.queue:remove
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {"server-one" => {"response" => {"outcome" => "success"}}}}}}
}
```

「パスワード」システムプロパティーのマスク

テキストパスワードが含まれるシステムプロパティー（大文字と小文字を区別しない）は、ロギングによる出力時に リダクション されたテキストに置き換えられます。これにより、ログファイルにパスワードをプレーンテキストで出力しないようにするため、セキュリティーが向上します。

バグの報告

3.5.12. 管理 CLI を使用したサーバーの新規作成

以下の例では、ホスト マスター に新しいサーバーを作成し、**clustered-server-group** に追加します。

```
[domain@localhost:9999 /] /host=master/server-config=clustered-server-1:add(group=clustered-server-group)
```

バグの報告

3.6. 管理 CLI コマンド履歴

3.6.1. 管理 CLI コマンド履歴

管理 CLI にはコマンド履歴機能があり、アプリケーションサーバーインストールではデフォルトで有効になっています。履歴は、アクティブな CLI セッションの揮発性メモリーにレコードとして保持され、**.jboss-cli-history** としてユーザーのホームディレクトリーに自動的に保存されるログファイルに追加されます。この履歴ファイルは、デフォルトで最大 500 の CLI コマンドを記録するように設定されています。

history コマンド自体は現在のセッションの履歴を返します。また、管理 CLI にはキーボードの矢印キーを使用してコマンドおよび操作の履歴を移動できる機能も含まれています。

管理 CLI の履歴機能

- 「管理 CLI コマンド履歴の表示」

- 「管理 CLI コマンド履歴の消去」
- 「管理 CLI コマンド履歴の無効化」
- 「管理 CLI コマンド履歴の有効化」

バグの報告

3.6.2. 管理 CLI コマンド履歴の表示

前提条件

- 「管理 CLI の起動」

手順3.20 管理 CLI コマンド履歴の表示

- **history** コマンドを実行します。
  管理 CLI で 履歴 コマンドを入力します。

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

結果

CLI の起動後または履歴削除コマンドの実行後にメモリーに格納された CLI コマンドの履歴が表示されます。

バグの報告

3.6.3. 管理 CLI コマンド履歴の消去

前提条件

- 「管理 CLI の起動」

手順3.21 管理 CLI コマンド履歴の消去

- **history --clear** コマンドを実行します。
  管理 CLI で 履歴 **--clear** コマンドを入力します。

```
[standalone@localhost:9999 /] history --clear
```

結果

CLI の起動後に記録されたコマンドの履歴がセッションメモリーから削除されます。コマンド履歴は、ユーザーのホームディレクトリーに保存されている **.jboss-cli-history** ファイルに残ります。

バグの報告

3.6.4. 管理 CLI コマンド履歴の無効化

前提条件

- 「管理 CLI の起動」

手順3.22 管理 CLI コマンド履歴の無効化

- ***history --disable*** *コマンドを実行します。*
  管理 CLI で ***history --disable*** *コマンドを入力します。*

```
[standalone@localhost:9999 /] history --disable
```

結果

***CLI** で実行されたコマンドは、メモリー内またはユーザーのホームディレクトリーに保存された **.jboss-cli-history** ファイルに記録されません。*

バグの報告

3.6.5. 管理 CLI コマンド履歴の有効化

前提条件

- *「管理 CLI の起動」*

手順3.23 管理 CLI コマンド履歴の有効化

- ***history --enable*** *コマンドを実行します。*
  管理 CLI で ***history --enable*** *コマンドを入力します。*

```
[standalone@localhost:9999 /] history --enable
```

結果

***CLI** で実行されたコマンドは、メモリーと、ユーザーのホームディレクトリーに保存された **.jboss-cli-history** ファイルに記録されます。*

バグの報告

3.7. 管理インターフェース監査ロギング

3.7.1. 管理インターフェース監査ロギング

監査ロギングを有効にすると、管理コンソール、管理 CLI インターフェース、またはカスタム作成の管理アプリケーションを使用して実行されるすべての操作は監査ロギングの対象となります。

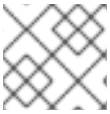
監査ログエントリーは JSON 形式で保存され、設定に基づいて、syslog サーバーまたは両方に送信するファイルに格納できます。監査ロギングは管理 CLI を使用してのみ設定でき、デフォルトでは無効になっています。

EAP には 'authenticated session' がないため、ログインおよびログアウトイベントは監査できません。代わりに、ユーザーから操作を受信すると監査メッセージがログに記録されます。

注記

デフォルトでは、監査ロギングはアクティブではありません。監査ロギングは管理 CLI を使用してのみ設定できます。

利用可能な管理インターフェース監査ロギング設定オプションとその現在の値を一覧表示するには、以下の管理 CLI コマンドを入力します。



注記

管理対象ドメインのコマンドに、プレフィックス **/host=HOST_NAME** を追加します。

```
[... /] /core-service=management/access=audit:read-resource(recursive=true)
```

バグの報告

3.7.2. ファイルへの管理インターフェース監査ロギングの有効化

ファイルへの監査ロギング出力を有効にするには、以下の管理 CLI コマンドを入力します。

注記

管理対象ドメインに変更を適用する場合は、以下のコマンドにプレフィックス **/host=HOST_NAME** を追加します。

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)
```

管理操作がファイルに記録されるようになりました。

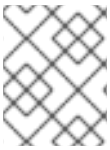
- スタンドアロンモード： **EAP_HOME/standalone/data/audit-log.log**
- ドメインモード： **EAP_HOME/domain/data/audit-log.log**

すべてのファイルハンドラー属性の詳細は、「管理インターフェース監査ロギングリファレンス」 を参照してください。

バグの報告

3.7.3. syslog サーバーへの管理インターフェース監査ロギングの有効化

デフォルトでは、監査ロギングが有効である場合にファイルへ出力するよう事前設定されています。この手順では、syslog サーバーへ出力を設定し、ファイルへの監査ロギングを有効にします。すべての Syslog ハンドラー属性の詳細は、「管理インターフェース監査ロギングリファレンス」 を参照してください。



注記

管理対象ドメインに変更を適用する場合は、プレフィックス **/host=HOST_NAME** を **/core-service** コマンドに追加します。

手順3.24 Syslog サーバーへの監査ロギングの有効化

1. 監査ロギングの有効化
   以下のコマンドを実行します。

```
[.. /]/core-service=management/access=audit/logger=audit-log:write-
attribute(name=enabled,value=true)
```

2. **syslog** ハンドラーの作成
   この例では、**syslog** サーバーが JBoss EAP インスタンスと同じサーバーで、ポート 514 で実行します。ホスト 属性の値を、実際の環境に適した値に置き換えます。

   例3.34 syslog ハンドラーの例

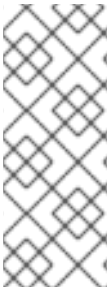
   ```
   [.. /]batch
   [.. / #]/core-service=management/access=audit/syslog-handler=mysyslog:add(formatter=json-formatter)
   [.. / #]/core-service=management/access=audit/syslog-handler=mysyslog/protocol=udp:add(host=localhost,port=514)
   [.. /]run-batch
   ```

3. **syslog** ハンドラーへの参照の追加
   以下のコマンドを実行します。

   ```
   [.. /]/core-service=management/access=audit/logger=audit-log/handler=mysyslog:add
   ```

結果

管理インターフェースの監査ログエントリーが **syslog** サーバーに記録されます。



注記

オペレーティングシステムでロギングが有効になっていない限り、JBoss EAP の Syslog サーバーへの監査ロギングを有効にしても動作しません。

Red Hat Enterprise Linux の **rsyslog** 設定に関する詳細は、Red Hat Enterprise Linux の『システム管理者 のガイド』の「**rsyslog** の 基本設定」 セクションを参照してください。 https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/

バグの報告

3.7.4. 管理インターフェース監査ロギングの無効化

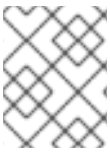
ファイルまたは **syslog** サーバーへの監査ロギングは、以下のコマンドを実行して無効にできます。

```
/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=false)
```

バグの報告

3.7.5. 管理インターフェース監査ログの読み取り

ファイルに出力される監査ログエントリーは、テキスト**ビューアー** で参照するのが最適ですが、**syslog** サーバーへ出力される監査ログエントリーは **syslog** ビューアーアプリケーションを使用して参照するのが最適です。



注記

ログファイルの参照にテキスト**エディター** を使用することは推奨されません。 これは、追加のログエントリーがログファイルに書き込まれなくなることがあるためです。

*管理インターフェース監査ログは JSON 形式で出力されます。各ログエントリーは任意のタイムスタンプで始まり、『**管理インターフェースの監査ログフィールド の表に記載されている**』 フィールドです。*

表3.5 管理インターフェース監査ログフィールド

フィールド名	説明
type	管理操作であることを示す **core** または **jmx** は JMX サブシステムからの意味を持つことができます（JMX サブシステムの監査ロギングの設定については JMX サブシステムを参照してください）。
r/o	操作によって管理モデルが変更されない場合は、値が **true** になります。そうでない場合は **false** になります。
起動	起動プロセス中に操作が実行された場合は、値が **true** になります。サーバーの起動後に操作が実行された場合は **false** になります。
version	JBoss EAP インスタンスのバージョン番号。
user	認証されたユーザーのユーザー名。実行中のサーバーと同じマシンの管理 CLI で操作を行う場合は、特別な **$local** ユーザーが使用されます。
domainUUID	ドメインコントローラーからサーバー、スレーブホストコントローラー、およびスレーブホストコントローラーサーバーへ伝播されるすべての操作をリンクする ID。
access	以下のいずれかの値を使用できます。 ● NATIVE - 操作が管理 CLI などのネイティブ管理インターフェースから実行されます。 ● HTTP - 操作が管理コンソールなどのドメイン HTTP インターフェースから実行されます。 ● JMX - 操作が JMX サブシステムから実行されます。JMX の監査ロギングの設定方法は JMX を参照してください。
remote-address	この操作を実行するクライアントのアドレス。
success	操作に成功した場合は値が **true** になります。ロールバックされた場合は **false** になります。
ops	実行される操作。JSON へシリアライズ化された操作のリストになります。起動時は、XML の解析で生じた演算です。通常、一覧を起動するとエントリーが 1 つ含まれます。

バグの報告

第4章 ユーザー管理

4.1. JBOSS EAP ユーザー管理

JBoss EAP 6 のインストール方法によっては、管理インターフェースにアクセスできるユーザーアカウントが初期設定できない場合があります。

グラフィカルインストーラーを使用してプラットフォームをインストールする場合は、インストール時に JBoss EAP 6 管理インターフェースにアクセスするために必要な権限を持つユーザーアカウント 1 つのみが作成されます。

ZIP アーカイブを使用してプラットフォームを手動でインストールした場合には、インストール時に適切な権限を持つユーザーアカウントが作成されません。

コンソールで JAR インストーラーを使用してプラットフォームを手動でインストールする場合は、インストール時に適切な権限を持つユーザーアカウントを 1 つ作成します。

トラフィックの送信元がローカルホストであっても、JBoss EAP 6 との HTTP ベースの通信はリモートアクセスと見なされます。したがって、管理コンソールを使用するには、少なくとも 1 人の管理ユーザーを作成する必要があります。ユーザーを追加する前に管理コンソールにアクセスしようとすると、ユーザーが追加されるまでデプロイされないのでエラーが発生します。

本ガイドでは、**add-user.sh** スクリプトを使用して JBoss EAP 6 の簡単なユーザー管理について説明します。LDAP やロールベースアクセス制御(RBAC)などの高度な認証および承認オプションは、JBoss EAP 『『**セキュリティーアーキテクチャー**』』の「『**コア管理認証**』」を参照してください。

バグの報告

4.2. ユーザーの作成

4.2.1. 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でユーザーが作成されていない場合に、最初の管理ユーザーを作成する方法を説明します。この初期管理ユーザーは、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4.1 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/ ディレクトリーに移動します。オペレーティングシステムに適したスクリプトを実行します。

   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、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 ベースの管理コンソールまたはコマンドラインベースの管理 CLI を使用して管理操作を実行することが承認されます。もう 1 つの選択肢は b で、ユーザーを ApplicationRealm に追加し、特定のパーミッションは提供されません。そ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。*
   プロンプトが表示されたら、ユーザー名とパスワードを入力します。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、コンマ区切りリストを入力します。ユーザーがグループに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求められます。問題がなければ、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、ManagementRealm の JBoss EAP 6 の別のインスタンスを表すユーザーである で JBoss EAP 6 に追加する必要がある他のタイプのユーザーは、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したユーザーを指定できます。yes を選択すると、ユーザーのパスワードを表すハッシュ化された secret 値が表示されます。これは別の設定ファイルに追加する必要があります。このタスクの目的上、この質問には no と回答します。

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。実行中のシステムのいつでも追加することもできます。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、承認を微調整できます。

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は抑制できます。これは、最小限のパラメーターの username および password が指定されている場合にのみ該当します。エラーメッセージが表示されます。*

結果

追加したユーザーは、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。

バグの報告

4.2.2. ユーザー管理の add-user スクリプトへ引数を渡す

add-user.sh または add-user.bat コマンドを対話的に実行することも、コマンドラインで引数を渡すこともできます。本セクションでは、コマンドライン引数を add-user スクリプトに渡す際に利用可能なオプションを説明します。

add-user.sh または add-user.bat コマンドで使用できるコマンドライン引数の一覧は、「add-user コマンド引数」 を参照してください。

代替のプロパティーファイルおよび場所を指定する方法は、「ユーザー管理情報の代替プロパティーファイルの指定」 を参照してください。

add-user.sh または add-user.bat コマンドで引数を渡す方法を実証する例は、「デフォルトのプロパティーファイルを使用した単一のグループに属するユーザーの作成」、「デフォルトのプロパティーファイルを使用した複数のグループへのユーザーの作成」、「デフォルトのプロパティーファイルを使用したデフォルトのレルムの管理者権限でのユーザーの作成」、および「代替プロパティーファイルを使用した情報の保存における単一グループへのユーザーの作成」 を参照してください。.

バグの報告

4.2.3. add-user コマンド引数

以下の表は、add-user.sh または add-user.bat コマンドで使用できる引数を示しています。

表4.1 add-user コマンド引数

コマンドライン引数	引数の値	説明
-a	該当なし	この引数は、アプリケーションレルムでユーザーを作成するように指定します。省略した場合、デフォルトでは管理レルムでユーザーが作成されます。
-dc	*DOMAIN_CONFIGURATION_DIRECTORY*	この引数は、プロパティーファイルが含まれるドメイン設定ディレクトリーを指定します。省略した場合、デフォルトのディレクトリーは ***EAP_HOME*/domain/configuration/** になります。
-sc	*SERVER_CONFIGURATION_DIRECTORY*	この引数は、プロパティーファイルが含まれる代替のスタンドアロンサーバー設定ディレクトリーを指定します。省略した場合、デフォルトのディレクトリーは ***EAP_HOME*/standalone/configuration/** になります。
-up --user-properties	*USER_PROPERTIES_FILE*	この引数は、代替のユーザープロパティーファイルの名前を指定します。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。
-g --group	*GROUP_LIST*	このユーザーに割り当てるグループのコンマ区切りリスト。

コマンドライン引数	引数の値	説明
-gp --group-properties	*GROUP_PROPERTIES_FILE*	この引数は、代替のグループプロパティーファイルの名前を指定します。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。
-p --password	*PASSWORD*	ユーザーのパスワード。パスワードは以下の要件を満たす必要があります。 ● 8 文字以上でなければなりません。 ● 1 文字以上のアルファベットが含まれなくてはなりません。 ● 1 文字以上の数字が含まれなければなりません。 ● 英数字以外の記号が 1 つ以上含まれなければなりません。
-u --user	*USER_NAME*	ユーザーの名前。英数字と以下のシンボルのみを使用できます：,./=@\.
-r --realm	*REALM_NAME*	管理インターフェースをセキュアにするために使用されるレルムの名前。省略した場合、デフォルト値は **ManagementRealm** です。
-s --silent	該当なし	コンソールへ出力せずに add-user スクリプトを実行します。
-h --help	該当なし	add-user スクリプトの使用情報を表示します。

バグの報告

4.2.4. ユーザー管理情報の代替プロパティーファイルの指定

概要

*デフォルトでは、**add-user.sh** または **add-user.bat** スクリプトを使用して作成されたユーザーおよびロール情報は、サーバー設定ディレクトリーにあるプロパティーファイルに保存されます。サーバー設定情報は **EAP_HOME/standalone/configuration/** ディレクトリーに保存され、ドメイン設定情報は **EAP_HOME/domain/configuration/** ディレクトリーに保存されます。このトピックでは、デフォルトのファイル名および場所を上書きする方法について説明します。*

代替プロパティーファイルの指定

- *サーバー設定の代替ディレクトリーを指定するには、**-sc** 引数を使用します。この引数は、サーバー設定プロパティーファイルが含まれる代替のディレクトリーを指定します。*

- *ドメイン設定の代替ディレクトリーを指定するには、**-dc** 引数を使用します。この引数は、ドメイン設定プロパティーファイルを含む代替ディレクトリーを指定します。*
- *代替のユーザー設定プロパティーファイルを指定するには、**-up** 引数または **--user-properties** 引数を使用します。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。*
- *代替のグループ設定プロパティーファイルを指定するには、**-gp** 引数または **--group-properties** 引数を使用します。絶対パスを使用でき、代替の設定ディレクトリーを指定する **-sc** または **-dc** 引数と共に使用されるファイル名を使用することもできます。*

注記

***add-user** コマンドは、既存のプロパティーファイルでの操作を行うことを目的としています。コマンドライン引数に指定された代替のプロパティーファイルが存在しない場合は、以下のエラーが表示されます。*

> ***JBAS015234: No appusers.properties files found***

コマンド引数の詳細は、「add-user コマンド引数」を参照してください。

バグの報告

4.3. ADD-USER スクリプトのコマンドラインの例

4.3.1. デフォルトのプロパティーファイルを使用した単一のグループに属するユーザーの作成

例4.1 単一グループに属するユーザーの作成

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g 'guest'
```

この例では、以下の結果を表示しています。

- *ユーザー **appuser1** は、ユーザー情報が保存される以下のデフォルトプロパティーファイルに追加されます。*

  EAP_HOME/standalone/configuration/application-users.properties

  EAP_HOME/domain/configuration/application-users.properties

- *グループ **guest** を持つユーザー **appuser1** が、グループ情報を格納するデフォルトのプロパティーファイルに追加されます。*

  EAP_HOME/standalone/configuration/application-roles.properties

  EAP_HOME/domain/configuration/application-roles.properties

バグの報告

4.3.2. デフォルトのプロパティーファイルを使用した複数のグループへのユーザーの作成

例4.2 複数のグループに属するユーザーの作成

```
EAP_HOME/bin/add-user.sh -a -u 'appuser1' -p 'password1!' -g 'guest,app1group,app2group'
```

この例では、以下の結果を表示しています。

- ユーザー**appuser1** は、ユーザー情報が保存される以下のデフォルトプロパティーファイルに追加されます。

  EAP_HOME/standalone/configuration/application-users.properties

  EAP_HOME/domain/configuration/application-users.properties

- グループ**guest**、**app1group**、および**app2group** を持つユーザー**appuser1** が、グループ情報を格納するデフォルトのプロパティーファイルに追加されます。

  EAP_HOME/standalone/configuration/application-roles.properties

  EAP_HOME/domain/configuration/application-roles.properties

バグの報告

4.3.3. デフォルトのプロパティーファイルを使用したデフォルトのレルムの管理者権限でのユーザーの作成

例4.3 デフォルトレルムで管理者権限を持つユーザーの作成

```
EAP_HOME/bin/add-user.sh -u 'adminuser1' -p 'password1!' -g 'admin'
```

この例では、以下の結果を表示しています。

- ユーザー**adminuser1** は、ユーザー情報が保存される以下のデフォルトプロパティーファイルに追加されます。

  EAP_HOME/standalone/configuration/mgmt-users.properties

  EAP_HOME/domain/configuration/mgmt-users.properties

- グループ**admin** のユーザー**adminuser1** は、グループ情報を格納するデフォルトのプロパティーファイルに追加されます。

  EAP_HOME/standalone/configuration/mgmt-groups.properties

  EAP_HOME/domain/configuration/mgmt-groups.properties

バグの報告

4.3.4. 代替プロパティーファイルを使用した情報の保存における単一グループへのユーザーの作成

例4.4 代替のプロパティーファイルを使用した単一グループに属するユーザーの作成

```
EAP_HOME/bin/add-user.sh -a -u appuser1 -p password1! -g app1group -sc
/home/someusername/userconfigs/ -up appusers.properties -gp appgroups.properties
```

この例では、以下の結果を表示しています。

- ユーザー**appuser1** は以下のプロパティーファイルに追加され、このファイルがユーザー情報を保存するデフォルトファイルになります。

  /home/someusername/userconfigs/appusers.properties

- **app1group** グループを持つユーザー**appuser1** が以下のプロパティーファイルに追加され、このファイルがグループ情報を保存するデフォルトファイルになりました。

  /home/someusername/userconfigs/appgroups.properties

バグの報告

第5章 ネットワークおよびポート設定

5.1. インターフェース

5.1.1. インターフェース

JBoss EAP は設定全体で名前付きインターフェース参照を使用します。これにより、使用ごとにインターフェースの完全な詳細を必要とせずに、論理名で個々のインターフェース宣言を参照する機能が提供されます。

論理名を使用すると、名前付きインターフェースへのグループ参照の整合性も可能にします。この場合、管理対象ドメインのサーバーインスタンスには複数のマシン間でさまざまなインターフェースの詳細が含まれる可能性があります。論理名では、各サーバーインスタンスが論理名グループに対応しているため、インターフェースグループの管理が容易になります。

ネットワークインターフェースは、物理インターフェースの論理名と選択基準を指定して宣言されます。

JBoss EAP のデフォルト設定には、管理 インターフェースとパブリック インターフェース名の両方が含まれます。管理 インターフェース名は、管理レイヤーを必要とするすべてのコンポーネントおよびサービス（HTTP 管理エンドポイントを含む）に使用できます。**public** インターフェース名は、Web や Messaging などのアプリケーション関連のネットワーク通信すべてに使用できます。

デフォルト名の使用は必須ではありません。新しい論理名を作成し、デフォルト名に置き換えることができます。

domain.xml、**host.xml**、および **standalone.xml** 設定ファイルには、インターフェース宣言が含まれます。宣言基準はワイルドカードアドレスを参照したり、有効な一致となるためにインターフェースまたはアドレスで必要となる 1 つ以上の特性のセットを指定したりできます。

3 つの設定ファイルは直接編集可能ですが、手動編集は不要になりました。管理 CLI および管理コンソールは、設定の変更にセキュアで制御された永続的な環境を提供します。

以下の例は、通常は **standalone.xml** または **host.xml** 設定ファイルのいずれかで定義されるインターフェース宣言の複数の設定を示しています。これらのファイルを使用すると、リモートホストグループが特定のインターフェース属性を維持でき、引き続きドメインコントローラーインターフェースへの参照が許可されます。

以下の例は、管理およびパブリックの相対名グループに指定された特定の **inet-address** 値を示しています。

例5.1 **inet-address** 値で作成されたインターフェースグループ

```
<interfaces>
  <interface name="management">
   <inet-address value="127.0.0.1"/>
  </interface>
  <interface name="public">
   <inet-address value="127.0.0.1"/>
  </interface>
</interfaces>
```

以下の例は、グローバルインターフェースグループを示しています。**any-address** 要素を使用してワイルドカードアドレスを宣言します。

例5.2 ワイルドカード宣言で作成されたグローバルグループ

```
<interface name="global">
  <!-- Use the wild-card address -->
  <any-address/>
</interface>
```

以下の例では、**external** という相対グループの下でネットワークインターフェースカード(eth0)を宣言します。

例5.3 NIC 値で作成された外部グループ

```
<interface name="external">
  <nic name="eth0"/>
</interface>
```

以下の例では、要件のあるデフォルトグループを宣言します。これらの要件は、インターフェースの条件を有効な一致に設定します。これは、JBoss EAP がインターフェースの名前を使用して参照できる特定のプロパティーを持つインターフェース宣言グループの作成を許可する方法の例になります。これは、複数のサーバーインスタンスにおける設定の複雑さおよび管理オーバーヘッドを軽減するのに役立ちます。

例5.4 特定の条件値で作成されたデフォルトグループ

```
<interface name="default">
  <!-- Match any interface/address on the right subnet if it's
       up, supports multicast, and isn't point-to-point -->
  <subnet-match value="192.168.0.0/16"/>
  <up/>
  <multicast/>
  <not>
    <point-to-point/>
  </not>
</interface>
```

バグの報告

5.1.2. インターフェースの設定

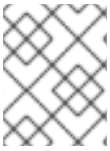
standalone.xml および **host.xml** 設定ファイルのデフォルトのインターフェース設定は、それぞれ相対的なインターフェーストークンを持つ 3 つの名前付きインターフェースを提供します。以下の表にあるように、管理コンソールまたは管理 CLI を使用して追加の属性および値を設定します。必要に応じて、相対インターフェースバインディングは特定の値に置き換えることができますが、これを行う場合は、**-b** スイッチが相対値しか上書きできないため、サーバー実行時にインターフェース値を渡すことができないことに注意してください。

例5.5 デフォルトインターフェース設定

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

管理対象ドメインで複数のサーバーを実行している間、インターフェースバインディングはそれぞれの ***host.xml*** *ファイルの個別のサーバーに割り当てることができます。以下に例を示します。*

```
<servers>
  <server name="server-name" group="main-server-group">
    <interfaces>
      <interface name="public">
        <inet-address value="ip-address"/>
      </interface>
    </interfaces>
  </server>
</servers>
```



注記

上記の例では、server-name を実際のサーバー名に置き換え、ip-address を実際の IP アドレスに置き換えます。

表5.1 インターフェース属性と値

インターフェース要素	説明
any	インターフェースの選択基準の一部は、最低でも基準のネストされたセットの 1 つ (すべてとは限らない) を満たす必要があることを示す要素。
any-address	このインターフェースを使用するソケットをワイルドカードアドレスにバインドする必要があることを示す空の要素。 java.net.preferIpV4Stack システムプロパティーが true に設定されていない限り、IPv6 ワイルドカードアドレス(::)が使用されます。true に設定すると、IPv4 ワイルドカードアドレス(0.0.0.0)が使用されます。 ソケットがデュアルスタックマシンの IPv6 anylocal アドレスにバインドされた場合は、IPv6 および IPv4 トラフィックを受け入れることができます。 IPv4 (IPv4 マッピング) anylocal アドレスにバインドされた場合は、IPv4 トラフィックのみを受け入れることができます。
any-ipv4-address	このインターフェースを使用するソケットを IPv4 ワイルドカードアドレス (0.0.0.0) にバインドする必要があることを示す空の要素。

インターフェース要素	説明
any-ipv6-address	このインターフェースを使用するソケットを IPv6 ワイルドカードアドレス (::) にバインドする必要があることを示す空の要素。
inet-address	IPv6 または IPv4 のドット区切り表記の IP アドレス、または IP アドレスに解決できるホスト名。
link-local-address	インターフェースの選択基準の一部として、関連付けられたアドレスがリンクローカルであるかどうかを示す空の要素。
loopback	インターフェースの選択基準の一部として、ループバックインターフェースであるかどうかを示す空の要素。
loopback-address	マシンのループバックインターフェースで実際には設定できないループバックアドレス。IP アドレスが関連付けられた NIC が見つからない場合であっても該当する値が使用されるため、inet-address タイプとは異なります。
multicast	インターフェースの選択基準の一部として、マルチキャストをサポートするかどうかを示す空の要素。
nic	ネットワークインターフェースの名前 (eth0、eth1、lo など)。
nic-match	使用できるインターフェースを見つけるために、マシンで利用可能なネットワークインターフェースの名前を検索する正規表現。
not	インターフェースの選択基準の一部は、基準のネストされたセットを満たしてはならないことを示す要素。
point-to-point	インターフェースの選択基準の一部として、ポイントツーポイントインターフェースであるかどうかを示す空の要素。
public-address	インターフェースの選択基準の一部として、公開されたルーティング可能なアドレスを持つかどうかを示す空の要素。
site-local-address	インターフェースの選択基準の一部として、関連付けられたアドレスがサイトローカルであるかどうかを示す空の要素。
subnet-match	「スラッシュ表記法」で記述されたネットワーク IP アドレスとアドレスのネットワーク接頭辞のビット数（例："192.168.0.0/16".
up	インターフェースの選択基準の一部として、現在稼動しているかどうかを示す空の要素。
virtual	インターフェースの選択基準の一部として、仮想インターフェースであるかどうかを示す空の要素。

- *インターフェース属性の設定*

- 管理CLI でのインターフェース属性の設定
  タブ補完を使用して、入力しながらコマンド文字列を補完したり、利用可能な属性を表示したりできます。

  管理CLI を使用して、新しいサーバーを追加してインスタンスをそのサーバーに設定し、同じ設定をXML に追加します。server-name を実際のサーバー名に置き換え、ip-address を実際のIP アドレスに置き換えます。

  ```
  /host=master/server-config=server-name:add(group=main-server-group)
  /host=master/server-config=server-name/interface=public:add(inet-address=ip-address)
  ```

  管理CLI を使用して、新しいインターフェースを追加し、インターフェース属性に新しい値を書き込みます。

  a. 新しいインターフェースの追加
     add 操作は必要に応じて新しいインターフェースを作成します。**add** コマンドは、管理CLI セッションのルートから実行され、以下の例では interfacename というタイトルの新しいインターフェース名を作成し、**inet-address** が12.0.0.2 として宣言されます。

     ```
     /interface=interfacename/:add(inet-address=12.0.0.2)
     ```

  b. インターフェース属性の編集
     write-attribute 操作は、新しい値を属性に書き込みます。以下の例では、**inet-address** の値を12.0.0.8 に更新します。

     ```
     /interface=interfacename/:write-attribute(name=inet-address, value=12.0.0.8)
     ```

  c. インターフェース属性の検証
     include-runtime=true パラメーターを指定して**read-resource** 操作を実行し、サーバーモデルでアクティブな現在の値をすべて公開して、属性値が変更されたことを確認します。以下に例を示します。

     ```
     [standalone@localhost:9999 interface=public] :read-resource(include-runtime=true)
     ```

- 管理コンソールでのインターフェース属性の設定

  a. 管理コンソールへのログイン
     管理対象ドメインまたはスタンドアロンサーバーインスタンスの管理コンソールにログインします。

  b. Configuration タブに移動します。
     画面上部の**Configuration** タブを選択します。

     注記

     ドメインモードの場合は、画面左上の**Profile** ドロップダウンメニューからプロファイルを選択します。

  c. ナビゲーションメニュー から インターフェース を選択します。
     ナビゲーションメニューから**Interfaces** メニュー項目を選択します。

d. 新しいインターフェースの追加

i. **Add** をクリックします。

ii. **Name**、**Inet Address**、および **Address Wildcard** に、必要な値を入力します。

iii. **Save** をクリックします。

e. インターフェース属性の編集

i. 利用可能なインターフェース の一覧から編集する必要があるインターフェースを選択し、編集 をクリックします。

ii. **Name**、**Inet Address**、および **Address Wildcard** に、必要な値を入力します。

iii. **Save** をクリックします。

バグの報告

5.2. ソケットバインディンググループ

5.2.1. ソケットバインディンググループ

ソケットバインディングとソケットバインディンググループを使用することにより、ネットワークポートと、JBoss EAP 6 の設定で必要なネットワーキングインターフェースとの関係を定義できます。

ソケットバインディングはソケットの名前付き設定です。これらの名前付き設定の宣言は **domain.xml** と **standalone.xml** 設定ファイルの両方にあります。設定の他のセクションは、ソケット設定の完全な詳細を含めるのではなく、論理名でこれらのソケットを参照できます。これにより、マシンごとに異なる可能性のある相対ソケット設定を参照できます。

ソケットバインディングはソケットバインディンググループで収集されます。ソケットバインディンググループは、ある論理名でグループ化されたソケットバインディング宣言のコレクションです。named グループは、設定全体で参照できます。スタンドアロンサーバーにはこのようなグループが 1 つだけ含まれますが、管理対象ドメインインスタンスには複数のグループを含めることができます。管理対象ドメインで各サーバーグループのソケットバインディンググループを作成するか、複数のサーバーグループ間でソケットバインディンググループを共有することができます。

命名グループを使用すると、管理対象ドメインの場合にサーバーグループを設定するときにソケットバインディングの特定のグループに、簡素化された参照を使用できます。もう 1 つの一般的な使用方法は、1 つのシステム上のスタンドアロンサーバーの複数のインスタンスの設定および管理です。以下の例は、スタンドアロンおよびドメインインスタンスの設定ファイルのデフォルトのソケットバインディンググループを示しています。

> 例5.6 スタンドアロン設定のデフォルトソケットバインディング
>
> **standalone.xml** 設定ファイルのデフォルトのソケットバインディンググループは、**standard-sockets** でグループ化されます。このグループは、同じ論理参照方法を使用してパブリック インターフェースでも参照されます。
>
> ```
> <socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
>     <socket-binding name="management-native" interface="management" port="${jboss.management.native.port:9999}"/>
>     <socket-binding name="management-http" interface="management" port="${jboss
> ```

```
.management.http.port:9990}"/>
        <socket-binding name="management-https" interface="management" port="${jboss
.management.https.port:9443}"/>
        <socket-binding name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="remoting" port="4447"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

例5.7 ドメイン設定のデフォルトソケットバインディング

domain.xml *設定ファイルのデフォルトのソケットバインディンググループには、***standard-sockets***、***ha-sockets***、***full-sockets***、***full-ha-sockets*** *グループの 4 つのグループが含まれます。これらのグループは、***public*** という名前のインターフェースでも参照されます。*

```
    <socket-binding-groups>
        <socket-binding-group name="standard-sockets" default-interface="public">
            <!-- Needed for server groups using the 'default' profile  -->
            <socket-binding name="ajp" port="8009"/>
            <socket-binding name="http" port="8080"/>
            <socket-binding name="https" port="8443"/>
            <socket-binding name="remoting" port="4447"/>
            <socket-binding name="txn-recovery-environment" port="4712"/>
            <socket-binding name="txn-status-manager" port="4713"/>
            <outbound-socket-binding name="mail-smtp">
                <remote-destination host="localhost" port="25"/>
            </outbound-socket-binding>
        </socket-binding-group>
        <socket-binding-group name="ha-sockets" default-interface="public">
            <!-- Needed for server groups using the 'ha' profile  -->
            <socket-binding name="ajp" port="8009"/>
            <socket-binding name="http" port="8080"/>
            <socket-binding name="https" port="8443"/>
            <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
            <socket-binding name="jgroups-tcp" port="7600"/>
            <socket-binding name="jgroups-tcp-fd" port="57600"/>
            <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
            <socket-binding name="jgroups-udp-fd" port="54200"/>
            <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105"
multicast-port="23364"/>
            <socket-binding name="remoting" port="4447"/>
            <socket-binding name="txn-recovery-environment" port="4712"/>
            <socket-binding name="txn-status-manager" port="4713"/>
            <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
            </outbound-socket-binding>
        </socket-binding-group>
```

```
        <socket-binding-group name="full-sockets" default-interface="public">
            <!-- Needed for server groups using the 'full' profile  -->
            <socket-binding name="ajp" port="8009"/>
            <socket-binding name="http" port="8080"/>
            <socket-binding name="https" port="8443"/>
            <socket-binding name="jacorb" interface="unsecure" port="3528"/>
            <socket-binding name="jacorb-ssl" interface="unsecure" port="3529"/>
            <socket-binding name="messaging" port="5445"/>
            <socket-binding name="messaging-group" port="0" multicast-address="${jboss
.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
            <socket-binding name="messaging-throughput" port="5455"/>
            <socket-binding name="remoting" port="4447"/>
            <socket-binding name="txn-recovery-environment" port="4712"/>
            <socket-binding name="txn-status-manager" port="4713"/>
            <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
            </outbound-socket-binding>
        </socket-binding-group>
        <socket-binding-group name="full-ha-sockets" default-interface="public">
            <!-- Needed for server groups using the 'full-ha' profile  -->
            <socket-binding name="ajp" port="8009"/>
            <socket-binding name="http" port="8080"/>
            <socket-binding name="https" port="8443"/>
            <socket-binding name="jacorb" interface="unsecure" port="3528"/>
            <socket-binding name="jacorb-ssl" interface="unsecure" port="3529"/>
            <socket-binding name="jgroups-mping" port="0" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45700"/>
            <socket-binding name="jgroups-tcp" port="7600"/>
            <socket-binding name="jgroups-tcp-fd" port="57600"/>
            <socket-binding name="jgroups-udp" port="55200" multicast-address="${jboss
.default.multicast.address:230.0.0.4}" multicast-port="45688"/>
            <socket-binding name="jgroups-udp-fd" port="54200"/>
            <socket-binding name="messaging" port="5445"/>
            <socket-binding name="messaging-group" port="0" multicast-address="${jboss
.messaging.group.address:231.7.7.7}" multicast-
port="${jboss.messaging.group.port:9876}"/>
            <socket-binding name="messaging-throughput" port="5455"/>
            <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105"
multicast-port="23364"/>
            <socket-binding name="remoting" port="4447"/>
            <socket-binding name="txn-recovery-environment" port="4712"/>
            <socket-binding name="txn-status-manager" port="4713"/>
            <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
            </outbound-socket-binding>
        </socket-binding-group>
    </socket-binding-groups>
```

ソケットバインディングインスタンスは、アプリケーションサーバーディレクトリーの ***standalone.xml*** *および* ***domain.xml*** *ソースファイルで作成および編集できます。バインディングの管理方法として、管理コンソールまたは管理 CLI の使用が推奨されます。管理コンソールを使用する利点として、* ***General Configuration*** *セクションの下に専用の Socket Binding Group 画面を備えたグラフィカルユーザーインターフェースがあります。管理 CLI は、アプリケーションサーバー設定の高いレベル*

で、バッチ処理とスクリプトの使用を可能にするコマンドラインアプローチに基づいてAPIおよびワークフローを提供します。両方のインターフェースにより、変更を永続化するか、サーバー設定に保存できます。

バグの報告

5.2.2. ソケットバインディングの設定

ソケットバインディングは固有のソケットバインディンググループで定義できます。スタンドアロンサーバーにはこのようなグループである ***standard-sockets*** *グループが含まれ、さらにグループを作成することができません。代わりに、代替のスタンドアロンサーバー設定ファイルを作成できます。ただし、管理対象ドメインでは、複数のソケットバインディンググループを作成し、必要に応じて含まれるソケットバインディングを設定できます。以下の表は、各ソケットバインディングで使用できる属性を示しています。*

表5.2 ソケットバインディング属性

属性	説明	ロール
name	設定の他の場所で使用する必要があるソケット設定の論理名。	必須
port	この設定に基づいたソケットをバインドする必要があるベースポート。すべてのポート値にインクリメントまたはデクリメントを適用して、サーバーがこのベース値をオーバーライドするように設定できます。	必須
interface	この設定に基づいたソケットをバインドする必要があるインターフェースの論理名。定義されないと、エンクロージングソケットバインディンググループからの **default-interface** の値が使用されます。	任意
multicast-address	マルチキャストにソケットが使用される場合は、使用するマルチキャストアドレス。	任意
multicast-port	マルチキャストにソケットが使用される場合は、使用するマルチキャストポート。	任意
fixed-port	**true** の場合、**port** の値を常にソケットに使用する必要があり、インクリメントまたはデクリメントを適用して上書きしないでください。	任意

- *ソケットバインディンググループのソケットバインディングの設定*

管理 CLI または管理コンソールを選択して、必要に応じてソケットバインディングを設定します。

- *管理 CLI を使用したソケットバインディングの設定*
  管理 CLI を使用してソケットバインディングを設定します。

  a. *新しいソケットバインディングの追加*
  *必要に応じて **add** 操作を使用して新規アドレス設定を作成します。管理 CLI セッションのルートからこのコマンドを実行できます。以下の例では、**newsocket** という新しいソケットバインディングが作成され、**port** 属性は1234 として宣言されています。この例は、以下に示すように、スタンドアロンサーバーと、**standard-sockets** ソケットバインディンググループで編集される管理対象ドメインの両方に適用されます。*

  ```
  [domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:add(port=1234)
  ```

  b. *パターン属性の編集*
  ***write-attribute** 操作を使用して、新しい値を属性に書き込みます。タブ補完を使用すると、入力時のコマンド文字列の補完に役立つほか、利用可能な属性を明らかにできます。以下の例では、**port** の値を2020 に更新します。*

  ```
  [domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:write-attribute(name=port,value=2020)
  ```

  c. *パターン属性の確認*
  *値が変更されたことを確認するには、**include-runtime=true** パラメーターを指定して **read-resource** 操作を実行し、サーバーモデルでアクティブな現在の値をすべて公開します。*

  ```
  [domain@localhost:9999 /] /socket-binding-group=standard-sockets/socket-binding=newsocket/:read-resource
  ```

- *管理コンソールを使用したソケットバインディングの設定*
  管理コンソールを使用してソケットバインディングを設定します。

  a. *管理コンソールへのログイン*
  管理対象ドメインまたはスタンドアロンサーバーの管理コンソールにログインします。

  b. ***Configuration** タブに移動します。*
  *画面上部の **Configuration** タブを選択します。*

  c. *ナビゲーションメニューから **Socket Binding** アイテムを選択します。*
  ***General Configuration** メニューを展開します。**Socket Binding** を選択します。管理対象ドメインを使用している場合は、**Socket Binding Groups** リストで対象のグループを選択します。*

  d. *新しいソケットバインディングの追加*

  i. *追加 ボタンをクリックします。*

  ii. ***Name**、**Port**、および **Binding Group** に必要な値を入力します。*

  iii. ***Save** をクリックして終了します。*

  e. *ソケットバインディングの編集*

i. 一覧からソケットバインディングを選択し、**Edit** をクリックします。

ii. **Name**、**Interface**、**Port** などの必要な値を入力します。

iii. **Save** をクリックして終了します。

バグの報告

5.2.3. JBoss EAP 6 により使用されるネットワークポート

JBoss EAP 6 のデフォルト設定で使用されるポートは、次の複数の要因によって異なります。

- サーバーグループがデフォルトのソケットバインディンググループのいずれかを使用するか、またはカスタムグループを使用するかどうか。
- 個別デプロイメントの要件。

数値ポートオフセット

同じ物理サーバーで複数のサーバーを実行する際にポートの競合を軽減するために、数値ポートオフセットを設定できます。サーバーが数値ポートオフセットを使用する場合は、サーバーグループのソケットバインディンググループのデフォルトのポート番号にオフセットを追加します。たとえば、ソケットバインディンググループの **HTTP** ポートが **8080** で、サーバーが **100** のポートオフセットを使用する場合、**HTTP** ポートは **8180** になります。

特に指定がない限り、ポートは TCP プロトコルを使用します。

デフォルトのソケットバインディンググループ

- **full-ha-sockets**
- **full-sockets**
- **ha-sockets**
- **standard-sockets**

これらのソケットバインディンググループは **domain.xml** でのみ利用できます。スタンドアロンサーバープロファイルには、標準のソケットバインディンググループのみが含まれます。このグループは **standalone.xml** の standard-sockets、**standalone-ha.xml** の **ha-sockets**、**standalone-full.xml** の場合は **full-sockets**、**standalone-full-ha.xml** の場合は **full-ha-sockets** に対応します。スタンドアロンプロファイルには、management-{native,http,https} などのソケットバインディングが含まれます。

表5.3 デフォルトのソケットバインディングの参照

名前	ポート	マルチキャストポート	説明	full-ha-sockets	full-sockets	ha-socket	standard-socket
ajp	8009		Apache JServ プロトコル。HTTP クラスタリングおよび負荷分散に使用します。	はい	はい	はい	はい

名前	ポート	マルチキャストポート	説明	full-ha-sockets	full-sockets	ha-socket	standard-socket
http	8080		デプロイされた Web アプリケーションのデフォルトポート。	はい	はい	はい	はい
https	8443		デプロイされた Web アプリケーションとクライアント間の SSL 暗号化接続。	はい	はい	はい	はい
jacorb	3528		JTS トランザクションおよび他の ORB 依存サービス用の CORBA サービス。	はい	はい	いいえ	いいえ
jacorb-ssl	3529		SSL 暗号化 CORBA サービス。	はい	はい	いいえ	いいえ
jgroups-diagnostics		7500	マルチキャスト。HA クラスターでのピア検出に使用されます。管理インターフェースを使用して設定できません。	はい	いいえ	はい	いいえ
jgroups-mping		45700	マルチキャスト。HA クラスターでの初期メンバーシップの検出に使用されます。	はい	いいえ	はい	いいえ
jgroups-tcp	7600		TCP を使用した、HA クラスター内でのユニキャストピア検出。	はい	いいえ	はい	いいえ
jgroups-tcp-fd	57600		TCP を介した HA 障害検出に使用されます。	はい	いいえ	はい	いいえ
jgroups-udp	55200	45688	UDP を使用した、HA クラスター内でのマルチキャストピア検出。	はい	いいえ	はい	いいえ
jgroups-udp-fd	54200		UDP を介した HA 障害検出に使用されます。	はい	いいえ	はい	いいえ

名前	ポート	マルチキャストポート	説明	full-ha-sockets	full-sockets	ha-socket	standard-socket
messaging	5445		JMS サービス。	はい	はい	いいえ	いいえ
messaging-group	0	9876	HornetQ JMS ブロードキャストと検出グループにより参照されます。	はい	はい	いいえ	いいえ
messaging-throughput	5455		JMS Remoting により使用されます。	はい	はい	いいえ	いいえ
mod_cluster		23364	JBoss EAP 6 と HTTP ロードバランサー間の通信に対するマルチキャストポート。	はい	いいえ	はい	いいえ
remoting	4447		リモート EJB の呼び出しに使用されます。	はい	はい	はい	はい
txn-recovery-environment	4712		JTA トランザクションリカバリーマネージャー。	はい	はい	はい	はい
txn-status-manager	4713		JTA / JTS トランザクションマネージャー。	はい	はい	はい	はい

管理ポート

ソケットバインディンググループの他に、各ホストコントローラーによって2 つのポートが管理目的で開かれます。

- ***9990 - Web 管理コンソールポート***
- ***9999 - 管理コンソールと管理 API が使用するポート***

*さらに、管理コンソールに対して **HTTPS** が有効になっている場合、**9443** もデフォルトポートとして開かれます。*

バグの報告

5.2.4. ソケットバインディンググループのポートオフセット

ポートオフセットは、そのサーバーのソケットバインディンググループによって提供されたポート値に追加される数値オフセットです。これにより、単一のサーバーは所属するサーバーグループのソケットバインディングを継承でき、オフセットを使用してグループ内の他のサーバーと競合しないようにできます。たとえば、ソケットバインディンググループの **HTTP** ポートが **8080** で、サーバーが **100** のポートオフセットを使用する場合、**HTTP** ポートは **8180** になります。

バグの報告

5.2.5. ポートオフセットの設定

- ポートオフセットの設定
  管理 **CLI** または管理コンソールを選択してポートオフセットを設定します。
  - 管理 **CLI** を使用したポートオフセットの設定
    管理 **CLI** を使用してポートオフセットを設定します。
    a. ポートオフセットの編集
       write-attribute 操作を使用して、新しい値をポートオフセットに書き込みます。以下の例では、server-two の **socket-binding-port-offset** 値を **250** に更新しています。このサーバーは、デフォルトのローカルホストグループのメンバーです。変更を有効にするには、再起動が必要です。

       ```
       [domain@localhost:9999 /] /host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)
       ```

    b. ポートオフセット属性の確認
       値が変更されたことを確認するには、**include-runtime=true** パラメーターを指定して **read-resource** 操作を実行し、サーバーモデルでアクティブな現在の値をすべて公開します。

       ```
       [domain@localhost:9999 /] /host=master/server-config=server-two/:read-resource(include-runtime=true)
       ```

  - 管理コンソールを使用したポートオフセットの設定
    管理コンソールを使用してポートオフセットを設定します。
    a. 管理コンソールへのログイン
       管理対象ドメインの管理コンソールにログインします。
    b. ドメイン タブの選択
       画面上部の ドメイン タブを選択します。
    c. ポートオフセット属性の編集
       i. **Available Server Configurations** リストでサーバーを選択し、以下のアセンブリーリストの上部にある **Edit** をクリックします。
       ii. **Port Offset** フィールドに任意の値を入力します。
       iii. **Save** をクリックして終了します。

バグの報告

5.3. IPV6

5.3.1. IPv6 ネットワーキング向け JVM スタック設定の指定

概要

このトピックでは、JBoss EAP 6 インストールでの IPv6 ネットワーキングの有効化について説明します。

手順5.1 IPv4 Stack Java プロパティーの無効化

1. *インストールに関連するファイルを開きます。*
   - *スタンドアロンサーバーの場合:*
     *Open **EAP_HOME/bin/standalone.conf**.*
   - *管理対象ドメインの場合:*
     *Open **EAP_HOME/bin/domain.conf**.*
2. *IPv4 Stack Java プロパティーを false に変更します。*

   ```
   -Djava.net.preferIPv4Stack=false
   ```

   以下に例を示します。

   例5.8 JVM オプション

   ```
   # Specify options to pass to the Java VM.
   #
   if [ "x$JAVA_OPTS" = "x" ]; then
      JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
   Djava.net.preferIPv4Stack=false
      -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
      -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv6Addresses=true"
   fi
   ```

バグの報告

5.3.2. IPv6 ネットワークに対するインターフェース宣言の設定

概要

次の手順に従って、インターフェース inet アドレスを IPv6 のデフォルトに設定します。

前提条件

- *「JBoss EAP 6 の起動」*
- *「管理コンソールへのログイン」*

手順5.2 IPv6 ネットワーキングのインターフェースの設定

1. 画面上部の **Configuration** タブを選択します。
2. **General Configuration** メニューを展開し、**Interfaces** を選択します。
3. **Available Interfaces** リストからインターフェースを選択します。
4. 詳細一覧で **Edit** をクリックします。
5. 下記の **inet** アドレスを設定します。

   ```
   ${jboss.bind.address.management:[ADDRESS]}
   ```

6. **Save** をクリックして終了します。
7. サーバーを再起動し、変更を実装します。

バグの報告

5.3.3. IPv6 アドレス用 JVM スタック設定の指定

概要

このトピックでは、**JBoss EAP 6** インストールで **IPv6** アドレスを優先するよう設定ファイルで設定する方法について説明します。

手順5.3 IPv6 アドレスを優先するよう JBoss EAP 6 インストールを設定する

1. インストールに関連するファイルを開きます。
   - スタンドアロンサーバーの場合:
     Open **EAP_HOME/bin/standalone.conf**.
   - 管理対象ドメインの場合:
     Open **EAP_HOME/bin/domain.conf**.
2. 以下の **Java** プロパティーを **Java VM** オプションに追加します。

   ```
   -Djava.net.preferIPv6Addresses=true
   ```

   以下に例を示します。

   例5.9 JVM オプション

   ```
   # Specify options to pass to the Java VM.
   #
   if [ "x$JAVA_OPTS" = "x" ]; then
      JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
   Djava.net.preferIPv4Stack=false
      -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000
      -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv6Addresses=true"
   fi
   ```

バグの報告

5.4. REMOTING

5.4.1. リモーティングのメッセージサイズの設定

remoting サブシステムは、リモーティングプロトコルのメッセージのサイズを制限するオプションを提供します。最大インバウンドメッセージサイズ(**MAX_INBOUND_MESSAGE_SIZE**)および最大送信メッセージサイズ(**MAX_OUTBOUND_MESSAGE_SIZE**)を設定して、適切なサイズ制限内でメッセージが受信され、送信されるようにすることができます。

MAX_INBOUND_MESSAGE_SIZE および **MAX_OUTBOUND_MESSAGE_SIZE** は ejb3 サブシステムで設定でき、値はバイト単位です。

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.5">
.....
 <remote connector-ref="remoting-connector" thread-pool-name="default">
  <channel-creation-options>
   <option name="MAX_INBOUND_MESSAGE_SIZE" value="xxxxx" type="remoting"/>
   <option name="MAX_OUTBOUND_MESSAGE_SIZE" value="xxxxx" type="remoting"/>
  </channel-creation-options>
 </remote>
.....
</subsystem>
```

リモーティングプロトコルのメッセージサイズを設定すると、システムメモリーが効率的に使用され、重要な操作の実行中にメモリー不足が発生しないようになります。

送信側が最大許容制限(**MAX_OUTBOUND_MESSAGE_SIZE**)を超えるメッセージを送信する場合、サーバーは例外をスローし、データの送信を取り消します。しかし、コネクションは開いたままとなり、必要に応じて送信側はメッセージを閉じることができます。

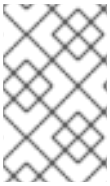
受信したメッセージが最大許容制限(**MAX_INBOUND_MESSAGE_SIZE**)を超える場合、接続が開いたまま、メッセージは非同期に閉じられます。

バグの報告

5.4.2. Remoting サブシステムの設定

概要

JBoss Remoting には、ワーカースレッドプール、1 つ以上のコネクター、および一連のローカルおよびリモート接続 URI の 3 つのトップレベル設定可能な要素があります。このトピックでは、設定可能な各項目の説明、各項目の設定方法についての CLI コマンドの例、および完全に設定されたサブシステムの XML の例を紹介します。この設定はサーバーにのみ適用されます。独自のアプリケーションにカスタムコネクターを使用しない限り、ほとんどのユーザーは Remoting サブシステムをまったく設定する必要はありません。EJB などの Remoting クライアントとして動作するアプリケーションには、特定のコネクターに接続するための個別の設定が必要です。



注記

Remoting サブシステム設定は Web ベースの管理コンソールに公開されませんが、コマンドラインベースの管理 CLI から完全に設定可能です。手作業による XML の編集は推奨されません。

CLI コマンドの適合

CLI コマンドは、デフォルト のプロファイルの設定時に管理対象ドメインに対して式化されます。別のプロファイルを設定するには、その名前を置き換えます。スタンドアロンサーバーの場合は、コマンドの **/profile=default** 部分を省略します。

Remoting サブシステム外の設定

remoting サブシステム以外の設定側面がいくつかあります。

ネットワークインターフェース

remoting サブシステムによって使用されるネットワークインターフェースは、**domain/configuration/domain.xml** または **standalone/configuration/standalone.xml** で定義されたパブリック インターフェースです。

```
<interfaces>
  <interface name="management"/>
  <interface name="public"/>
  <interface name="unsecure"/>
</interfaces>
```

パブリック インターフェースのホストごとの定義は、**domain.xml** または **standalone.xml** と同じディレクトリーの **host.xml** で定義されます。このインターフェースは、他のサブシステムによっても使用されます。変更時には注意が必要です。

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:127.0.0.1}"/>
  </interface>
  <interface name="unsecure">
    <!-- Used for IIOP sockets in the standard configuration.
         To secure JacORB you need to setup SSL -->
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

socket-binding

remoting サブシステムによって使用されるデフォルトの socket-binding は TCP ポート 4447 にバインドされます。これを変更する必要がある場合は、ソケットバインディングおよびソケットバインディンググループのドキュメントを参照してください。

ソケットバインディンググループとソケットバインディンググループに関する情報は、JBoss EAP 『**管理ガイドのソケットバインディンググループの章を参照し**』てください。 『』
https://access.redhat.com/documentation/ja-jp/red_hat_jboss_enterprise_application_platform/?version=6.4

EJB のリモーティングコネクターリファレンス

EJB サブシステムには、リモートメソッド呼び出しに対するリモーティングコネクターへの参照が含まれます。デフォルト設定は次のとおりです。

```
<remote connector-ref="remoting-connector" thread-pool-name="default"/>
```

セキュアなトランスポート設定

リモーティングトランスポートは、クライアントが要求した場合に StartTLS を使用してセキュアな（HTTPS、Secure Servlet など）接続を使用します。セキュアな接続とセキュアでない接続に同じソケットバインディング（ネットワークポート）が使用されるため、サーバー側の追加設定は必要ありません。クライアントは必要に応じてセキュアなトランスポートまたはセキュアでないトランスポートを要求します。EJB、ORB、JMS プロバイダーなどの Remoting を使用する JBoss EAP 6 コンポーネントは、デフォルトでセキュアなインターフェースを要求します。

警告：STARTTLS セキュリティーに関する考慮事項

StartTLS は、クライアントが要求した場合にセキュアな接続をアクティベートし、セキュアでない接続にデフォルト設定することで機能します。これは、攻撃者がクライアントの要求をインターセプト してセキュアでない接続を要求するために Middle スタイルの不正使用における Man に悪影響を与えることが本質的に行われます。セキュアでない接続が実際に適切なフォールバックである場合を除き、クライアントがセキュアな接続を受信しない場合、適切に失敗するよう記述する必要があります。

ワーカースレッドプール

ワーカースレッドプールは、Remoting コネクターを介して提供される作業の処理に使用できるスレッドのグループです。これは単一の要素 &lt ;worker-thread-pool> で、複数の属性を取ります。ネットワークタイムアウトの取得、スレッドの不足、またはメモリー使用量の制限が必要な場合には、これらの属性をチューニングします。特定の推奨事項は、特定の状況によって異なります。詳細は Red Hat グローバルサポートサービスまでお問い合わせください。

表5.4 ワーカースレッドプールの属性

属性	説明	CLI コマンド
read-threads	リモーティングワーカーに対して作成する読み取りスレッドの数。デフォルトは**1**です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-read-threads,value=1)**
write-threads	リモーティングワーカーに作成する書き込みスレッドの数。デフォルトは**1**です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-write-threads,value=1)**
task-keepalive	コアでないリモーティングワーカーのタスクスレッドにキープアライブを使用する期間 (ミリ秒単位)。デフォルトは **60** です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-keepalive,value=60)**

属性	説明	CLI コマンド
task-max-threads	リモーティングワーカーのタスクスレッドプールに対するスレッドの最大数。デフォルトは **16** です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-max-threads,value=16)**
task-core-threads	リモーティングワーカーのタスクスレッドプールに対するコアスレッドの数。デフォルトは **4** です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-core-threads,value=4)**
task-limit	許可するリモーティングワーカータスクの最大数。 この数を超えるリモーティングワーカータスクは拒否されます。デフォルトは **16384** です。	**/profile=default/subsystem=remoting/:write-attribute(name=worker-task-limit,value=16384)**

コネクター

*コネクターは主な **Remoting** 設定要素です。複数のコネクターを設定できます。それぞれは複数のサブ要素を持つ **<connector>** 要素といくつかの属性で構成されます。デフォルトのコネクターは、**JBoss EAP 6** の複数のサブシステムによって使用されます。カスタムコネクターの要素や属性の設定はアプリケーションに依存するため、詳細は **Red Hat** グローバルサポートサービスにお問い合わせください。*

表5.5 コネクターの属性

属性	説明	CLI コマンド
socket-binding	このコネクターに使用するソケットバインディングの名前。	**/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=socket-binding,value=remoting)**
authentication-provider	このコネクターで使用する JASPIC(Java Authentication Service Provider Interface for Containers)モジュール。モジュールはクラスパスにある必要があります。	**/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=authentication-provider,value=myProvider)**
security-realm	オプション。アプリケーションのユーザー、パスワード、およびロールが含まれるセキュリティーレルム。EJB または Web アプリケーションはセキュリティーレルムに対して認証できます。**ApplicationRealm** はデフォルトの JBoss EAP 6 インストールで利用できます。	**/profile=default/subsystem=remoting/connector=remoting-connector/:write-attribute(name=security-realm,value=ApplicationRealm)**

表5.6 コネクター要素

属性	説明	CLI コマンド
sasl	SASL(Simple Authentication and Security Layer)認証メカニズムの組み込み要素	**該当なし**
properties	1 つ以上の &lt **;property&gt**; 要素が含まれ、それぞれ **name** 属性と任意の **value** 属性が含まれます。	**/profile=default/subsystem=remoting/connector=remoting-connector/property=myProp/:add(value=myPropValue)**

送信接続

3 種類のアウトバウンド接続を指定することができます。

- ***URI** への送信接続。*
- *ローカルアウトバウンド接続： ソケットなどのローカルリソースに接続します。*
- *リモートアウトバウンド接続： リモートリソースに接続し、セキュリティーレルムを使用して認証します。*

*すべてのアウトバウンド接続は< **outbound-connections>** 要素に囲まれて います。これらの各接続タイプは **outbound-socket-binding-ref** 属性を取ります。outbound-connection は **uri** 属性を取ります。リモートアウトバウンド接続は、承認に使用する任意の **username** および **security-realm** 属性を取ります。*

表5.7 送信接続要素

属性	説明	CLI コマンド
outbound-connection	汎用アウトバウンド接続。	**/profile=default/subsystem=remoting/outbound-connection=my-connection/:add(uri=http://my-connection)**
local-outbound-connection	暗黙的な local:// URI スキームを使用した送信接続。	**/profile=default/subsystem=remoting/local-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting2)**
remote-outbound-connection	セキュリティーレルムで basic/digest 認証を使用した remote:// URI スキームの送信接続。	**/profile=default/subsystem=remoting/remote-outbound-connection=my-connection/:add(outbound-socket-binding-ref=remoting,username=myUser,security-realm=ApplicationRealm)**

SASL 要素

SASL 子要素を定義する前に、最初の SASL 要素を作成する必要があります。以下のコマンドを使用します。

```
/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:add
```

SASL 要素の子要素は以下の表で説明されています。

表5.8 SASL 子要素

属性	説明	CLI コマンド
include-mechanisms	SASL メカニズムのリストである **value** 属性が含まれます。	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=include-mechanisms,value=["DIGEST","PLAIN","GSSAPI"])`
qop	優先順で SASL Quality of 保護値のリストである **value** 属性が含まれます。	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=qop,value=["auth"])`
strength	**value** 属性が含まれます。これは SASL 暗号強度の値のリストで優先順で表されます。	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=strength,value=["medium"])`
reuse-session	ブール値である **value** 属性が含まれます。true の場合、セッションを再利用しようとします。	`/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=reuse-session,value=false)`

属性	説明	CLI コマンド
server-auth	ブール値である **value** 属性が含まれます。true の場合、サーバーはクライアントに対して認証します。	/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl:write-attribute(name=server-auth,value=false)
policy	ゼロ以上の要素が含まれるローカリング要素。それぞれが単一の **値** を取ります。 ● forward-secrecy: 前方機密性の実装にメカニズムが必要であるかどうか（1 つのセッションに分割しても、今後のセッションに侵入するための情報が自動的に提供されない） ● 非アクティブ - 辞書攻撃を受けやすいメカニズムを許可するかどうか。**false** を指定すると許可され、**true** は拒否されます。 ● no-anonymous: 匿名ログインを受け入れるメカニズムを許可するかどうか。**false** を指定すると許可され、**true** は拒否されます。 ● no-dictionary: パッシブ辞書攻撃を受けやすいメカニズムを許可するかどうか。**false** を指定すると許可され、**true** は拒否されます。 ● no-plain-text: 単純なプレーンテキスト攻撃を受けやすいメカニズムを許可するかどうか。**false** を指定すると許可され、**true** は拒否されます。 ● pass-credentials: クライアントのクレデンシャルを渡すメカニズムを許可するかどうか。	/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:add /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=forward-secrecy,value=true) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-active,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-anonymous,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-dictionary,value=true)

属性	説明	CLI コマンド
		/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=no-plain-text,value=false) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/sasl-policy=policy:write-attribute(name=pass-credentials,value=true)
properties	1 つ以上の &lt **;property&gt**; 要素が含まれ、それぞれ **name** 属性と任意の **value** 属性が含まれます。	/profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/property=myprop:add(value=1) /profile=default/subsystem=remoting/connector=remoting-connector/security=sasl/property=myprop2:add(value=2)

例5.10 設定例

以下の例は、JBoss EAP 6 に同梱されるデフォルトの remoting サブシステムを示しています。

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
    <connector name="remoting-connector" socket-binding="remoting" security-
realm="ApplicationRealm"/>
</subsystem>
```

この例には多くのハイフン的な値が含まれており、前述の要素と属性をコンテキストに配置するように提示されています。

```
<subsystem xmlns="urn:jboss:domain:remoting:1.1">
    <worker-thread-pool read-threads="1" task-keepalive="60" task-max-threads="16" task-core-
thread="4" task-limit="16384" write-threads="1" />
    <connector name="remoting-connector" socket-binding="remoting" security-
realm="ApplicationRealm">
        <sasl>
            <include-mechanisms value="GSSAPI PLAIN DIGEST-MD5" />
            <qop value="auth" />
            <strength value="medium" />
            <reuse-session value="false" />
            <server-auth value="false" />
```

```
            <policy>
                <forward-secrecy value="true" />
                <no-active value="false" />
                <no-anonymous value="false" />
                <no-dictionary value="true" />
                <no-plain-text value="false" />
                <pass-credentials value="true" />
            </policy>
            <properties>
                <property name="myprop1" value="1" />
                <property name="myprop2" value="2" />
            </properties>
        </sasl>
        <authentication-provider name="myprovider" />
        <properties>
            <property name="myprop3" value="propValue" />
        </properties>
    </connector>
    <outbound-connections>
        <outbound-connection name="my-outbound-connection" uri="http://myhost:7777/"/>
        <remote-outbound-connection name="my-remote-connection" outbound-socket-binding-
ref="my-remote-socket" username="myUser" security-realm="ApplicationRealm"/>
        <local-outbound-connection name="myLocalConnection" outbound-socket-binding-ref="my-
outbound-socket"/>
    </outbound-connections>
</subsystem>
```

設定イントロスペクションが文書化されていない

- ***JNDI** およびマルチキャストの自動検出*

バグの報告

第6章 データソース管理

6.1. はじめに

6.1.1. JDBC

JDBC API は、Java アプリケーションがデータベースにアクセスする方法を定義する基準です。アプリケーションは JDBC ドライバーを参照するデータソースを設定します。その後、データベースではなくドライバーに対してアプリケーションを記述できます。ドライバーはコードをデータベース言語に変換します。そのため、適切なドライバーがインストールされていればアプリケーションをサポートされるデータベースで使用できます。

JDBC 4.0 仕様は、に定義されてい http://jcp.org/en/jsr/detail?id=221 ます。

手順6.1 管理コンソールを使用したデータソースプール接続のテスト

1. 「管理コンソールへのログイン」.
2. 管理コンソールの上部から Runtime タブを選択します。
3. 左側の Subsystems メニューを展開し、Datasources を選択します。
4. **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。

手順6.2 管理 CLI を使用したデータソース接続のテスト

1. CLI ツールを起動し、サーバーに接続します。
2. 以下の管理 CLI コマンドを実行して接続をテストします。
   - スタンドアロンモードでは、以下のコマンドを入力します。

     ```
     /subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
     ```

   - ドメインモードでは、以下のコマンドを入力します。

     ```
     host=master/server=server-one/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
     ```

JDBC とデータソースの使用を開始するにあたっては、JBoss EAP 6 の管理設定ガイドの JDBC ドライバーのセクションを参照してください。

バグの報告

6.1.2. JBoss EAP 6 でサポートされるデータベース

JBoss EAP 6 でサポートされる JDBC 準拠のデータベースの一覧は、を参照し https://access.redhat.com/articles/111663 てください。

バグの報告

6.1.3. データソースの型

リソースの一般的なタイプには、データソースと **XA** データソース があります。

非 XA データソースは、トランザクションを使用しないアプリケーション、または単一のデータベースでトランザクションを使用するアプリケーションに使用されます。

XA データソースは、トランザクションが複数のデータベースにわたって分散されるアプリケーションによって使用されます。XA データソースを使用すると追加のオーバーヘッドが発生します。

管理コンソールあるいは管理 CLI でデータソースを作成するときに、データソースの型を指定します。

バグの報告

6.1.4. データソースの例

JBoss EAP 6 には H2 データベースが含まれています。これは、開発者がアプリケーションを迅速に構築できるように、軽量でリレーショナルデータベース管理システムであり、プラットフォームのデータソースの例になります。

警告

ただし、実稼働環境では使用しないでください。これは、アプリケーションをテストおよび構築するために必要なすべての標準をサポートする非常に小さい自己完結型のデータソースです。ただし、本番稼働用として堅牢または拡張性はありません。

サポートされるデータソースおよび認定済みのデータソースのリストは、「JBoss EAP 6 でサポートされるデータベース」 を参照してください。

バグの報告

6.1.5. -ds.xml ファイルのデプロイメント

JBoss EAP 6 では、データソースは server サブシステムのリソースとして定義されます。以前のバージョンでは、サーバー設定のデプロイメントディレクトリーに ***-ds.xml** データソース設定ファイルが必要でした。***-ds.xml** ファイルは、以下の 『Schemas』 で利用可能な 1.1 データソーススキーマに従って JBoss EAP 6 にデプロイでき http://www.ironjacamar.org/documentation.html ます。

警告

この機能は開発目的でのみ使用してください。JBoss の管理ツールではサポートされないため、本番環境では推奨されません。この機能は JBoss EAP 6.4 で非推奨となり、製品の次期メジャーリリースではサポートされません。

重要

***-ds.xml** ファイルのデプロイ時に、すでにデプロイ/定義された<driver> エントリーへの参照を使用する必要があります。*

バグの報告

6.2. JDBC ドライバー

6.2.1. 管理コンソールを用いた JDBC ドライバーのインストール

概要

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。これは、管理対象ドメインを使用する場合、サーバーグループの複数のサーバーにデプロイできることを意味します。

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 準拠のドライバーは自動的に認識され、名前とバージョンでシステムにインストールされます。JDBC JAR は、Java サービスプロバイダー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 Driver クラスの名前が含まれる META-INF/services/java.sql.Driver テキストが含まれます。

手順6.3 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. *「管理コンソールへのログイン」*
7. *管理対象ドメインを使用する場合は、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。「管理コンソールを使用してデプロイされたアプリケーションを有効化」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。

バグの報告

6.2.2. コアモジュールとしての JDBC ドライバーのインストール

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。JDBC ドライバーのダウンロード場所は、「JDBC ドライバーをダウンロードできる場所」に記載されています。*
- *アーカイブを展開します。*

手順6.4 管理 CLI を使用してコアモジュールとしての JDBC ドライバーのインストール

1. *サーバーを起動します。*
2. *管理 CLI を起動しますが、稼働中のインスタンスへの接続に **--connect** 引数または **-c** 引数を使用しないでください。*
3. ***module add** CLI コマンドを使用して、新しいモジュールを追加します。*

> *例6.1 MySQL JDBC ドライバーモジュールを追加する CLI コマンドの例*
>
> ```
> module add --name=com.mysql --resources=/path/to/mysql.jar --
> dependencies=javax.api,javax.transaction.api
> ```

注記

***module** 管理 CLI コマンドを使用してモジュールの追加および削除はテクノロジープレビューとしてのみ提供されるため、リモート JBoss EAP インスタンスにモジュールを追加しないでください。本番環境では、モジュールを手作業で追加および削除する必要があります。*

以下の手順に従って、モジュールを手動で追加します。

1. *ファイルパス構造は **EAP_HOME/modules/** ディレクトリーの下に作成されます。たとえば、MySQL JDBC ドライバーの場合は、**EAP_HOME/modules/com/mysql/main/** のようになります。*
2. *リソースとして指定された JAR ファイルは **main/** サブディレクトリーにコピーされます。*
3. *指定の依存関係を持つ module.xml ファイルが **main/** サブディレクトリーに作成されます。**module.xml** ファイルの例は、「モジュール」を参照してください。*

*このコマンドを使用してモジュールを追加および削除する方法は、**module --help** を実行します。*

4. *CLI コマンド **connect** を使用して、実行中のインスタンスに接続します。*

5. *CLI コマンドを実行して JDBC ドライバーモジュールをサーバー設定に追加します。*

   *選択するコマンドは、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ているクラスの数によって異なります。たとえば、MySQL 5.1.20 JDBC JAR の **/META-INF/services/java.sql.Driver** ファイルには、2 つのクラスが一覧表示されます。*

   - ***com.mysql.jdbc.Driver***
   - ***com.mysql.fabric.jdbc.FabricMySQLDriver***

   複数のエントリーがある場合は、ドライバークラスの名前も指定する必要があります。これを実行しないと、以下のようなエラーが発生します。

   > *例6.2 ドライバークラスエラー*
   >
   > ***JBAS014749: Operation handler failed: Service jboss.jdbc-driver.mysql is already registered***

   - *1 つのクラスエントリーを含む JDBC JAR に対して CLI コマンドを実行します。*

     > ***/subsystem=datasources/jdbc-driver=***DRIVER_NAME***:add(driver-name=***DRIVER_NAME***,driver-module-name=***MODULE_NAME***,driver-xa-datasource-class-name=***XA_DATASOURCE_CLASS_NAME***)***

     > *例6.3 1 つのドライバークラスを持つ JDBC JAR に対するスタンドアロンモードの CLI コマンド*
     >
     > ***/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)***

     > *例6.4 1 つのドライバークラスを持つ JDBC JAR に対するドメインモードの CLI コマンド*
     >
     > ***/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource)***

   - *複数のクラスエントリーを含む JDBC JAR に対して CLI コマンドを実行します。*

     > ***/subsystem=datasources/jdbc-driver=***DRIVER_NAME***:add(driver-name=***DRIVER_NAME***,driver-module-name=***MODULE_NAME***,driver-xa-datasource-class-name=***XA_DATASOURCE_CLASS_NAME***, driver-class-name=***DRIVER_CLASS_NAME***)***

     > *例6.5 複数のドライバークラスエントリーを持つ JDBC JAR に対するスタンドアロンモードの CLI コマンド*

```
/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-name=com.mysql.jdbc.Driver)
```

例6.6 複数のドライバークラスエントリーを持つ JDBC JAR に対するドメインモードの CLI コマンド

```
/profile=ha/subsystem=datasources/jdbc-driver=mysql:add(driver-name=mysql,driver-module-name=com.mysql,driver-xa-datasource-class-name=com.mysql.jdbc.jdbc2.optional.MysqlXADataSource, driver-class-name=com.mysql.jdbc.Driver)
```

例6.7 複数の非 XA ドライバークラスエントリーを持つ JDBC JAR に対するドメインモードの CLI コマンド

```
/profile=ha/subsystem=datasources/jdbc-driver=oracle/:add(driver-module-name=com.oracle,driver-name=oracle,jdbc-compliant=true,driver-datasource-class-name=oracle.jdbc.OracleDriver)
```

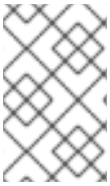
結果

JDBC ドライバーがインストールされ、コアモジュールとして設定されます。アプリケーションのデータソースが JDBC ドライバーを参照できる状態になります。

バグの報告

6.2.3. JDBC ドライバーをダウンロードできる場所

以下の表は、JBoss EAP 6 で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。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6.1 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

ベンダー	ダウンロード場所
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/

バグの報告

6.2.4. ベンダー固有クラスへのアクセス

概要

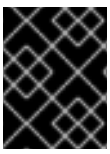
このトピックでは、JDBC 固有クラスを使用するために必要な手順について説明します。これは、アプリケーションが JDBC API の一部ではないベンダー固有の機能を使用する必要がある場合はに必要です。

警告

これは高度な使用法です。JDBC API に含まれない機能を必要とするアプリケーションのみこれを実装します。

重要

このプロセスは、再認証メカニズムを使用し、ベンダー固有のクラスにアクセスする場合に必要です。



重要

接続は IronJacamar コンテナーによって制御されるため、ベンダー固有の API ガイドラインに厳密に従ってください。

前提条件

- *「コアモジュールとしての JDBC ドライバーのインストール」.*

手順6.5 アプリケーションへの依存関係の追加

以下の方法のいずれかを使用して、依存関係をアプリケーションに追加できます。任意の方法を選択します。

- ○ ***MANIFEST.MF** ファイルの設定*
    a. *テキストエディターでアプリケーションの **META-INF/MANIFEST.MF** ファイルを開きます。*

b. JDBC モジュールの依存関係を追加し、ファイルを保存します。

> **Dependencies:** MODULE_NAME

> 例6.8 com.mysql が依存関係として宣言されている MANIFEST.MF ファイル
>
> ```
> Dependencies: com.mysql
> ```

- a. **jboss-deployment-structure.xml** ファイルの作成
  アプリケーションの **META-INF/** または **WEB-INF** フォルダーに **jboss-deployment-structure.xml** というファイルを作成します。

> 例6.9 com.mysql が依存関係として宣言されている **jboss-deployment-structure.xml** ファイル
>
> ```
> <jboss-deployment-structure>
>  <deployment>
>   <dependencies>
>    <module name="com.mysql" />
>   </dependencies>
>  </deployment>
> </jboss-deployment-structure>
> ```

例6.10 ベンダー固有 API へのアクセス

以下のサンプルは MySQL API にアクセスします。

```
import java.sql.Connection;
import org.jboss.jca.adapters.jdbc.WrappedConnection;

 Connection c = ds.getConnection();
 WrappedConnection wc = (WrappedConnection)c;
 com.mysql.jdbc.Connection mc = wc.getUnderlyingConnection();
```

バグの報告

6.3. 非 XA データソース

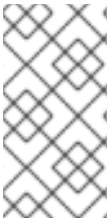
6.3.1. 管理インターフェースによる非 XA データソースの作成

概要

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。



ORACLE のデータソース

バージョン 10.2 以前の Oracle データソースでは、トランザクション以外の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するため、<no-tx-separate-pools/> パラメーターが必要でした。特定のアプリケーションでこのパラメーターが不要になる場合があります。

ドメインモード

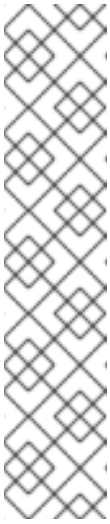
ドライバーリストの重複などの問題を防ぐために、選択したドライバーがプロファイルで利用できないか、JBoss EAP 6.4 以降ではプロファイルのサーバーが実行されていない場合に表示されないため、JBoss EAP 6.4 以降ではモジュールとしてインストールされ、プロファイルから適切に参照される JDBC ドライバーのみが、ドメインモードで管理コンソールを使用してデータソースを作成する時に検出可能です。

手順6.6 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

- *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ている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、値は JAR の名前になります。複数のクラスがある場合、値は JAR + driverClassName + "_" + majorVersion +"_" + minorVersion の名前になります。これを実行しないと、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 ドライバーに必要な 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

- *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. *コンソールの上部にある **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から **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** をクリックします。

ii. **Create Datasource** ウィザードに新しいデータソース属性を入力し、**Next** ボタンをクリックします。

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

iv. **Create Datasource** ウィザードにコネクション設定を入力します。

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. 完了 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。これで、**standalone.xml** ファイルまたは **domain.xml** ファイル、および管理インターフェースが表示されます。

バグの報告

6.3.2. 管理インターフェースによる非 XA データソースの編集

概要

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 データソースを編集する手順について取り上げます。

前提条件

- 「JBoss EAP 6 の起動」.

JTA の統合

非 XA データソースは JTA トランザクションと統合できます。データソースを JTA と統合するには、**jta** パラメーターが **true** に設定されていることを確認します。

手順6.7 非 XA データソースの編集

- 管理 CLI

a. 「管理 CLI の起動」.

b. **write-attribute** コマンドを使用してデータソース属性を設定します。

```
/subsystem=datasources/data-source=DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

c. サーバーを再ロードして変更を確認します。

```
reload
```

- 管理コンソール

a. 「管理コンソールへのログイン」.

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

i. コンソールの上部にある **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. 拡張されたメニューから Datasources を選択します。

c. データソースを編集します。

i. **Available Datasources** リストからデータソースを選択します。データソース属性は以下に表示されます。

ii. **Edit** をクリックしてデータソース属性を編集します。

iii. **Save** をクリックして終了します。

結果

非 XA データソースが設定されている。変更が、**standalone.xml** ファイルまたは **domain.xml** ファイルのいずれかと管理インターフェースで表示されるようになりました。

- 新しいデータソースを作成するには、「管理インターフェースによる非 XA データソースの作成」 を参照してください。
- データソースを削除するには、「管理インターフェースによる非 XA データソースの削除」 を参照してください。

バグの報告

6.3.3. 管理インターフェースによる非 XA データソースの削除

概要

ここでは、管理コンソールまたは管理 CLI を使用して、JBoss EAP 6 より非 XA データソースを削除するために必要な手順について説明します。

前提条件

- 「JBoss EAP 6 の起動」.

手順6.8 非 XA データソースの削除

手順6.8 非XA データソースの削除

- 管理CLI
  a. 「管理CLI の起動」.
  b. 次のコマンドを実行して非XA データソースを削除します。

```
data-source remove --name=DATASOURCE_NAME
```

- 管理コンソール
  a. 「管理コンソールへのログイン」.
  b. 管理コンソールの **Datasources** パネルに移動します。
    i. コンソールの上部にある **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. 削除 するデータソースを選択し、Remove をクリックします。

結果

非XA データソースがサーバーより削除されます。

- 新しいデータソースを追加するには、「管理インターフェースによる非XA データソースの作成」 を参照してください。

バグの報告

6.4. XA データソース

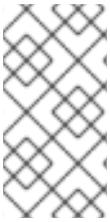
6.4.1. 管理インターフェースによるXA データソースの作成

要件:

- 「JBoss EAP 6 の起動」

概要

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。



ORACLE のデータソース

バージョン10.2 以前のOracle データソースでは、トランザクション以外の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するため、<no-tx-separate-pools/> パラメーターが必要でした。特定のアプリケーションでこのパラメーターが不要になる場合があります。

手順6.9 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

- *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ている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、値は JAR の名前になります。複数のクラスがある場合、値は JAR + driverClassName + "_" + majorVersion +"_" + minorVersion の名前になります。これを実行しないと、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 ドライバーに必要な 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

- *管理コンソール*

  a. *「管理コンソールへのログイン」.*

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

  i. *コンソールの上部にある **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. **Add** をクリックします。

ii. **Create XA Datasource** ウィザードで新しい XA データソース属性を入力し、**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. テスト接続 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. 完了 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ている。これで、**standalone.xml** ファイルまたは **domain.xml** ファイル、および管理インターフェースが表示されます。

以下も参照してください。

- *「管理インターフェースによる XA データソースの編集」*
- *「管理インターフェースによる XA データソースの削除」*

バグの報告

6.4.2. 管理インターフェースによる XA データソースの編集

概要

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 データソースを編集する手順について取り上げます。

前提条件

- *「JBoss EAP 6 の起動」.*

手順6.10 管理 CLI または管理コンソールのいずれかを使用した XA データソースの編集

- ○ 管理 CLI

a. 「管理 CLI の起動」.

b. XA データソース属性の設定
write-attribute コマンドを使用してデータソース属性を設定します。

```
/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

c. XA データソースプロパティーの設定
次のコマンドを実行して XA データソースサブリソースを設定します。

```
/subsystem=datasources/xa-data-source=DATASOURCE_NAME/xa-datasource-properties=PROPERTY_NAME:add(value=PROPERTY_VALUE)
```

d. サーバーを再ロードして変更を確認します。

```
reload
```

- 管理コンソール

a. 「管理コンソールへのログイン」.

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

i. コンソールの上部にある **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. データソースを編集します。

i. **Available XA Datasources** リストから関連する XA データソースを選択します。
XA データソース属性は、その下の **Attributes** パネルに表示されます。

ii. **Edit** ボタンを選択してデータソース属性を編集します。

iii. XA データソース属性を編集し、作業が完了したら **Save** ボタンを選択します。

結果

XA データソースが設定されている。変更が、**standalone.xml** ファイルまたは **domain.xml** ファイルのいずれかと管理インターフェースで表示されるようになりました。

- 新しいデータソースを作成するには、「管理インターフェースによる XA データソースの作成」 を参照してください。
- データソースを削除するには、「管理インターフェースによる XA データソースの削除」 を参照してください。

バグの報告

6.4.3. 管理インターフェースによる XA データソースの削除

概要

ここでは、管理コンソールまたは管理 CLI を使用して、JBoss EAP 6 から XA データソースを削除するために必要な手順について説明します。

前提条件

- 「JBoss EAP 6 の起動」.

手順6.11 管理 CLI または管理コンソールのいずれかを使用した XA データソースの削除

- ○ 管理 CLI
  a. 「管理 CLI の起動」.
  b. 次のコマンドを実行して XA データソースを削除します。

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

- ○ 管理コンソール
  a. 「管理コンソールへのログイン」.
  b. 管理コンソールの **Datasources** パネルに移動します。
     i. コンソールの上部にある **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. 削除する登録済みの XA データソースを選択し、Remove をクリックして XA データソースを永続的に 削除 します。

結果

XA データソースがサーバーより削除されます。

- 新しい XA データソースを追加するには、「管理インターフェースによる XA データソースの作成」 を参照してください。

バグの報告

6.4.4. XA リカバリー

6.4.4.1. XA リカバリーモジュール

各 XA リソースにはその設定に関連付けられたリカバリーモジュールが必要です。リカバリーモジュールはクラス **com.arjuna.ats.jta.recovery.XAResourceRecovery** を拡張する必要があります。

JBoss EAP 6 は、JDBC および JMS XA リソースのリカバリーモジュールを提供します。このようなタイプのリソースでは、リカバリーモジュールは自動的に登録されます。カスタムモジュールを使用する必要がある場合は、データソースに登録できます。

バグの報告

6.4.4.2. XA リカバリーモジュールの設定

ほとんどの JDBC および JMS リソースでは、リカバリーモジュールはリソースに自動的に関連付けられます。この場合は、リカバリーモジュールがリソースに接続してリカバリーを実行することを許可するオプションのみを設定する必要があります。

JDBC または JMS でないカスタムリソースの場合は、サポートされる設定について Red Hat グローバルサポートサービスにお問い合わせください。

これらの設定属性はデータソースの作成時または作成後に設定できます。Web ベースの管理コンソールまたはコマンドライン管理 CLI を使用して設定できます。XA データソースの設定に関する一般的な情報は、「管理インターフェースによる XA データソースの作成」および「管理インターフェースによる XA データソースの編集」を参照してください。

一般的なデータソース設定属性と、特定のデータベースベンダーに関連する設定詳細については、以下の表を参照してください。

表6.2 一般的な設定属性

属性	説明
recovery-username	リソースに接続してリカバリー行うためにリカバリーモジュールが使用する必要があるユーザー名。
recovery-password	リソースに接続してリカバリーを行うためにリカバリーモジュールが使用する必要があるパスワード。
recovery-security-domain	リカバリーを行う目的でリカバリーモジュールがリソースに接続するために使用するセキュリティードメイン。
recovery-plugin-class-name	カスタムのリカバリーモジュールを使用する必要がある場合は、この属性をモジュールの完全修飾クラス名に設定します。モジュールはクラス **com.arjuna.ats.jta.recovery.XAResourceRecovery** を拡張する必要があります。
recovery-plugin-properties	プロパティーを設定する必要があるカスタムのリカバリーモジュールを使用する場合は、この属性をプロパティーのコンマ区切りの *key=value* ペアのリストに設定します。

注記

表6.2「一般的な設定属性」で使用される名前は、データソースが CLI で設定されている場合に使用されるパラメーターです。XML 設定ファイルで使用される名前とは異なる場合があります。

ベンダー固有の設定情報

ここでは、特定のデータベースが JBoss EAP トランザクションマネージャーによって管理される XA トランザクションで連携するのに必要な特定の設定について説明します。詳細は、特定のデータベースのドキュメントを参照してください。

Oracle

Oracle データソースが不適切に設定された場合は、ログ出力に次のようなエラーが示されることがあります。

例6.11 誤った設定エラー

```
WARN  [com.arjuna.ats.jta.logging.loggerI18N]
[com.arjuna.ats.internal.jta.recovery.xarecovery1] Local XARecoveryModule.xaRecovery  got
XA exception javax.transaction.xa.XAException, XAException.XAER_RMERR
```

このエラーを解決するには、**recovery-username** で設定した Oracle ユーザーがリカバリーに必要なテーブルにアクセスできるようにします。以下の SQL ステートメントは、Oracle バグ5945463 にパッチが適用された Oracle 11g または Oracle 10g R2 インスタンスに対する適切な付与を示しています。

例6.12 付与の設定

```
GRANT SELECT ON sys.dba_pending_transactions TO recovery-username;
GRANT SELECT ON sys.pending_trans$ TO recovery-username;
GRANT SELECT ON sys.dba_2pc_pending TO recovery-username;
GRANT EXECUTE ON sys.dbms_xa TO recovery-username;
```

11g より前の Oracle 11 バージョンを使用している場合は、最終的な **EXECUTE** ステートメントを以下のように変更します。

```
GRANT EXECUTE ON sys.dbms_system TO recovery-username;
```

PostgreSQL および Postgres Plus Advanced Server

PostgreSQL が XA トランザクションを処理できるようにするには、設定パラメーター **max_prepared_transactions** を 0 よりも大きな値に変更します。

MySQL

特別な設定は必要ありません。詳細は MySQL のドキュメントを参照してください。

IBM DB2

特別な設定は必要ありません。詳細は IBM DB2 のドキュメントを参照してください。

Sybase

Sybase は、XA トランザクションがデータベース上で有効であることを想定します。XA トランザクションはデータベース設定が正しくないと動作しません。**xact** コーディネーションを有効 にすると、Adaptive Server トランザクションコーディネーションサービスを有効または無効にします。このパラメーターを有効にすると、リモート Adaptive Server データのアップデートが、確実に元のトラザクションでコミットまたはロールバックされるようになります。トラザクションコーディネーションを有効にするには、以下を使用します。

```
sp_configure 'enable xact coordination', 1
```

.

MSSQL

詳細は Microsoft ドキュメントを参照してください。また、開始点として参照 http://msdn.microsoft.com/en-us/library/aa342335.aspx することもできます。

ベンダー固有の問題とその結果

ここでは、XA トランザクションの処理に現在既知のデータベース固有の問題を説明します。これらの問題は、特定の EAP バージョンで現在対応しているデータベースバージョンおよび jdbc ドライバーを指します。

Oracle

データベースに関連する既知の問題はありません。

PostgreSQL

- コミットメソッドプロトコルの呼び出し中にエラーが発生した場合、JDBC ドライバーは XAER_RMERR を返します。データベースはこのエラーコードを返し、トランザクションをデータベース側 でインダウト 状態のままにします。正しい戻りコードは XAER_RMFAIL または XAER_RETRY である必要があります。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/236。これにより、トランザクションは EAP 側で **Heuristic** 状態のままになり、データベースでロックが保持されるため、ユーザーの介入が必要になります。

- jdbc ドライバーによって返される誤ったエラーコードにより、場合によってはデータの不整合が発生する可能性があります。詳細は、を参照して https://developer.jboss.org/thread/251537 ください。 https://github.com/pgjdbc/pgjdbc/issues/236

Postgres Plus Advanced Server

- コミットメソッドプロトコルの呼び出し中にエラーが発生した場合、JDBC ドライバーは XAER_RMERR を返します。データベースはこのエラーコードを返し、トランザクションをデータベース側 でインダウト 状態のままにします。正しい戻りコードは XAER_RMFAIL または XAER_RETRY である必要があります。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/236。これにより、トランザクションは EAP 側で **Heuristic** 状態のままになり、データベースでロックが保持されるため、ユーザーの介入が必要になります。

- jdbc ドライバーによって返される誤ったエラーコードにより、場合によってはデータの不整合が発生する可能性があります。詳細は以下を参照してください。 https://developer.jboss.org/thread/251537

- 同じ XID に対して **XAResource.rollback** が呼び出されると、**XAException** がスローされます。同じ XID に対するロールバックを複数回呼び出しても JTS 仕様に準拠し、**XAException** はスローされません。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/78。

MySQL

MySQL は XA トランザクションを処理できません。クライアントが MySQL を切断されると、このようなトランザクションに関する情報がすべて失われます。詳細はを参照してください http://bugs.mysql.com/bug.php?id=12161。

IBM DB2

データベースに関連する既知の問題はありません。

Sybase

- コミットメソッドプロトコルの呼び出し中にエラーが発生した場合、JDBC ドライバーは XAER_RMERR を返します。データベースはこのエラーコードを返し、トランザクションをデータベース側 でインダウト 状態のままにします。正しい戻りコードはXAER_RMFAIL またはXAER_RETRY である必要があります。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/236。これにより、トランザクションは EAP 側で **Heuristic** 状態のままになり、データベースでロックが保持されるため、ユーザーの介入が必要になります。

- 同じ XID に対して **XAResource.rollback** が呼び出されると、**XAException** がスローされます。同じ XID に対するロールバックを複数回呼び出しても JTS 仕様に準拠し、**XAException** はスローされません。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/78。

MSSQL

- コミットメソッドプロトコルの呼び出し中にエラーが発生した場合、JDBC ドライバーは XAER_RMERR を返します。データベースはこのエラーコードを返し、トランザクションをデータベース側 でインダウト 状態のままにします。正しい戻りコードはXAER_RMFAIL またはXAER_RETRY である必要があります。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/236。これにより、トランザクションは EAP 側で **Heuristic** 状態のままになり、データベースでロックが保持されるため、ユーザーの介入が必要になります。

  この問題は、の Microsoft Connect サイトで報告されてい https://connect.microsoft.com/SQLServer/feedback/details/1207381/jdbc-driver-is-not-xa-compliant-in-case-of-connection-failure-error-code-xaer-rmerr-is-returned-instead-of-xaer-rmretry ます。

- 同じ XID に対して **XAResource.rollback** が呼び出されると、**XAException** がスローされます。同じ XID に対するロールバックを複数回呼び出しても JTS 仕様に準拠し、**XAException** はスローされません。詳細はを参照してください https://github.com/pgjdbc/pgjdbc/issues/78。

- 同じ XID の **XAResource.rollback** の呼び出しにより、サーバーログファイルに以下のメッセージが記録されます。

> **WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an error from resource ... in transaction ...: java.lang.NullPointerException**

Messaging

IBM Web Warehouse

- JTS を使用する場合、定期的なリカバリー時に同じ XID に対するロールバックの 2 回目の呼び出しにより、エラーがログファイルに記録される可能性があります。同じ XID に対する 2 回目のロールバックは JTS の構成であり、無視できます。

RAR がこのような XID が分からない場合は、XAER_NOTA の戻りコードが予想されます。ただし、IBM Web Warehouse は誤った戻りコード XAER_RMFAIL を返す可能性があります。

```
The method 'xa_rollback' has failed with errorCode '-7' due to the resource being closed.'
```

ActiveMQ

- コミットメソッドプロトコルの呼び出し中にエラーが発生した場合（ネットワークの切断など）は、XAER_RMERR を返します。リソースアダプターはこのエラーコードをトランザクションマネージャーに戻し、メッセージブローカー側でトランザクションがインダウト 状態のままになります。この動作は XA 仕様に対して行われ、正しい戻りコードは XAER_RMFAIL または XAER_RETRY である必要があります。誤ったエラーコードにより、場合によってはデータの不整合が発生する可能性があります。詳細は、を参照して https://developer.jboss.org/thread/251537 ください。

```
WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an XA error: ARJUNA016099: Unknown error code:0 from resource ...  in transaction ...: javax.transaction.xa.XAException: Transaction ...  has not been started.
```

TIBCO

- JTS を使用する場合、定期的なリカバリー時に同じ XID に対するロールバックの 2 回目の呼び出しにより、エラーがログファイルに記録される可能性があります。同じ XID に対する 2 回目のロールバックは JTS の構成であり、無視できます。

```
WARN [com.arjuna.ats.jtax] ...: XAResourceRecord.rollback caused an XA error: XAException.XAER_RMFAIL from resource ... in transaction ...: javax.transaction.xa.XAException
```

バグの報告

6.5. データソースセキュリティー

6.5.1. データソースセキュリティー

データソースセキュリティーとは、データソース接続のパスワードを暗号化したり分かりにくくすることを言います。これらのパスワードはプレーンテキストで設定ファイルに保存できますが、セキュリティーリスクが高くなります。

データソースセキュリティーの推奨ソリューションは、セキュリティードメインまたはパスワード vault を使用することです。以下には、各メソッドの例が含まれています。詳細は、セキュリティー『アーキテクチャー』およびその他の JBoss EAP セキュリティードキュメントを参照してください。

例6.13 セキュリティードメインの例

```
<security-domain name="DsRealm" cache-type="default">
 <authentication>
  <login-module code="ConfiguredIdentity" flag="required">
   <module-option name="userName" value="sa"/>
```

```
        <module-option name="principal" value="sa"/>
        <module-option name="password" value="sa"/>
      </login-module>
    </authentication>
  </security-domain>
```

DsRealm ドメインはデータソースによって参照されます。

```
<datasources>
  <datasource jndi-name="java:jboss/datasources/securityDs"
    pool-name="securityDs">
    <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <new-connection-sql>select current_user()</new-connection-sql>
      <security>
        <security-domain>DsRealm</security-domain>
      </security>
    </datasource>
</datasources>
```

注記

セキュリティードメインが複数のデータソースと使用される場合は、セキュリティードメインでキャッシュを無効にする必要があります。これは、**cache-type** 属性の値を **none** に設定するか、この属性を削除します。ただし、キャッシュが必要な場合は、各データソースに個別のセキュリティードメインを使用する必要があります。

例6.14 パスワード vault の例

```
<security>
  <user-name>admin</user-name>

<password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEtMWNlMDMyNDdmNml2TElORV9CUkVBS3ZhdWx0}</password>
</security>
```

バグの報告

6.6. データベース接続の検証

6.6.1. データベース接続検証設定の指定

概要

データベースのメンテナンス、ネットワークの問題、またはその他の障害により、JBoss EAP 6 がデータベースへの接続を失うことがあります。データベース接続の検証は、サーバー設定ファイルの **<datasource>** セクション内の **<validation >** 要素を使用して有効にします。以下の手順に従ってデータソースを設定し、JBoss EAP 6 でデータベース接続検証を有効にします。

手順6.12 データベース接続検証設定の指定

1. *検証方法の選択*
   以下のいずれかの検証方法を選択します。

   - *<validate-on-match>true</validate-on-match>*
     *< **validate-on-match** > オプションを**true** に設定すると、次の手順で指定された検証メカニズムを使用して接続プールからチェックアウトされるたびにデータベース接続が検証されます。*

     *接続が有効でない場合は、警告がログに書き込まれ、プール内の次の接続が取得されます。このプロセスは、有効な接続が見つかるまで続行します。プール内の各接続を繰り返し処理しない場合は、< **use-fast-fail** > オプションを使用できます。有効な接続がプールにない場合は、新しい接続が作成されます。接続の作成に失敗すると、例外が要求元アプリケーションに返されます。*

     この設定により、最も早いリカバリーが実現されますが、データベースへの負荷が最も大きくなります。ただし、これは、パフォーマンスを気にする必要がない場合は最も安全な方法です。

   - *<background-validation>true</background-validation>*
     *< **background-validation** > オプションを**true** に設定すると、< **background-validation-millis** > 値と組み合わせて、バックグラウンド検証が実行される頻度を決定します。< **background-validation-millis** > パラメーターのデフォルト値は 0 ミリ秒で、デフォルトでは無効になっています。この値は、< **idle-timeout-minutes>** 設定と同じ値に設定しないでください。*

     *特定のシステムのoptum < **background-validation-millis** > 値を決定するためのバランス動作です。値が小さいほどプールの検証頻度が高くなり、より迅速に無効な接続がプールから削除されます。ただし、値が小さいほどデータベースリソースが多くなります。値が大きいほど接続検証チェックの頻度が低くなり、データベースリソースの使用量が減りますが、無効な接続が検出されない期間が長くなります。*

注記

*< **validate-on-match** > オプションを**true** に設定すると、< **background-validation>** オプション を **false** に設定する必要があります。その逆も同様です。< **background-validation>** オプション を **true** に設定すると、< **validate-on-match>** オプション を **false** に設定する必要があります。*

2. *検証メカニズムの選択*
   以下のいずれかの検証メカニズムを選択します。

   - *<valid-connection-checker> クラス名の指定*
     これは、使用中の特定の pid に対して最適化されているため、推奨されるメカニズムです。JBoss EAP 6 は以下の接続チェッカーを提供します。

     - *org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker*
     - *org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker*
     - *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecke*
     - *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker*
     - *org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker*
     - *org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker*

- *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker*

- *<check-valid-connection-sql> の SQL の指定*
  接続を検証するために使用する SQL ステートメントを提供します。

  以下は、Oracle 用接続を検証するために SQL ステートメントをどのように指定するかの例です。

  ```
  <check-valid-connection-sql>select 1 from dual</check-valid-connection-sql>
  ```

  MySQL または PostgreSQL の場合は、以下の SQL ステートメントを指定する必要があります。

  ```
  <check-valid-connection-sql>select 1</check-valid-connection-sql>
  ```

3. *クラス名の <exception-sorter> 設定*
   例外が致命的とマークされた場合、接続はトランザクションに参加していてもすぐに閉じられます。致命的な接続例外を適切に検出およびクリーンアップするには、例外ソータークラスオプションを使用します。JBoss EAP 6 は以下の例外ソーターを提供します。

   - *org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.informix.InformixExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter*
   - *org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter*

バグの報告

6.7. データソース設定

6.7.1. データソースのパラメーター

表6.3 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

パラメーター	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。

パラメーター	説明
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。
spy	JDBC レイヤーで **スパイ** 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 **jboss.jdbc.spy** も logging サブシステムのログレベル **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。
security	セキュリティー設定である子要素が含まれます。表6.8「セキュリティーパラメーター」を参照してください。
検証	検証設定である子要素が含まれます。表6.9「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6.10「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6.11「ステートメントのパラメーター」を参照してください。

表6.4 非 XA データソースのパラメーター

パラメーター	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	メソッド **Driver.connect(url,props)** 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から取得され、値は要素のコンテンツから取得されます。
pool	プーリング設定である子要素が含まれます。表 6.6「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6.5 XA データソースのパラメーター

パラメーター	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* で指定します。setter メソッドが存在する場合は、**set*Name*** の形式で、**set*Name*(*value*)** の形式で setter メソッドを呼び出すことでプロパティーが設定されます。
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。使用できる形式は *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 6.6「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6.7「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 6.12「リカバリーパラメーター」を参照してください。

表6.6 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

パラメーター	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。デフォルトは **false** です。
use-strict-min	**min-pool-size** に達すると、アイドル接続スキャンが、追加の接続を明確化するために厳密に停止するかどうか。デフォルト値は **false** です。
flush-strategy	エラーの場合にプールがフラッシュされるかどうか。有効な値は以下のとおりです。 ● FailingConnectionOnly ● IdleConnections ● EntirePool デフォルトは **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6.7 XA プールパラメーター

パラメーター	説明
is-same-rm-override	**javax.transaction.xa.XAResource.isSameRM(XAResource)** クラスが **true** または **false** を返すかどうか。
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。
no-tx-separate-pools	コンテキストごとに個別のサブプールを作成するかどうか。これは、JTA トランザクションの内部と外部の両方で XA 接続を使用できない Oracle データソースに必要です。 このオプションを使用すると、実際のプールが 2 つ作成されるため、合計プールサイズが **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。

パラメーター	説明
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。

表6.8 セキュリティーパラメーター

パラメーター	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	認証を処理する JAAS security-manager の名前が含まれます。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6.9 検証パラメーター

パラメーター	説明
valid-connection-checker	接続を検証する **org.jboss.jca.adaptors.jdbc.ValidConnectionChecker** メソッドを提供するインターフェース **SQLException.isValidConnection(Connection e)** の実装。接続が破棄されると例外が発生します。これにより、パラメーター **check-valid-connection-sql** が上書きされます（存在する場合）。
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、管理接続がプールから取得されたときに呼び出される場合があります。
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。 **validate-on-match** に「true」を指定することは、通常 **background-validation** に「true」を指定するとともには実行されません。使用前にクライアントが接続を検証する必要がある場合に、validate-**on-match** が必要です。このパラメーターはデフォルトで false です。

パラメーター	説明
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。バックグラウンド検証は、**validate-on-match** とともに使用されていない場合のパフォーマンスの最適化です。**validate-on-match** が true の場合、**background-validation** を使用すると冗長チェックが生じる可能性があります。バックグラウンド検証では、使用のために不適切な接続がクライアントに付与される可能性が開かれます（検証スキャンにかかる時間とクライアントに渡す前に接続が悪い場合）、クライアントアプリケーションはこ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の試行で接続の割り当てに失敗します。デフォルトは **false** です。
stale-connection-checker	ブール値の **org.jboss.jca.adapters.jdbc.StaleConnectionChecker** メソッドを提供する **isStaleConnection(SQLException e)** のインスタンス。このメソッドが **true** を返すと、例外は **org.jboss.jca.adapters.jdbc.StaleConnectionException** のサブクラスである **SQLException** でラップされます。
exception-sorter	ブール値の **org.jboss.jca.adapters.jdbc.ExceptionSorter** メソッドを提供する **isExceptionFatal(SQLException e)** 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にブロードキャストされているかどうかを確認します。

表6.10 タイムアウトパラメーター

パラメーター	説明
use-try-lock	**tryLock()** の代わりに **lock()** を使用します。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、タイムアウトする前に設定された秒数間ロックの取得を試みます。デフォルトは **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。

パラメーター	説明
blocking-timeout-millis	接続の待機中にブロックする最大時間（ミリ秒単位）。この時間を過ぎると、例外が発生します。このブロックは、接続許可の待機中にのみブロックし、新規接続の作成に長時間要している場合は例外は発生しません。デフォルトは 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）。指定がない場合、デフォルトは **30** 分になります。実際の最大時間は、idleRemover スキャン時間によって異なります。これは、プールの最小 **idle-timeout-minutes** の半分です。
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。デフォルトは **false** です。
query-timeout	クエリーのタイムアウト（秒単位）。デフォルトではタイムアウトはありません。
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時間（ミリ秒単位）。デフォルトは 5000（5 秒）です。
xa-resource-timeout	ゼロ以外の値はメソッド **XAResource.setTransactionTimeout** に渡されます。

表6.11 ステートメントのパラメーター

パラメーター	説明

パラメーター	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。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は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。これがデフォルトです。
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	アプリケーションに提供されたラッパーがアプリケーションコードによって閉じられたときに、JBoss EAP が基盤の物理ステートメントを閉じたり終了せずに、キャッシュするかどうか。デフォルトは **false** です。

表6.12 リカバリーパラメーター

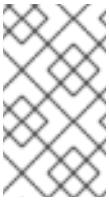
パラメーター	説明
recover-credential	リカバリーに使用するユーザー名とパスワードのペア、あるいはセキュリティードメイン。
recover-plugin	リカバリーに使用される **org.jboss.jca.core.spi.recoveryRecoveryPlugin** クラスの実装。

バグの報告

6.7.2. データソース接続 URL

表6.13 データソース接続 URL

データソース	接続 URL
PostgreSQL	jdbc:postgresql://*SERVER_NAME*:*PORT*/*DATABASE_NAME*

データソース	接続 URL
MySQL	jdbc:mysql://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
Oracle	jdbc:oracle:thin:@*ORACLE_HOST*:*PORT*:*ORACLE_SID*
IBM DB2	jdbc:db2://*SERVER_NAME*:*PORT*/*DATABASE_NAME*
Microsoft SQLServer	jdbc:sqlserver://*SERVER_NAME*:*PORT*;DatabaseName=*DATABASE_NAME*  **注記** jdbc:microsoft:sqlserver://*SERVER_NAME*:*PORT*;DatabaseName=*DATABASE_NAME* テンプレートは、新しいデータベースでは機能しません。

バグの報告

6.7.3. データソースの拡張

データソースデプロイメントでは、JDBC リソースアダプターの複数の拡張機能を使用して接続の検証を改善し、例外が接続を再確立すべきかどうかを確認できます。これらの拡張機能は、以下のとおりです。

表6.14 データソースの拡張

データソースの拡張	設定パラメーター	説明
org.jboss.jca.adapters.jdbc.spi.ExceptionSorter	<exception-sorter>	SQLException が発生した接続にとってこの例外は致命的かどうかを確認します。
org.jboss.jca.adapters.jdbc.spi.StaleConnectionChecker	<stale-connection-checker>	古くなった SQLExceptions をラップします。**org.jboss.jca.adapters.jdbc.StaleConnectionException**
org.jboss.jca.adapters.jdbc.spi.ValidConnection	<valid-connection-checker>	アプリケーションによる接続の使用が有効であるかどうかを確認します。

JBoss EAP 6 は、これらの拡張の実装を、サポートされるいくつかのデータベース用に提供します。

拡張の実装

汎用

- *org.jboss.jca.adapters.jdbc.extensions.novendor.NullExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.novendor.NullStaleConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.novendor.NullValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker*

PostgreSQL

- *org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker*

MySQL

- *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLReplicationValidConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker*

IBM DB2

- *org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker*

Sybase

- *org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker*

Microsoft SQLServer

- *org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker*

Oracle

- *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter*
- *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker*
- *org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker*

バグの報告

6.7.4. データソース統計の表示

*以下のコマンドが適切に変更されたバージョンを使用すると、**jdbc** と **pool** の両方の定義されたデータソースから統計を表示できます。*

例6.15 ドメインモードの場合の CLI:

*環境に応じて **/host=master/server=server-one** および **data-source=ExampleDS** を変更します。*

```
[domain@localhost:9999 /]  /host=master/server=server-one/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => "0",
        "AvailableCount" => "20",
        "AverageBlockingTime" => "0",
        "AverageCreationTime" => "0",
        "CreatedCount" => "0",
        "DestroyedCount" => "0",
        "MaxCreationTime" => "0",
        "MaxUsedCount" => "0",
        "MaxWaitTime" => "0",
        "TimedOut" => "0",
        "TotalBlockingTime" => "0",
        "TotalCreationTime" => "0"
    }
}
```

例6.16 スタンドアロンモードの CLI:

*環境に応じて **data-source=ExampleDS** を変更します。*

```
[standalone@localhost:9999 /]  /subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
{
    "outcome" => "success",
    "result" => {
        "ActiveCount" => "0",
        "AvailableCount" => "20",
        "AverageBlockingTime" => "0",
        "AverageCreationTime" => "0",
        "CreatedCount" => "0",
        "DestroyedCount" => "0",
        "MaxCreationTime" => "0",
        "MaxUsedCount" => "0",
        "MaxWaitTime" => "0",
        "TimedOut" => "0",
        "TotalBlockingTime" => "0",
        "TotalCreationTime" => "0"
    }
}
```

注記

統計はすべてランタイムのみの情報で、デフォルトは ***false*** *であるため、必ず* ***include-runtime=true*** *引数を指定してください。*

バグの報告

6.7.5. データソースの統計

主要統計

サポートされるデータソースの主要統計は下表のとおりです。

表6.15 主要統計

名前	説明
ActiveCount	アクティブな接続の数。各接続はアプリケーションによって使用されているか、プールで使用可能な状態であるかのいずれかになります。
AvailableCount	プールの使用可能な接続の数。
AverageBlockingTime	プールの排他ロックの取得をブロックするために費やされた平均時間。値はミリ秒単位です。
AverageCreationTime	接続の作成に費やされた平均時間。値はミリ秒単位です。
CreatedCount	作成された接続の数。
DestroyedCount	破棄された接続の数。
InUseCount	現在使用中の接続の数。
MaxCreationTime	接続の作成にかかった最大時間。値はミリ秒単位です。
MaxUsedCount	使用される接続の最大数。
MaxWaitCount	同時に接続を待機する要求の最大数。
MaxWaitTime	プールの排他ロックの待機に費やされた最大時間。
TimedOut	タイムアウトした接続の数。
TotalBlockingTime	プールの排他ロックの待機に費やされた合計時間。値はミリ秒単位です。
TotalCreationTime	接続の作成に費やされた合計時間。値はミリ秒単位です。

JDBC の統計

サポートされるデータソースの JDBC 統計は下表のとおりです。

表6.16 JDBC の統計

名前	説明
PreparedStatementCacheAccessCount	ステートメントキャッシュがアクセスされた回数。
PreparedStatementCacheAddCount	ステートメントキャッシュに追加されたステートメントの数。
PreparedStatementCacheCurrentSize	ステートメントキャッシュに現在キャッシュされた準備済みおよび呼び出し可能ステートメントの数。
PreparedStatementCacheDeleteCount	キャッシュから破棄されたステートメントの数。
PreparedStatementCacheHitCount	キャッシュからのステートメントが使用された回数。
PreparedStatementCacheMissCount	ステートメント要求がキャッシュのステートメントと一致しなかった回数。

*以下のコマンドの適切に変更されたバージョンを使用すると、コアおよび**JDBC** の統計を有効にできます。*

- ***/subsystem=datasources/data-source=ExampleDS/statistics=pool:write-attribute(name=statistics-enabled,value=true)***

  /subsystem=datasources/data-source=ExampleDS/statistics=jdbc:write-attribute(name=statistics-enabled,value=true)

バグの報告

6.8. データソース例

6.8.1. PostgreSQL データソースの例

PostgreSQL のデータソース設定例は以下のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.17 PostSQL データソースの設定

```
<datasources>
  <datasource jndi-name="java:jboss/PostgresDS" pool-name="PostgresDS">
    <connection-url>jdbc:postgresql://localhost:5432/postgresdb</connection-url>
    <driver>postgresql</driver>
    <security>
      <user-name>admin</user-name>
```

```
      <password>admin</password>
    </security>
    <validation>
     <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChec
ker"></valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
</exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="postgresql" module="org.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 PostgreSQL データソースの **module.xml** ファイルの例は次のとおりです。

例6.18 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.2. PostgreSQL XA データソースの例

PostgreSQL XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.19 PostSQL XA datasource

```
<datasources>
  <xa-datasource jndi-name="java:jboss/PostgresXADS" pool-name="PostgresXADS">
    <driver>postgresql</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="PortNumber">5432</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">postgresdb</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
```

```
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChec
ker">
      </valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter">
      </exception-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="postgresql" module="org.postgresql">
      <xa-datasource-class>org.postgresql.xa.PGXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 PostgreSQL XA データソースの **module.xml** *ファイルの例は次のとおりです。*

例6.20 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="org.postgresql">
  <resources>
    <resource-root path="postgresql-9.1-902.jdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.3. MySQL データソースの例

MySQL データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.21 MySQL データソースの設定

```
<datasources>
  <datasource jndi-name="java:jboss/MySqlDS" pool-name="MySqlDS">
    <connection-url>jdbc:mysql://mysql-localhost:3306/jbossdb</connection-url>
    <driver>mysql</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
```

```
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter">
</exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 MySQL データソースの **module.xml** *ファイルの例は次のとおりです。*

例6.22 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.4. MySQL XA データソースの例

MySQL XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.23 MySQL XA データソース

```
<datasources>
  <xa-datasource jndi-name="java:jboss/MysqlXADS" pool-name="MysqlXADS">
  <driver>mysql</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mysqldb</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
```

```
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mysql.MySQLExceptionSorter">
</exception-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="mysql" module="com.mysql">
      <xa-datasource-class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-
datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 MySQL XA データソースの **module.xml** *ファイルの例は次のとおりです。*

例6.24 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.0.8-bin.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.5. Oracle データソースの例

ORACLE のデータソース

バージョン 10.2 以前の Oracle データソースでは、トランザクション以外の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するため、<no-tx-separate-pools/> パラメーターが必要でした。特定のアプリケーションでこのパラメーターが不要になる場合があります。

Oracle のデータソース設定例は以下のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.25 Oracle データソースの設定

```
<datasources>
  <datasource jndi-name="java:/OracleDS" pool-name="OracleDS">
    <connection-url>jdbc:oracle:thin:@localhost:1521:XE</connection-url>
    <driver>oracle</driver>
```

```
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker">
</valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker">
</stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exception-
sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
    </driver>
  </drivers>
</datasources>
```

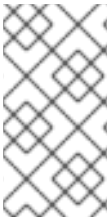
上記 Oracle データソースの **module.xml** *ファイルの例は次のとおりです。*

例6.26 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

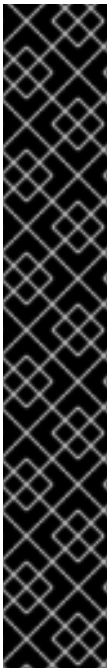
バグの報告

6.8.6. Oracle XA データソースの例



ORACLE のデータソース

バージョン 10.2 以前の Oracle データソースでは、トランザクション以外の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するため、<no-tx-separate-pools/> パラメーターが必要でした。特定のアプリケーションでこのパラメーターが不要になる場合があります。



重要

*Oracle XA データソースにアクセスするユーザーは、以下の設定を適用しないと XA リカバリーが適切に操作しません。値 **user** は、JBoss から Oracle に接続するために定義されたユーザーです。*

- *GRANT SELECT ON sys.dba_pending_transactions TO user;*
- *GRANT SELECT ON sys.pending_trans$ TO user;*
- *GRANT SELECT ON sys.dba_2pc_pending TO user;*
- *GRANT EXECUTE ON sys.dbms_xa TO user; (パッチ済みの Oracle 10g R2、または Oracle 11g を使用する場合)*

  または、

  GRANT EXECUTE ON sys.dbms_system TO user; (パッチされていない 11g 以前のバージョンの Oracle を使用する場合)

Oracle XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.27 Oracle XA データソース

```
<datasources>
  <xa-datasource jndi-name="java:/XAOracleDS" pool-name="XAOracleDS">
    <driver>oracle</driver>
    <xa-datasource-property name="URL">jdbc:oracle:oci8:@tc</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
      <no-tx-separate-pools />
    </xa-pool>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleValidConnectionChecker">
</valid-connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleStaleConnectionChecker">
</stale-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExceptionSorter"></exception-
sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="oracle" module="com.oracle">
      <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
```

```
    </driver>
  </drivers>
</datasources>
```

上記 Oracle XA データソースの **module.xml** ファイルの例は次のとおりです。

例6.28 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.7. Microsoft SQLServer のデータソースの例

Microsoft SQLServer データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.29 SQLServer データソースの設定

```
<datasources>
  <datasource jndi-name="java:/MSSQLDS" pool-name="MSSQLDS">
    <connection-
url>jdbc:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter">
</exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
  </datasources>
```

上記 Microsoft SQLServer データソースの **module.xml** ファイルの例は次のとおりです。

例6.30 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.8. Microsoft SQLServer XA のデータソースの例

Microsoft SQLServer XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.31 SQLserver XA datasource

```
<datasources>
  <xa-datasource jndi-name="java:/MSSQLXADS" pool-name="MSSQLXADS">
    <driver>sqlserver</driver>
    <xa-datasource-property name="ServerName">localhost</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mssqldb</xa-datasource-property>
    <xa-datasource-property name="SelectMethod">cursor</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLExceptionSorter">
</exception-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sqlserver" module="com.microsoft">
      <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-
datasource-class>
```

```
    </driver>
  </drivers>
</datasources>
```

上記 Microsoft SQLServer XA データソースの **module.xml** ファイルの例は次のとおりです。

例6.32 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.microsoft">
  <resources>
    <resource-root path="sqljdbc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.9. IBM DB2 データソースの例

IBM DB2 のデータソース設定例は以下のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.33 IBM DB2 データソースの設定

```
<datasources>
  <datasource jndi-name="java:/DB2DS" pool-name="DB2DS">
    <connection-url>jdbc:db2:ibmdb2db</connection-url>
    <driver>ibmdb2</driver>
    <pool>
      <min-pool-size>0</min-pool-size>
      <max-pool-size>50</max-pool-size>
    </pool>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"></valid-
connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stale-
connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-sorter>
    </validation>
  </datasource>
```

```
    <drivers>
      <driver name="ibmdb2" module="com.ibm">
      </driver>
    </drivers>
  </datasources>
```

上記 IBM DB2 データソースの **module.xml** ファイルの例は次のとおりです。

例6.34 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.10. IBM DB2 XA のデータソースの例

IBM DB2 XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.35 IBM DB2 XA データソースの設定

```
<datasources>
  <xa-datasource jndi-name="java:/DB2XADS" pool-name="DB2XADS">
    <driver>ibmdb2</driver>
    <xa-datasource-property name="DatabaseName">ibmdb2db</xa-datasource-property>
    <xa-datasource-property name="ServerName">hostname</xa-datasource-property>
    <xa-datasource-property name="PortNumber">port</xa-datasource-property>
    <xa-datasource-property name="DriverType">4</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ValidConnectionChecker"></valid-
connection-checker>
      <stale-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2StaleConnectionChecker"></stale-
connection-checker>
```

```
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.db2.DB2ExceptionSorter"></exception-
sorter>
    </validation>
    <recovery>
      <recover-plugin class-
name="org.jboss.jca.core.recovery.ConfigurableRecoveryPlugin">
        <config-property name="EnableIsValid">false</config-property>
        <config-property name="IsValidOverride">false</config-property>
        <config-property name="EnableClose">false</config-property>
      </recover-plugin>
    </recovery>
  </xa-datasource>
  <drivers>
    <driver name="ibmdb2" module="com.ibm">
      <xa-datasource-class>com.ibm.db2.jcc.DB2XADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 IBM DB2 XA データソースの **module.xml** *ファイルの例は次のとおりです。*

例6.36 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.ibm">
  <resources>
    <resource-root path="db2jcc4.jar"/>
    <resource-root path="db2jcc_license_cisuz.jar"/>
    <resource-root path="db2jcc_license_cu.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.11. Sybase データソースの例

Sybase データソースの設定例は以下のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.37 Sybase データソースの設定

```
<datasources>
  <datasource jndi-name="java:jboss/SybaseDB" pool-name="SybaseDB" enabled="true">
    <connection-url>jdbc:sybase:Tds:localhost:5000/DATABASE?
JCONNECT_VERSION=6</connection-url>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
```

```
    </security>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter">
</exception-sorter>
    </validation>
  </datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <datasource-class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 Sybase データソースの **module.xml** ファイルの例は次のとおりです。

例6.38 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

6.8.12. Sybase XA データソースの例

Sybase XA データソースの設定例は次のとおりです。データソースが有効になり、ユーザーが追加され、検証オプションが設定されます。

例6.39 Sybase XA データソースの設定

```
<datasources>
  <xa-datasource jndi-name="java:jboss/SybaseXADS" pool-name="SybaseXADS"
enabled="true">
    <xa-datasource-property name="NetworkProtocol">Tds</xa-datasource-property>
    <xa-datasource-property name="ServerName">myserver</xa-datasource-property>
    <xa-datasource-property name="PortNumber">4100</xa-datasource-property>
    <xa-datasource-property name="DatabaseName">mydatabase</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
```

```
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <validate-on-match>true</validate-on-match>
      <background-validation>false</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseValidConnectionChecker">
</valid-connection-checker>
      <exception-sorter class-
name="org.jboss.jca.adapters.jdbc.extensions.sybase.SybaseExceptionSorter">
</exception-sorter>
    </validation>
  </xa-datasource>
  <drivers>
    <driver name="sybase" module="com.sybase">
      <datasource-class>com.sybase.jdbc4.jdbc.SybDataSource</datasource-class>
      <xa-datasource-class>com.sybase.jdbc4.jdbc.SybXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記 Sybase XA データソースの ***module.xml*** *ファイルの例は次のとおりです。*

例6.40 module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.sybase">
  <resources>
    <resource-root path="jconn2.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

バグの報告

第7章 モジュールの設定

7.1. はじめに

7.1.1. モジュール

モジュールは、クラスローディングおよび依存関係管理に使用されるクラスの論理グループです。JBoss EAP 6 は、静的モジュールと動的モジュールと呼ばれる 2 種類のモジュールを識別します。ただし、この 2 つの間の唯一の違いは、パッケージ化方法です。

静的モジュール

静的モジュールは、アプリケーションサーバーの **EAP_HOME/modules/** ディレクトリーで事前定義されます。各サブディレクトリーは 1 つのモジュールを表し、設定ファイル (**module.xml**) と必要な JAR ファイルが含まれる **main/** サブディレクトリーを定義します。モジュールの名前は **module.xml** ファイルで定義されます。アプリケーションサーバーにより提供される API は、Java EE API や JBoss Logging などの他の API を含む静的モジュールとして提供されます。

例7.1 module.xml ファイルの例

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java-5.1.15.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
  </dependencies>
</module>
```

モジュール名 **com.mysql** は、**main/** サブディレクトリー名を除く、モジュールのディレクトリー構造と一致する必要があります。

JBoss EAP ディストリビューションで提供されるモジュールは、**EAP_HOME/modules** ディレクトリー内の **system** ディレクトリーにあります。このため、サードパーティーによって提供されるモジュールから分離されます。

JBoss EAP 6.1 以降では、Red Hat が提供するレイヤー化された製品も、**system** ディレクトリー内にそのモジュールをインストールします。

カスタム静的モジュールの作成は、同じサードパーティーライブラリーを使用する同じサーバー上に多くのアプリケーションがデプロイされる場合に役立ちます。これらのライブラリーを各アプリケーションとバンドルする代わりに、JBoss 管理者によってこれらのライブラリーが含まれるモジュールを作成およびインストールできます。アプリケーションは、カスタム静的モジュールで明示的な依存関係を宣言できます。

モジュールレイアウトごとに 1 つのディレクトリーを使用して、カスタムモジュールが **EAP_HOME/modules** ディレクトリーにインストールされるようにする必要があります。これにより、同梱されたバージョンではなく、**system** ディレクトリーに存在するカスタムバージョンのモジュールがロードされるようになります。このようにして、ユーザー提供のモジュールはシステムモジュールよりも優先されます。

JBOSS_MODULEPATH 環境変数を使用して JBoss EAP がモジュールを検索する場所を変更する場合、指定された場所の1 つで **system** サブディレクトリー構造を探します。システム 構造は、**JBOSS_MODULEPATH** で指定された場所のどこかに存在する必要があります。

動的モジュール

動的モジュールは、各 JAR または WAR デプロイメント（または EAR 内のサブデプロイメント）に対してアプリケーションサーバーによって作成およびロードされます。動的モジュールの名前は、デプロイされたアーカイブの名前に由来します。デプロイメントはモジュールとしてロードされるため、依存関係を設定し、他のデプロイメントで依存関係として使用することが可能です。

モジュールは必要な場合にのみロードされます。通常、モジュールは、明示的または暗黙的な依存関係があるアプリケーションがデプロイされる場合にのみロードされます。

バグの報告

7.1.2. グローバルモジュール

グローバルモジュールは、JBoss EAP 6 が各アプリケーションへの依存関係として提供するモジュールです。モジュールは、アプリケーションサーバーのグローバルモジュールの一覧に追加することでグローバルモジュールを作成できます。モジュールへの変更は必要ありません。

バグの報告

7.1.3. モジュールの依存性

モジュール依存関係は、あるモジュールが機能するために別のモジュールのクラスを必要とする宣言です。モジュールは、他のモジュールの依存関係を宣言できます。アプリケーションサーバーがモジュールをロードすると、モジュールクラスローダーはそのモジュールの依存関係を解析し、各依存関係のクラスをクラスパスに追加します。指定の依存関係が見つからない場合、モジュールはロードできません。

デプロイされたアプリケーション (JAR または WAR) は動的モジュールとしてロードされ、依存関係を利用して JBoss EAP 6 によって提供される API にアクセスします。

依存関係には明示的と暗黙的の 2 つのタイプがあります。

明示的な依存関係

明示的な依存関係は開発者が設定ファイルで宣言します。静的モジュールは、依存関係を **module.xml** ファイルに宣言できます。動的モジュールでは、デプロイメントの **MANIFEST.MF** または **jboss-deployment-structure.xml** デプロイメント記述子に依存関係を宣言できます。

明示的な依存関係は、オプションとして指定できます。オプションの依存関係をロードできなくても、モジュールのロードは失敗しません。ただし、依存関係が後で使用できるようになっても、モジュールのクラスパスには追加されません。依存関係はモジュールがロードされるときに利用可能である必要があります。

暗黙的な依存関係

暗黙的な依存関係は、デプロイメントで特定の条件またはメタデータが見つかると、アプリケーションサーバーによって自動的に追加されます。JBoss EAP 6 に同梱される Java EE 6 API は、デプロイメントで暗黙的な依存関係が検出されたときに追加されるモジュールの例になります。

特定の暗黙的な依存関係を除外するようにデプロイメントを設定することもできます。これは、**jboss-deployment-structure.xml** デプロイメント記述子ファイルで行います。これは、アプリケーション

サーバーが暗黙的な依存関係として追加しようとする特定バージョンのライブラリーをアプリケーションがバンドルする場合に一般的に行われます。

モジュールのクラスパスには、独自のクラスと、その直接の依存関係のクラスのみが含まれます。モジュールは1つの依存関係の依存関係クラスにはアクセスできませんが、暗黙的な依存関係のエクスポートを指定できます。ただし、モジュールは、明示的な依存関係をエクスポートするように指定できます。エクスポートした依存関係は、エクスポートするモジュールに依存するモジュールに提供されます。

> 例7.2 モジュールの依存関係
>
> モジュールAはモジュールBに依存し、モジュールBはモジュールCに依存します。モジュールAはモジュールBのクラスにアクセスでき、モジュールBはモジュールCのクラスにアクセスできます。以下の場合を除き、モジュールAはモジュールCのクラスにはアクセスできません。
>
> - モジュールAがモジュールCへの明示的な依存関係を宣言する場合。
> - または、モジュールBがモジュールBの依存関係をモジュールCでエクスポートする場合。

jboss-deployment-structure.xml デプロイメント記述子に関する詳細は、『**開発ガイド**』の「『**クラスローディングとモジュール**』」の章を参照してください。

バグの報告

7.1.4. サブデプロイメントクラスローダーの分離

エンタープライズアーカイブ(EAR)の各サブデプロイメントは独自のクラスローダーを持つ動的モジュールです。デフォルトでは、サブデプロイメントは他のサブデプロイメントのリソースにアクセスできます。

サブデプロイメントが他のサブデプロイメントのリソースにアクセスすることが許可されていない場合は、厳格なサブデプロイメントの分離を有効にできます。

バグの報告

7.2. すべてのデプロイメントを対象とするサブデプロイメントモジュール分離の無効化

このタスクでは、サーバー管理者がアプリケーションサーバーでサブデプロイメントモジュール分離を無効にする方法を説明します。これはすべてのデプロイメントに影響します。

> 警告
>
> このタスクでは、サーバーのXML設定ファイルを編集する必要があります。これを実行する前にサーバーを停止する必要があります。最終的なリリース管理ツールはこの種の設定をサポートするため、これは一時的なものです。

1. サーバーの停止

JBoss EAP 6 サーバーを停止します。

2. サーバー設定ファイルを開く
サーバー設定ファイルをテキストエディターで開きます。

このファイルは、管理対象ドメインまたはスタンドアロンサーバーの場合とは異なります。さらに、デフォルト以外の場所とファイル名を使用することもできます。デフォルトの設定ファイルは、管理対象ドメインとスタンドアロンサーバーの **domain/configuration/domain.xml** と **standalone/configuration/standalone.xml** です。

3. **EE** サブシステム設定の特定
設定ファイルで **EE** サブシステム設定要素を見つけます。設定ファイルの **<profile>** 要素には、複数のサブシステム要素が含まれます。**EE Subsystem** 要素には **urn:jboss:domain:ee:1.2** の名前空間があります。

```
<profile>

  ...

  <subsystem xmlns="urn:jboss:domain:ee:1.2" />

  ...
```

デフォルト設定には単一の自己終了タグがありますが、カスタム設定は、以下のような個別の開始タグと終了タグを持つことがあります (タグ間に別の要素が含まれることがあります)。

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ></subsystem>
```

4. 自己終了タグの置き換え (必要な場合)
EE サブシステム要素が単一の自己終了タグである場合は、以下のように適切な開始タグと終了タグで置き換えます。

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ></subsystem>
```

5. **ear-subdeployments-isolated** 要素の追加
ear-subdeployments-isolated 要素を **EE** サブシステム要素の子として追加し、以下のように **false** の内容を追加します。

```
<subsystem xmlns="urn:jboss:domain:ee:1.2" ><ear-subdeployments-isolated>false</ear-subdeployments-isolated></subsystem>
```

6. サーバーの起動
新しい設定で実行されるよう JBoss EAP 6 サーバーを再起動します。

結果

すべてのデプロイメントに対してサブデプロイメントモジュール分離が無効化された状態で、サーバーが実行されます。

バグの報告

7.3. すべてのデプロイメントへのモジュールの追加

このタスクは JBoss 管理者によるグローバルモジュールリストの定義方法を実証します。

前提条件

1. *グローバルモジュールとして設定するモジュールの名前を知っている必要があります。**JBoss EAP 6** に含まれる静的モジュールのリストは、「含まれるモジュール」 を参照してください。モジュールが別のデプロイメントにある場合は、「動的モジュールの名前付け」 を参照してモジュール名を判断してください。*

手順7.1 グローバルモジュールリストへのモジュールの追加

1. *管理コンソールへログインします。「管理コンソールへのログイン」*
2. ***EE Subsystem** パネルに移動します。*
   a. *コンソールの上部にある **Configuration** タブを選択します。*
   b. *ドメインモードの場合*
      i. *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。*
   c. *コンソールの左側にある **Subsystems** メニューを展開します。*
   d. *コンソールの左側にあるメニューから **Container → EE** を選択します。*
3. ***Subsystem Defaults** セクションで **Add** をクリックします。モジュールの作成ダイアログが表示されます。*
4. *モジュールの名前と任意でモジュールスロットを入力します。*
5. ***Save** をクリックして新しいグローバルモジュールを追加するか、**Cancel** をクリックして中断します。*
   - ***Save** をクリックすると、ダイアログが閉じられ、指定のモジュールがグローバルモジュールのリストに追加されます。*
   - ***Cancel** をクリックするとダイアログが閉じられ、変更は加えられません。*

結果

グローバルモジュールのリストに追加されたモジュールは各デプロイメントへの依存関係に追加されます。

バグの報告

7.4. カスタムモジュールの作成

*次の手順では、**JBoss EAP** サーバー上で実行されているすべてのアプリケーションがプロパティーファイルやその他のリソースを使用できるようにするために、カスタムモジュールを作成する方法について説明します。*

手順7.2 カスタムモジュールの作成

1. *モジュール/ ディレクトリー構造を作成し、これを追加します。*
   a. ***EAP_HOME/module** ディレクトリーにディレクトリー構造を作成し、ファイルと **JAR** が含まれるようにします。以下に例を示します。*

```
$ cd EAP_HOME/modules/
$ mkdir -p myorg-conf/main/properties
```

b. プロパティーファイルを、前のステップで作成した **EAP_HOME/modules/myorg-conf/main/properties/** ディレクトリーに移動します。

c. 以下の XML が含まれる **EAP_HOME/modules/myorg-conf/main/** ディレクトリーに **module.xml** ファイルを作成します。

```
<module xmlns="urn:jboss:module:1.1" name="myorg-conf">
  <resources>
    <resource-root path="properties"/>
  </resources>
</module>
```

2. サーバー設定ファイルの **ee** サブシステムを変更します。Managemet CLI を使用するか、ファイルを手動で編集できます。

- 管理 CLI を使用してサーバー設定ファイルを変更するには、以下の手順に従います。

a. サーバーを起動し、管理 CLI へ接続します。

- Linux の場合は、コマンドラインで以下を入力します。

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

- Windows の場合は、コマンドラインで以下を入力します。

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

次の応答が表示されるはずです。

```
Connected to standalone controller at localhost:9999
```

b. **ee** サブシステムで **myorg-conf**{-module> 要素を作成するには、コマンドラインで以下を入力します。

```
/subsystem=ee:write-attribute(name=global-modules, value=
[{"name"=>"myorg-conf","slot"=>"main"}])
```

以下の結果が表示されるはずです。

```
{"outcome" => "success"}
```

- サーバー設定ファイルを手作業で編集する場合は、次の手順に従ってください。

a. サーバーを停止し、テキストエディターでサーバー設定ファイルを開きます。スタンドアロンサーバーを実行している場合は、これは **EAP_HOME/standalone/configuration/standalone.xml** ファイル、管理対象ドメインを実行している場合は **EAP_HOME/domain/configuration/domain.xml** ファイルになります。

b. **ee** サブシステムを見つけ、**myorg-conf** のグローバルモジュールを追加します。以下は、**myorg-conf** 要素が含まれるように変更された **ee** サブシステム要素の例になります。

例7.3 **myorg-conf** 要素

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" >
    <global-modules>
        <module name="myorg-conf" slot="main" />
    </global-modules>
</subsystem>
```

3. **my.properties** という名前のファイルを正しいモジュールの場所にコピーした場合、以下のようなコードを使用してプロパティーファイルをロードできるようになります。

例7.4 プロパティーファイルの読み込み

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

バグの報告

7.5. 外部 JBOSS モジュールディレクトリーの定義

概要

デフォルトでは、JBoss EAP は **EAP_HOME/modules/** ディレクトリーでモジュールを検索します。**JBOSS_MODULEPATH** 環境変数を定義するか、起動設定ファイルで変数を設定することで、JBoss EAP が 1 つまたは複数の外部ディレクトリーを検索するように指示できます。このトピックでは、両方の方法について説明します。

手順7.3 JBOSS_MODULEPATH 環境変数の設定

- 1 つ以上の外部モジュールディレクトリーを指定するには、**JBOSS_MODULEPATH** 環境変数を定義します。

  Linux の場合は、コロンを使用してディレクトリーのリストを区切ます。以下に例を示します。

  例7.5 **JBOSS_MODULEPATH** 環境変数

  ```
  export JBOSS_MODULEPATH=EAP_HOME/modules/:/home/username/external/modules/directory/
  ```

  Windows の場合は、セミコロンを使用してディレクトリーのリストを区切ります。以下に例を示します。

  例7.6 **JBOSS_MODULEPATH** 環境変数

```
SET JBOSS_MODULEPATH=EAP_HOME\modules\;D:\JBoss-Modules\
```

手順7.4 起動設定ファイルでの JBOSS_MODULEPATH 変数の設定

- *グローバル環境変数を設定しない場合は、JBoss EAP 起動設定ファイルに **JBOSS_MODULEPATH** 変数を設定できます。スタンドアロンサーバーを実行している場合は、**EAP_HOME/bin/standalone.conf** ファイルになります。サーバーが管理対象ドメインで実行されている場合、これは **EAP_HOME/bin/domain.conf** ファイルになります。*

  *以下は、**standalone.conf** ファイルに **JBOSS_MODULEPATH** 変数を設定するコマンドの例です。*

  *例7.7 **standalone.conf** entry*

  ```
  JBOSS_MODULEPATH="EAP_HOME/modules/:/home/username/external/modules/directory/"
  ```

バグの報告

7.6. 参照資料

7.6.1. 含まれるモジュール

JBoss EAP 6 に含まれるモジュールとそれらのモジュールがサポートされるかどうかの表は、カスタマーポータルの を参照してください https://access.redhat.com/articles/1122333。

バグの報告

7.6.2. 動的モジュールの名前付け

すべてのデプロイメントは JBoss EAP 6 によってモジュールとしてロードされ、以下の慣例に従って名前が付けられます。

- *WAR および JAR ファイルのデプロイメントは次の形式で名前が付けられます。*

  ```
  deployment.DEPLOYMENT_NAME
  ```

  *たとえば、**inventory.war** と **store.jar** のモジュール名はそれぞれ **deployment.inventory.war** と **deployment.store.jar** になります。*

- *エンタープライズアーカイブ内のサブデプロイメントは次の形式で名前が付けられます。*

  ```
  deployment.EAR_NAME.SUBDEPLOYMENT_NAME
  ```

  *たとえば、エンタープライズアーカイブ **accounts.ear** 内の **reports.war** のサブデプロイメントは **deployment.accounts.ear.reports.war** のモジュール名を持ちます。*

バグの報告

第8章 JSVC

8.1. はじめに

8.1.1. Jsvc

Jsvc は、Java アプリケーションをバックグラウンドサービスとして UNIX および UNIX 系プラットフォームで実行できるライブラリーとアプリケーションのセットです。これにより、アプリケーションは特権ユーザーとして操作を実行でき、実行後に非特権ユーザーに切り替えできます。

Jsvc はランチャープロセス、コントローラープロセス、および制御されたプロセスの 3 つのプロセスを使用します。制御されたプロセスはメインの Java スレッドでもあります。JVM がクラッシュすると、コントローラープロセスが 60 秒以内に JVM を再起動します。Jsvc はデーモンプロセスで、JBoss EAP 6 では特権ユーザーによって起動される必要があります。

注記

Jsvc は Red Hat Enterprise Linux、Solaris、および HP-UX のみで使用できます。Microsoft Windows の同様の機能については、Red Hat カスタマーポータル から利用可能な **Native Utilities for Windows Server** ダウンロードの **prunsrv.exe** を参照してください。

バグの報告

8.1.2. Jsvc を使用した JBoss EAP の起動および停止

Jsvc を使用して JBoss EAP を起動および停止する手順は、スタンドアロンまたはドメインで動作するモードによって異なります。JBoss EAP がドメインモードで実行される場合、Jsvc はドメインコントローラーのプロセスのみを処理することに注意してください。Jsvc を使用して JBoss EAP を起動するために使用するコマンドはいずれも、特権ユーザーによって実行する必要があります。

前提条件

- Zip を使用して JBoss EAP がインストールされた場合、以下の条件を満たしている必要があります。
    - Red Hat カスタマーポータルからダウンロードできる、お使いのオペレーティングシステム用の『**ネイティブユーティリティー**』パッケージをインストールします。『『**インストール『ガイド』の「ネイティブコンポーネントおよびネイティブユーティリティーのインストール』（Zip、インストーラー）』**」を参照してください。
    - JBoss EAP 6 インスタンスが実行されるユーザーアカウントを作成します。サーバーの起動と停止に使用されるアカウントは、JBoss EAP がインストールされたディレクトリーへの読み書きアクセスを持っている必要があります。
- RPM を使用して JBoss EAP がインストールされた場合は、『apache-commons-daemon-jsvc-eap6』パッケージをインストールします。『『**インストール『ガイド』』の「ネイティブコンポーネントおよびネイティブユーティリティーのインストール（RPM インストール）』**」を参照してください。

以下のコマンドは、スタンドアロンまたはドメインモードのいずれかで JBoss EAP を起動および停止します。ファイルの場所は、JBoss EAP 6 で Jsvc をインストールする方法によって異なることに注意してください。以下の表を使用して、コマンドの変数を解決するために使用するファイルを決定しま

す。

スタンドアロンモード

以下は、スタンドアロンモードで JBoss EAP を起動または停止する手順です。

表8.1 Zip インストールの Jsvc ファイルの場所 - スタンドアロンモード

手順のファイル参照	ファイルの場所
EAP-HOME	${eap-installation-location}/jboss-eap-${version}
JSVC-BIN	EAP_HOME/modules/system/layers/base/native/sbin/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	EAP_HOME/standalone/configuration
LOG-DIR	EAP_HOME/standalone/log

表8.2 RPM インストールの Jsvc ファイルの場所 - スタンドアロンモード

手順のファイル参照	ファイルの場所
EAP-HOME	/usr/share/jbossas
JSVC-BIN	/usr/bin/jsvc-eap6/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	/etc/jbossas/standalone
LOG-DIR	/var/log/jbossas/standalone

スタンドモードでの JBoss EAP の起動

- *JSVC_BIN* \
  -outfile *LOG_DIR/**jsvc.out.log*** \
  -errfile *LOG_DIR/**jsvc.err.log*** \
  -pidfile *LOG_DIR/**jsvc.pid*** \
  -user jboss \
  -D[Standalone] -XX:+UseCompressedOops -Xms1303m \
  -Xmx1303m -XX:MaxPermSize=256m \
  -Djava.net.preferIPv4Stack=true
  -Djboss.modules.system.pkgs=org.jboss.byteman \
  -Djava.awt.headless=true \
  -Dorg.jboss.boot.log.file=*LOG_DIR/**server.log*** \
  -Dlogging.configuration=file:*CONF_DIR/**logging.properties*** \
  -Djboss.modules.policy-permissions \

```
-cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
-Djboss.home.dir=EAP_HOME \
-Djboss.server.base.dir=EAP_HOME/standalone  \
@org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules \
-jaxpmodule javax.xml.jaxp-provider \
org.jboss.as.standalone
```

スタンドアロンモードでの JBoss EAP の停止

- ```
  JSVC_BIN \
  -stop \
  -outfile LOG_DIR/jsvc.out.log  \
  -errfile LOG_DIR/jsvc.err.log  \
  -pidfile LOG_DIR/jsvc.pid  \
  -user jboss \
  -D[Standalone] -XX:+UseCompressedOops -Xms1303m \
  -Xmx1303m -XX:MaxPermSize=256m \
  -Djava.net.preferIPv4Stack=true \
  -Djboss.modules.system.pkgs=org.jboss.byteman \
  -Djava.awt.headless=true \
  -Dorg.jboss.boot.log.file=LOG_DIR/server.log \
  -Dlogging.configuration=file:CONF_DIR/logging.properties \
  -Djboss.modules.policy-permissions \
  -cp EAP_HOME/jboss-modules.jar:JSVC_JAR \
  -Djboss.home.dir=EAP_HOME \
  -Djboss.server.base.dir=EAP_HOME/standalone  \
  @org.jboss.modules.Main -start-method main \
  -mp EAP_HOME/modules \
  -jaxpmodule javax.xml.jaxp-provider \
  org.jboss.as.standalone
  ```

ドメインモード

以下は、ドメインモードで JBoss EAP を起動または停止する手順です。ドメインモードの場合は、JAVA_HOME 変数を Java ホームディレクトリーに置き換える必要があることに注意してください。

表8.3 Zip インストールの Jsvc ファイルの場所 - ドメインモード

手順のファイル参照	ファイルの場所
EAP-HOME	${eap-installation-location}/jboss-eap-${version}
JSVC-BIN	EAP_HOME/modules/system/layers/base/native/sbin/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	EAP_HOME/domain/configuration
LOG-DIR	EAP_HOME/domain/log
  ```

表8.4 RPM インストールの Jsvc ファイルの場所 - ドメインモード

手順のファイル参照	ファイルの場所
EAP-HOME	/usr/share/jbossas
JSVC-BIN	/usr/bin/jsvc-eap6/jsvc
JSVC-JAR	EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar
CONF-DIR	/etc/jbossas/domain
LOG-DIR	/var/log/jbossas/domain

ドメインモードでの JBoss EAP の起動

- *JSVC_BIN* \
  ***-outfile** LOG_DIR/**jsvc.out.log** \
  ***-errfile** LOG_DIR/**jsvc.err.log** \
  ***-pidfile** LOG_DIR/**jsvc.pid** \
  ***-user jboss** \
  ***-nodetach -D"[Process Controller]" -server -Xms64m** \
  ***-Xmx512m -XX:MaxPermSize=256m** \
  ***-Djava.net.preferIPv4Stack=true** \
  ***-Djboss.modules.system.pkgs=org.jboss.byteman** \
  ***-Djava.awt.headless=true** \
  ***-Dorg.jboss.boot.log.file=**LOG_DIR/**process-controller.log** \
  ***-Dlogging.configuration=file:**CONF_DIR/**logging.properties** \
  ***-Djboss.modules.policy-permissions** \
  ***-cp "**EAP_HOME/**jboss-modules.jar:**JSVC_JAR**"** \
  ***org.apache.commons.daemon.support.DaemonWrapper** \
  ***-start org.jboss.modules.Main -start-method main** \
  ***-mp EAP_HOME/modules org.jboss.as.process-controller** \
  ***-jboss-home** EAP_HOME **-jvm $**JAVA_HOME/**bin/java** \
  ***-mp** EAP_HOME/**modules --** \
  ***-Dorg.jboss.boot.log.file=**LOG_DIR/**host-controller.log** \
  ***-Dlogging.configuration=file:**CONF_DIR/**logging.properties** \
  ***-Djboss.modules.policy-permissions** \
  ***-server -Xms64m -Xmx512m -XX:MaxPermSize=256m** \
  ***-Djava.net.preferIPv4Stack=true** \
  ***-Djboss.modules.system.pkgs=org.jboss.byteman** \
  ***-Djava.awt.headless=true -- -default-jvm $**JAVA_HOME/**bin/java**

ドメインモードでの JBoss EAP の停止

- *JSVC_BIN* \
  ***-stop** \
  ***-outfile** LOG_DIR/**jsvc.out.log** \
  ***-errfile** LOG_DIR/**jsvc.err.log** \
  ***-pidfile** LOG_DIR/**jsvc.pid** \
  ***-user jboss** \

```
-nodetach -D"[Process Controller]" -server -Xms64m \
-Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true  \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true  \
-Dorg.jboss.boot.log.file=LOG_DIR/process-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-cp "EAP_HOME/jboss-modules.jar:JSVC_JAR" \
org.apache.commons.daemon.support.DaemonWrapper \
-start org.jboss.modules.Main -start-method main \
-mp EAP_HOME/modules org.jboss.as.process-controller \
-jboss-home EAP_HOME -jvm $JAVA_HOME/bin/java \
-mp EAP_HOME/modules -- \
-Dorg.jboss.boot.log.file=LOG_DIR/host-controller.log \
-Dlogging.configuration=file:CONF_DIR/logging.properties \
-Djboss.modules.policy-permissions \
-server -Xms64m -Xmx512m -XX:MaxPermSize=256m \
-Djava.net.preferIPv4Stack=true \
-Djboss.modules.system.pkgs=org.jboss.byteman \
-Djava.awt.headless=true -- -default-jvm $JAVA_HOME/bin/java
```

注記

JVM のクラッシュなど、JBoss EAP 6 が通常通りに終了した場合、Jsvc は自動的に再起動します。JBoss EAP 6 が正常に終了した場合、Jsvc も停止します。

バグの報告

第9章 グローバル値

9.1. バルブ

バルブは、アプリケーションのパイプラインを処理するリクエストに挿入される Java クラスです。これは、サーブレットフィルターの前にパイプラインに挿入されます。バルブは、認証などの他の処理にパスしたり、リクエストをキャンセルしたりする前に、リクエストに変更を加えることができます。

バルブは、サーバーレベルまたはアプリケーションレベルで設定できます。唯一の違いは、どのように設定およびパッケージ化されているかです。

- グローバルバルブはサーバーレベルで設定され、サーバーにデプロイされたすべてのアプリケーションに適用されます。グローバルバルブの設定手順は、JBoss EAP の『**管理および設定ガイド**』を参照してください。
- アプリケーションレベルで設定されたバルブはアプリケーションデプロイメントとパッケージ化され、特定のアプリケーションのみが影響を受けます。アプリケーションレベルでバルブを設定する手順は、JBoss EAP の『**開発ガイド**』を参照してください。

6.1.0 およびそれ以降のバージョンはグローバルバルブをサポートします。

バグの報告

9.2. グローバルバルブ

グローバルバルブは、デプロイされたすべてのアプリケーションのパイプラインを処理するリクエストに挿入されるバルブです。バルブは、JBoss EAP 6 で静的モジュールとしてパッケージ化およびインストールされ、グローバル化されます。グローバルバルブは Web サブシステムで設定されます。

6.1.0 およびそれ以降のバージョンのみがグローバルバルブをサポートします。

グローバルバルブの設定方法については、『オーバークラウドの高度なカスタマイズ』の「グローバルバルブの設定」を参照してください。 「グローバルバルブの設定」

バグの報告

9.3. オーセンティケーターバルブ

オーセンティケーターバルブは、リクエストの認証情報を認証するバルブです。このようなバルブは **org.apache.catalina.authenticator.AuthenticatorBase** のサブクラスで、**authenticate(Request request, Response response, LoginConfig config)** メソッドを上書きします。

このバルブを使用して追加の認証スキームを実装できます。

バグの報告

9.4. グローバルバルブのインストール

グローバルバルブは、JBoss EAP 6 で静的モジュールとしてパッケージ化およびインストールする必要があります。このタスクは、モジュールのインストール方法を示しています。

前提条件:

- バルブを作成し、JAR ファイルにパッケージ化する必要があります。

- モジュール用に **module.xml** ファイルがすでに作成されている必要があります。

  module.xml ファイルの例については、「モジュール」 を参照してください。

手順9.1 グローバルモジュールのインストール

1. モジュールインストールディレクトリーの作成
   インストールするモジュールのディレクトリーはアプリケーションサーバーの **modules** ディレクトリーに作成する必要があります。

   ```
   EAP_HOME/modules/system/layers/base/MODULENAME/main
   ```

   ```
   $ mkdir -P EAP_HOME/modules/system/layers/base/MODULENAME/main
   ```

2. ファイルのコピー
   JAR および **module.xml** ファイルを、手順1 で作成したディレクトリーにコピーします。

   ```
   $ cp MyValves.jar module.xml
   EAP_HOME/modules/system/layers/base/MODULENAME/main
   ```

モジュールで宣言したバルブクラスを **Web** サブシステムで設定できるようになります。

バグの報告

9.5. グローバルバルブの設定

グローバルバルブは **Web** サブシステムで有効化および設定されます。これは、**JBoss CLI** ツールを使用して行います。

手順9.2 グローバルバルブの設定

1. バルブの有効化
   add 操作を使用して、新しいバルブエントリーを追加します。

   ```
   /subsystem=web/valve=VALVENAME:add(module="MODULENAME",class-
   name="CLASSNAME")
   ```

   次の値を指定する必要があります。

   - **VALVENAME** アプリケーション設定でこのバルブを参照するために使用される名前。
   - **MODULENAME** 設定されている値が含まれるモジュール。
   - **CLASSNAME** モジュールの特定バルブのクラス名。

   以下に例を示します。

   ```
   /subsystem=web/valve=clientlimiter:add(module="clientlimitermodule",class-
   name="org.jboss.samplevalves.RestrictedUserAgentsValve")
   ```

2. オプション: パラメーターの指定
   バルブに設定パラメーターがある場合は、**add-param** 操作で指定します。

```
/subsystem=web/valve=VALVENAME:add-param(param-name="PARAMNAME",
param-value="PARAMVALUE")
```

次の値を指定する必要があります。

- **VALVENAME**アプリケーション設定でこのバルブを参照するために使用される名前。
- **PARAMNAME**特定のバルブに設定されているパラメーターの名前。
- **PARAMVALUE**（指定されたパラメーターの値）

以下に例を示します。

例9.1 バルブの設定

```
/subsystem=web/valve=clientlimiter:add-param(
   param-name="restrictedUserAgents",
   param-value="^.*MS Web Services Client Protocol.*$"
)
```

バルブがデプロイされたすべてのアプリケーションに対して有効になり、設定されます。

カスタム『**バルブの作成**』方法については、『『**開発ガイド**』』の「カスタムバルブの作成」セクションを参照してください。

バグの報告

第10章 アプリケーションデプロイメント

10.1. アプリケーションデプロイメント

JBoss EAP 6 には、管理環境と開発環境の両方に対応するよう、さまざまなアプリケーションデプロイメントおよび設定オプションが備えられています。管理者は、管理コンソールと管理 CLI で、実稼働環境のアプリケーションデプロイメントを管理するのに最適なグラフィカルおよびコマンドラインインターフェースを提供します。開発者は、設定可能なファイルシステムのデプロイメントスキャナー、JBoss Developer Studio などの IDE の使用、Maven を介したデプロイメントおよびアンデプロイメントなど、多くのアプリケーションデプロイメントテストオプションが含まれます。

管理

- 管理コンソール
  - 「管理コンソールを使用してデプロイされたアプリケーションを有効化」
  - 「管理コンソールを使用してデプロイされたアプリケーションを無効化」
- 管理 CLI
  - 「管理 CLI を使用した管理対象ドメインでのアプリケーションのデプロイ」
  - 「管理 CLI を使用したスタンドアロンサーバーでのアプリケーションのデプロイ」
  - 「管理 CLI を使用した管理対象ドメインでのアプリケーションのアンデプロイ」
  - 「管理 CLI を使用したスタンドアロンサーバーでのアプリケーションのアンデプロイ」
  - 「管理 CLI でのアプリケーションデプロイメントの管理」

開発

- デプロイメントスキャナー
  - 「デプロイメントスキャナーの設定」
  - 「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイ」
  - 「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスからアプリケーションをアンデプロイ」
  - 「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションを再デプロイ」
  - 「管理 CLI でのデプロイメントスキャナーの設定」
  - 「デプロイメントスキャナー属性のリファレンス」
  - 「デプロイメントスキャナーマーカーファイルのリファレンス」
- Maven

- 「Maven によるアプリケーションのデプロイ」
- 「Maven によるアプリケーションのアンデプロイ」

注記

アプリケーションをデプロイする前に、デプロイメント記述子の検証を有効にできます。これには、**org.jboss.metadata.parser.validate** システムプロパティーを **true** に設定します。これには、以下の 2 つの方法があります。

- サーバー起動時

  例:

  - ドメインモードの場合：

    ```
    ./domain.sh -Dorg.jboss.metadata.parser.validate=true
    ```

  - スタンドアロンモードの場合：

    ```
    ./standalone.sh -Dorg.jboss.metadata.parser.validate=true
    ```

- サーバー設定で定義する方法。

  管理 CLI を使用したシステムプロパティーの設定に関する詳細は、を参照してください。 「管理 CLI を使用したシステムプロパティーの設定」

バグの報告

10.2. 管理コンソールでのデプロイ

10.2.1. 管理コンソースでのアプリケーションデプロイメント管理

管理コンソールを介してアプリケーションをデプロイすると、使いやすいグラフィカルインターフェースを利用することができます。サーバーまたはサーバーグループにデプロイされたアプリケーションを一目で確認でき、必要に応じてアプリケーションをコンテンツリポジトリーから有効化、無効化、または削除できます。

バグの報告

10.2.2. 管理コンソールを使用してデプロイされたアプリケーションを有効化

前提条件

- 「管理コンソールへのログイン」
- 「管理コンソールでのデプロイメントの追加」

手順10.1 管理コンソールを使用してデプロイされたアプリケーションを有効化

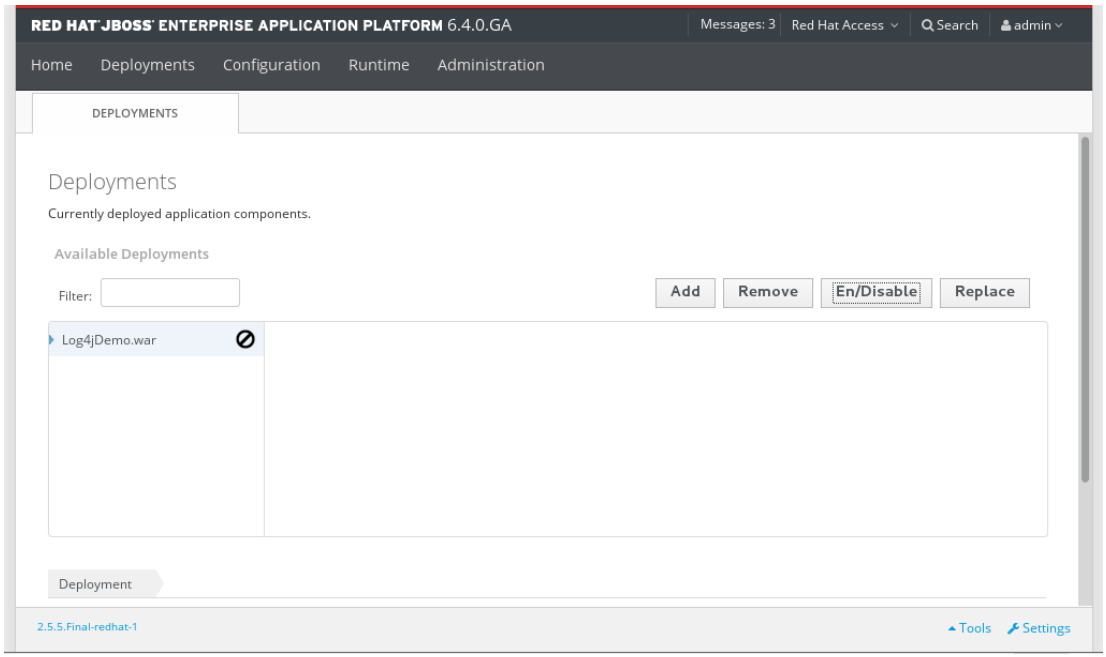
- コンソールの上部にある **Deployments** タブを選択します。

アプリケーションのデプロイメントの方法は、スタンドアロンサーバーインスタンスと管理対象ドメインのどちらにデプロイするかによって異なります。

- *スタンドアロンサーバーインスタンスでのアプリケーションの有効化*
  ***Available Deployments** の表には、利用可能なすべてのアプリケーションデプロイメントおよびそれらのステータスが表示されます。*

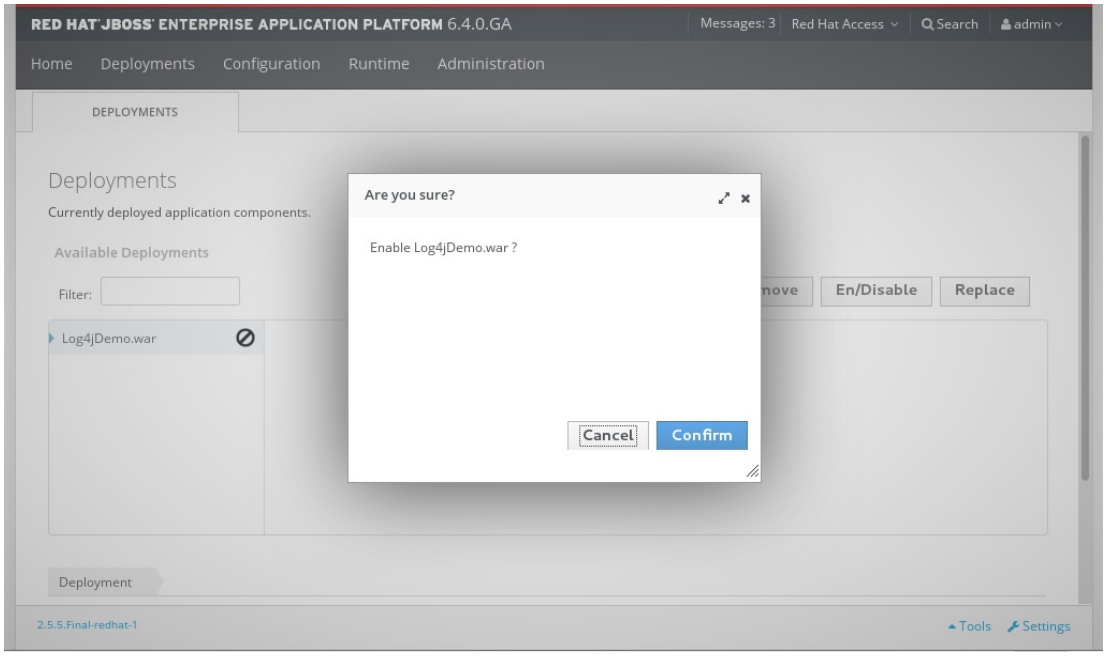
  a. *スタンドアロンサーバーインスタンスでアプリケーションを有効にするには、アプリケーションを選択し、**En/Disable** をクリックします。*

  図10.1 利用可能なデプロイメント



  b. ***Confirm** をクリックし、サーバーインスタンスでアプリケーションが有効になっていることを確認します。*

  図10.2 スタンドアロンサーバーで利用可能なデプロイメント

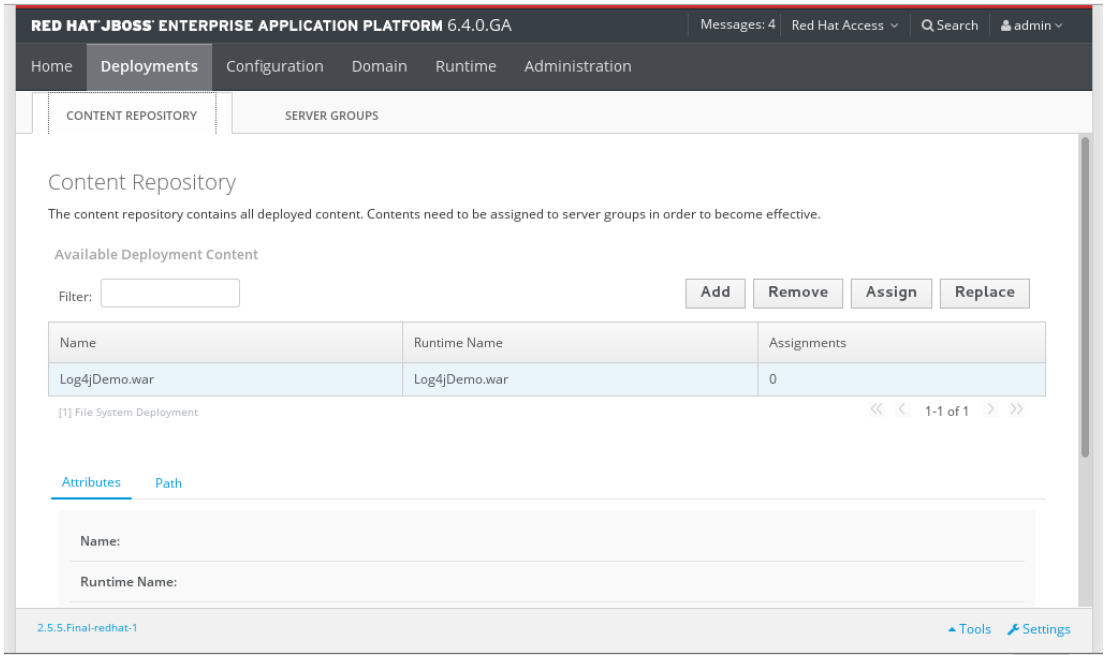


- *管理対象ドメインでのアプリケーションの有効化*

Content Repository *タブには、利用可能なデプロイメントコンテンツ テーブルがあり、利用可能なアプリケーションデプロイメントとそのステータスが表示されます。*

a. *管理対象ドメインでアプリケーションを有効にするには、デプロイするアプリケーションを選択します。****Available Deployment Content*** *の表の上にある****Assign*** *をクリックします。*

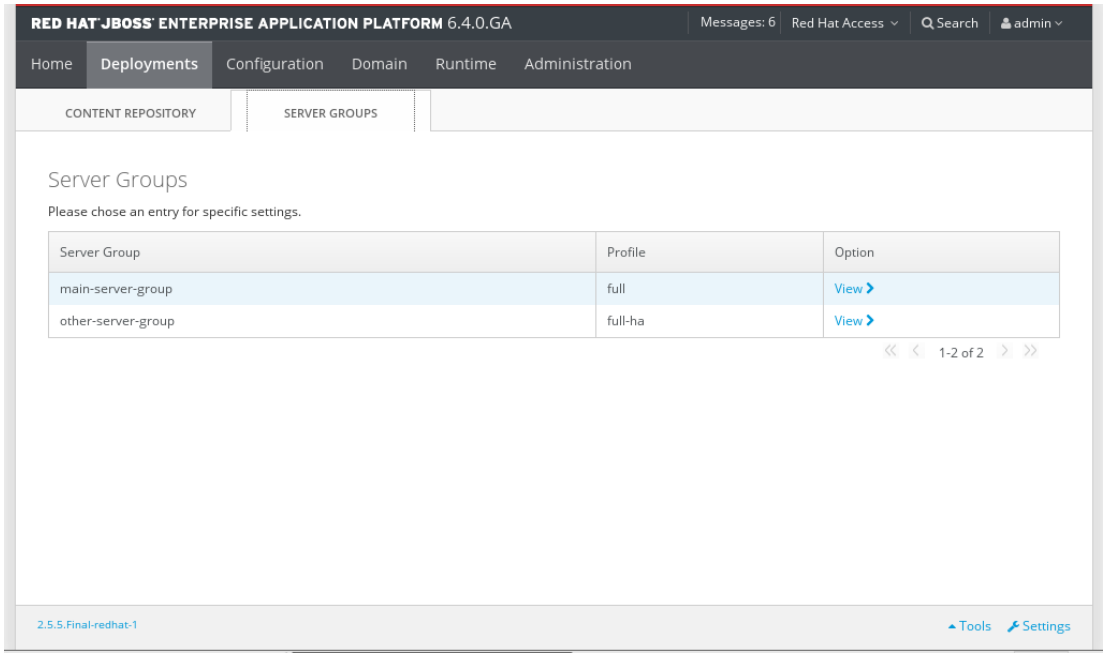
図10.3 管理対象ドメインで利用可能なデプロイメント



b. *アプリケーションを追加する各サーバーグループのボックスにチェックを入れ、****Save*** *をクリックして終了します。*

c. ***Server Groups*** *タブを選択して、****Server Groups*** *テーブルを表示します。*

図10.4 サーバーグループへのアプリケーションのデプロイメントの確認



d. *アプリケーションがまだ有効にされていない場合は、表示 をクリックして、****En/Disable*** *ボタンをクリックします。****Confirm*** *をクリックし、サーバーインスタンスでアプリケーションが有効になっていることを確認します。*

結果

関連するサーバーまたはサーバーグループでアプリケーションが有効になっている。

バグの報告

10.2.3. 管理コンソールを使用してデプロイされたアプリケーションを無効化

前提条件

- *「管理コンソールへのログイン」*
- *「管理コンソールでのデプロイメントの追加」*
- *「管理コンソールを使用してデプロイされたアプリケーションを有効化」*

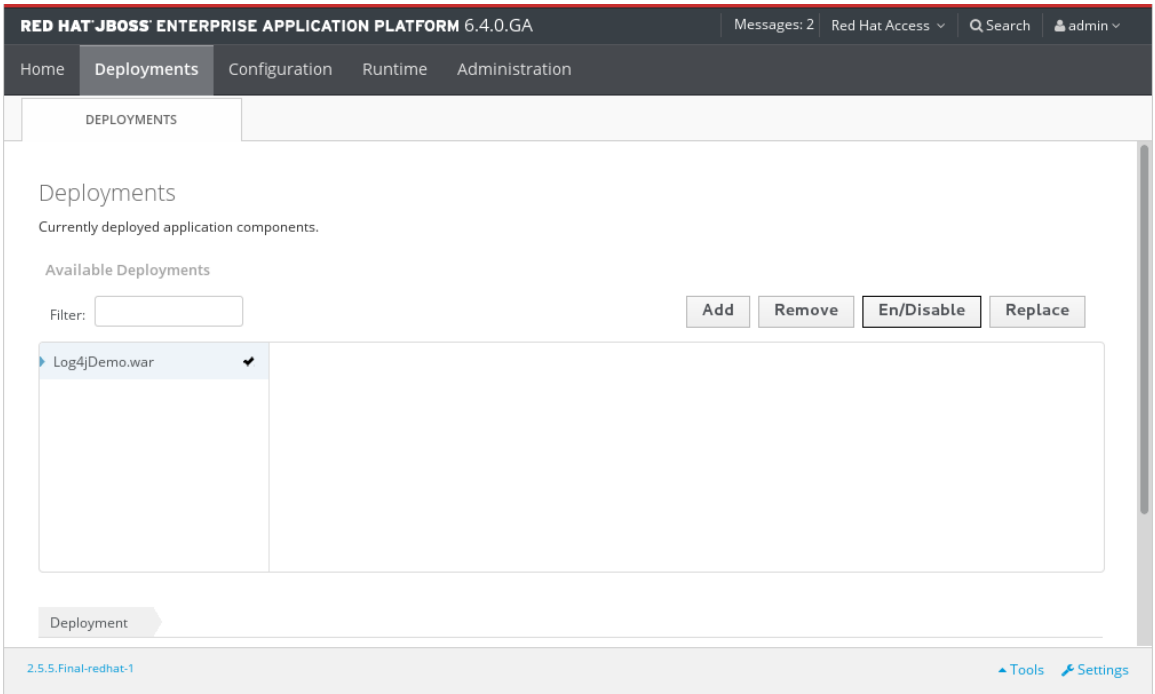
手順10.2 管理コンソールを使用してデプロイされたアプリケーションを無効化

- *コンソールの上部にある **Deployment** タブを選択します。*

  アプリケーションを無効化する方法は、スタンドアロンサーバーインスタンスと管理対象ドメインのどちらにデプロイするかどうかによって異なります。

  - *スタンドアロンサーバーインスタンスにデプロイされたアプリケーションを無効化*
    ***Available Deployments** の表には、利用可能なすべてのアプリケーションデプロイメントおよびそれらのステータスが表示されます。*

    図10.5 利用可能なデプロイメント



    a. *無効にするアプリケーションを選択します。**En/Disable** をクリックして、選択したアプリケーションを無効にします。*

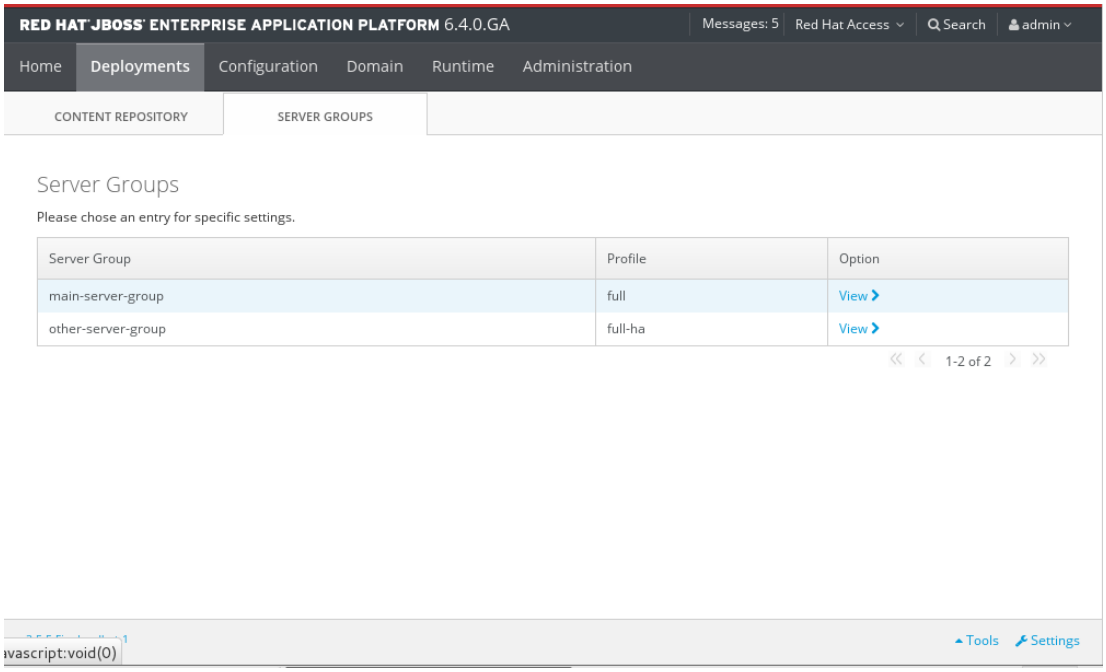
    b. ***Confirm** をクリックし、サーバーインスタンスでアプリケーションを無効にすることを確認します。*

  - *管理対象ドメインでのデプロイされたアプリケーションの無効化*

Deployments 画面には Content Repository タブが含まれます。Available Deployment Content の表には、利用可能なすべてのアプリケーションデプロイメントとそれらの状態が表示されます。

a. *Server Groups タブを選択して、サーバーグループを表示します。*

図10.6 サーバーグループのデプロイメント



b. *Server Group テーブルでサーバーグループの名前を選択し、アプリケーションをアンデプロイします。View をクリックしてアプリケーションを表示します。*

c. *アプリケーションを選択し、En/Disable をクリックして選択したサーバーのアプリケーションを無効にします。*

d. *Confirm をクリックし、サーバーインスタンスでアプリケーションを無効にすることを確認します。*

e. *必要に応じて他のサーバーグループにも同じ手順を繰り返します。アプリケーションステータスは、そのサーバーグループの Group Deployments テーブルの各サーバーグループに対して確認されます。*

結果

関連するサーバーまたはサーバーグループからアプリケーションが無効になっている。

バグの報告

10.2.4. 管理コンソールを使用したアプリケーションのアンデプロイ

前提条件

- *「管理コンソールへのログイン」*
- *「管理コンソールでのデプロイメントの追加」*
- *「管理コンソールを使用してデプロイされたアプリケーションを有効化」*

- *「管理コンソールを使用してデプロイされたアプリケーションを無効化」*

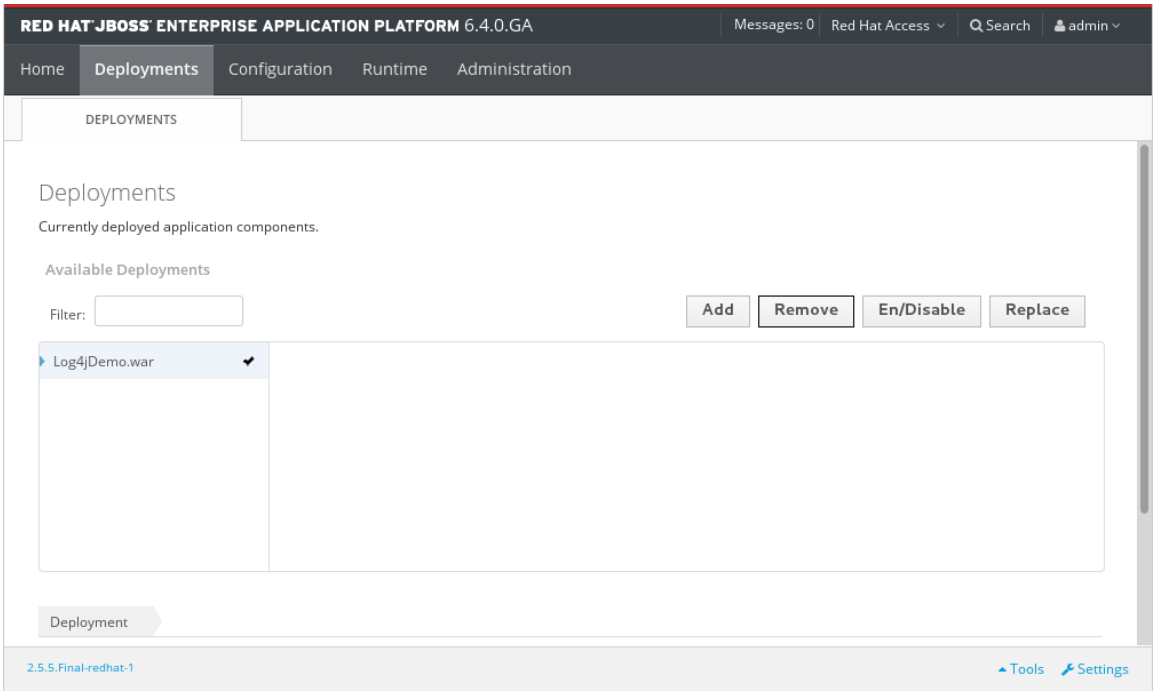
手順10.3 管理コンソールを使用したアプリケーションのアンデプロイ

- *コンソールの上部にある **Deployments** タブを選択します。*

  アプリケーションをアンデプロイする方法は、スタンドアロンサーバーインスタンスと管理対象ドメインのどちらからアンデプロイするかどうかによって異なります。

  - *デプロイされたアプリケーションをスタンドアロンサーバーインスタンスからアンデプロイ*
    ***Available Deployments** の表には、利用可能なすべてのアプリケーションデプロイメントおよびそれらのステータスが表示されます。*

    図10.7 利用可能なデプロイメント

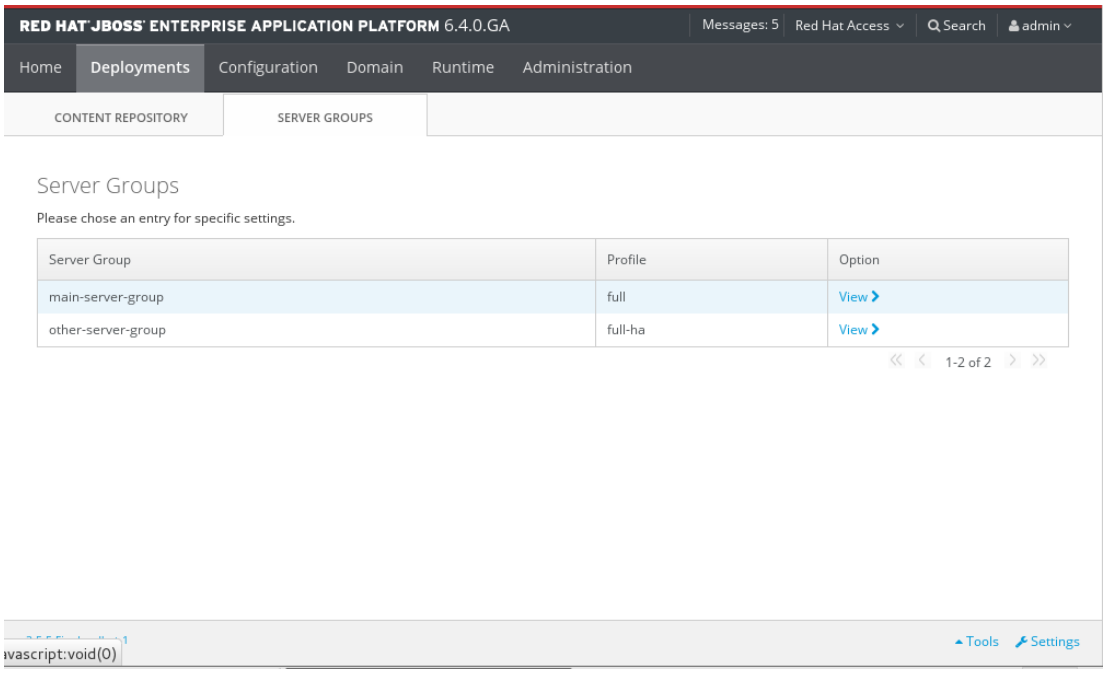
    a. *アンデプロイするアプリケーションを選択します。**Remove** をクリックして、選択したアプリケーションをアンデプロイします。*

    b. ***Confirm** をクリックし、サーバーインスタンスでアプリケーションをアンデプロイしていることを確認します。*

  - *デプロイされたアプリケーションを管理対象ドメインからアンデプロイ*
    ***Deployments** 画面には **Content Repository** タブが含まれます。**Available Deployment Content** の表には、利用可能なすべてのアプリケーションデプロイメントとそれらの状態が表示されます。*

    a. ***Server Groups** タブを選択して、サーバーグループとデプロイされたアプリケーションの状態を表示します。*

図10.8 サーバーグループのデプロイメント



b. ***Server Group*** *テーブルでサーバーグループの名前を選択し、アプリケーションをアンデプロイします。* ***View*** *をクリックしてアプリケーションを表示します。*

c. *アプリケーションを選択し、* ***Remove*** *をクリックして、選択したサーバーのアプリケーションをアンデプロイします。*

d. ***Confirm*** *をクリックし、サーバーインスタンスでアプリケーションをアンデプロイしていることを確認します。*

e. *必要に応じて他のサーバーグループにも同じ手順を繰り返します。アプリケーションステータスは、そのサーバーグループの* ***Group Deployments*** *テーブルの各サーバーグループに対して確認されます。*

結果

該当のサーバーあるいはサーバーグループからアプリケーションがアンデプロイされます。スタンドアロンインスタンスでは、デプロイメントコンテンツも削除されます。管理対象ドメインでは、デプロイメントコンテンツはコンテンツリポジトリーに残り、サーバーグループからのみアンデプロイされます。

バグの報告

10.3. 管理 CLI でのデプロイ

10.3.1. 管理 CLI でのアプリケーションデプロイメントの管理

管理 CLI を使用してアプリケーションをデプロイすると、単一のコマンドラインインターフェースでデプロイメントスクリプトを作成および実行できます。このスクリプト機能を使用して、特定のアプリケーションデプロイメントおよび管理シナリオを設定できます。スタンドアロンサーバーの場合は、1 台のサーバーのデプロイメントステータスや、管理対象ドメインの場合はサーバーのネットワーク全体を管理できます。

バグの報告

10.3.2. 管理 CLI を使用したスタンドアロンサーバーでのアプリケーションのデプロイ

前提条件

- *「管理 CLI の起動」*
- *「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」*

手順10.4 スタンドアロンサーバーでのアプリケーションのデプロイ

- *デプロイ コマンドの実行*
  *管理 CLI で、アプリケーションのデプロイメントへのパスを指定して **deploy** コマンドを入力します。*

  例10.1 デプロイコマンド

  ```
  [standalone@localhost:9999 /] deploy /path/to/test-application.war
  ```

  デプロイメントに成功しても CLI には何も出力されないことに注意してください。

結果

指定のアプリケーションが、スタンドアロンサーバーにデプロイされます。

バグの報告

10.3.3. 管理 CLI を使用したスタンドアロンサーバーでのアプリケーションのアンデプロイ

前提条件

- *「管理 CLI の起動」*
- *「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」*
- *「管理 CLI を使用したスタンドアロンサーバーでのアプリケーションのデプロイ」*

手順10.5 スタンドアロンサーバーでのアプリケーションのアンデプロイ

*デフォルトでは、**undeploy** コマンドは JBoss EAP のスタンドアロンインスタンスからデプロイメントコンテンツをアンデプロイ **および** 削除します。デプロイメントコンテンツを保持するには、パラメーター **--keep-content** を追加します。*

- ***undeploy** コマンドの実行*
  アプリケーションをアンデプロイし、デプロイメントコンテンツを削除するには、アプリケーションデプロイメントのファイル名を指定して管理 CLI の アンデプロイ コマンドを入力します。

  ```
  [standalone@localhost:9999 /] undeploy test-application.war
  ```

  *アプリケーションをアンデプロイし、デプロイメントコンテンツを保持するには、アプリケーションデプロイメントのファイル名とパラメーター **--keep-content** を指定して管理 CLI **undeploy** コマンドを入力します。*

```
[standalone@localhost:9999 /] undeploy test-application.war --keep-content
```

結果

*指定のアプリケーションがアンデプロイされます。**undeploy** コマンドは、成功すると管理 CLI へ何も出力されないことに注意してください。*

バグの報告

10.3.4. 管理 CLI を使用した管理対象ドメインでのアプリケーションのデプロイ

前提条件

- *「管理 CLI の起動」*
- *「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」*

手順10.6 管理対象ドメインでのアプリケーションのデプロイ

- *デプロイ コマンドの実行*
  *管理 CLI で、アプリケーションのデプロイメントへのパスを指定して **deploy** コマンドを入力します。**--all-server-groups** パラメーターを指定してすべてのサーバーグループにデプロイします。*

  ```
  [domain@localhost:9999 /] deploy /path/to/test-application.war --all-server-groups
  ```

  - *または、**--server-groups** パラメーターを使用して、デプロイメントの特定のサーバーグループを定義します。*

    ```
    [domain@localhost:9999 /] deploy /path/to/test-application.war --server-groups=server_group_1,server_group_2
    ```

  デプロイメントに成功しても CLI には何も出力されないことに注意してください。

結果

指定のアプリケーションが、管理対象ドメインのサーバーグループにデプロイされます。

バグの報告

10.3.5. 管理 CLI を使用した管理対象ドメインでのアプリケーションのアンデプロイ

前提条件

- *「管理 CLI の起動」*
- *「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」*
- *「管理 CLI を使用した管理対象ドメインでのアプリケーションのデプロイ」*

手順10.7 管理対象ドメインでのアプリケーションのアンデプロイ

- ***undeploy** コマンドの実行*

管理 CLI で、アプリケーションのデプロイメントのファイル名を指定して **undeploy** コマンドを入力します。**--all-relevant-server-groups** パラメーターを追加して、最初にデプロイしたサーバーグループからアプリケーションをアンデプロイできます。

```
[domain@localhost:9999 /] undeploy test-application.war --all-relevant-server-groups
```

アンデプロイメントに成功すると、CLI へ出力されません。

結果

指定のアプリケーションがアンデプロイされます。

バグの報告

10.4. HTTP API を用いたデプロイ

10.4.1. HTTP API を使用したアプリケーションのデプロイ

概要

以下の手順に従って、HTTP API よりアプリケーションをデプロイできます。

前提条件

- 管理インターフェースのユーザーを追加します。リモート管理インターフェースの初期管理ユーザーを作成する方法は、を参照してください。「管理インターフェースのユーザーの追加」

手順10.8 HTTP API を使用したアプリケーションのデプロイ

- 要件に関連する **deploy** または **undeploy** コマンドのいずれかを使用します。

例10.2 デプロイおよびアンデプロイコマンド

```
Deploy
-----------------------------
curl --digest -L -D - http://<host>:<port>/management --header "Content-Type:
application/json" -d '{"operation" : "composite", "address" : [], "steps" :
[{"operation" : "add", "address" : {"deployment" : "<runtime-name>"}, "content" :
[{"url" : "file:<path-to-archive>"}]},{"operation" : "deploy", "address" :
{"deployment" : "<runtime-name>"}}],"json.pretty":1}' -u <user>:<pass>

Example:
-------
curl --digest -L -D - http://localhost:9990/management --header "Content-Type:
application/json" -d '{"operation" : "composite", "address" : [], "steps" :
[{"operation" : "add", "address" : {"deployment" : "example.war"}, "content" :
[{"url" : "file:/home/$user/example.war"}]},{"operation" : "deploy", "address" :
{"deployment" : "example.war"}}],"json.pretty":1}' -u user:password


Undeploy
-----------------------------
curl --digest -L -D - http://<host>:<port>/management --header "Content-Type:
application/json" -d '{"operation" : "composite", "address" : [], "steps" :
```

```
[{"operation" : "undeploy", "address" : {"deployment" : "<runtime-name>"}},
{"operation" : "remove", "address" : {"deployment" : "<runtime-
name>"}}],"json.pretty":1}' -u <user>:<pass>

Example:
-------
curl --digest -L -D - http://localhost:9990/management --header "Content-Type:
application/json" -d '{"operation" : "composite", "address" : [], "steps" :
[{"operation" : "undeploy", "address" : {"deployment" : "example.war"}},
{"operation" : "remove", "address" : {"deployment" :
"example.war"}}],"json.pretty":1}' -u user:password
```

注記

JSON 要求をプログラムで生成する方法の詳細は、 を参照してください https://access.redhat.com/solutions/82463。

バグの報告

10.5. デプロイメントスキャナーでのデプロイ

10.5.1. デプロイメントスキャナーでのアプリケーションデプロイメント管理

デプロイメントスキャナーを介してスタンドアロンサーバーインスタンスにアプリケーションをデプロイすると、迅速な開発サイクルに適した方法でアプリケーションをビルドし、テストすることができます。デプロイメントスキャナーは、さまざまなアプリケーションタイプのデプロイメント頻度や動作にニーズに合わせて設定することができます。

バグの報告

10.5.2. デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイ

前提条件

- *「JBoss EAP 6 の起動」*

概要

このタスクは、デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイする方法を説明します。「アプリケーションデプロイメント」トピックで説明されているように、この方法は開発者の利便性のために保持され、本番稼働環境でのアプリケーション管理には管理コンソールと管理 CLI の方法が推奨されます。

手順10.9 デプロイメントスキャナーを使用したアプリケーションのデプロイ

1. *デプロイメントフォルダーへのコンテンツのコピー*
   *アプリケーションファイルを **EAP_HOME/standalone/deployments/** にあるデプロイメントフォルダーにコピーします。*

2. *デプロイメントスキャンモード*

アプリケーションのデプロイメント方法は 2 つあります。自動デプロイメントスキャナーモードと手動のデプロイメントスキャナーモードを選択できます。デプロイメント方法のいずれかを開始する前に、「管理 CLI でのデプロイメントスキャナーの設定」 を読んでください。

- 自動デプロイメント
  デプロイメントスキャナーはフォルダーの状態の変更を選択し、「管理 CLI でのデプロイメントスキャナーの設定」 で定義されたマーカーファイルを作成します。

- 手動デプロイメント
  デプロイメントスキャナーには、デプロイメントプロセスをトリガーするマーカーファイルが必要です。以下の例では、**Unix touch** コマンドを使用して新しい **.dodeploy** ファイルを作成します。

例10.3 **touch** コマンドを使用したデプロイ

```
[user@host bin]$ touch
$EAP_HOME/standalone/deployments/example.war.dodeploy
```

結果

アプリケーションファイルがアプリケーションサーバーにデプロイされます。デプロイメントフォルダーにマーカーファイルが作成され、デプロイメントが成功したことを示し、アプリケーションは管理コンソールで **Enabled** とフラグが付けられます。

例10.4 デプロイメント後のデプロイメントフォルダーコンテンツ

```
example.war
example.war.deployed
```

バグの報告

10.5.3. デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスからアプリケーションをアンデプロイ

前提条件

- 「JBoss EAP 6 の起動」
- 「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイ」

概要

このタスクでは、デプロイメントスキャナーでデプロイされたスタンドアロンサーバーインスタンスからアプリケーションをアンデプロイする方法を説明します。「アプリケーションデプロイメント」 トピックで説明されているように、この方法は開発者の利便性のために保持され、本番稼働環境でのアプリケーション管理には管理コンソールと管理 CLI の方法が推奨されます。

注記

*デプロイメントスキャナーは、アプリケーション管理の他のデプロイメント方法と併用しないでください。管理コンソールでアプリケーションサーバーから削除されたアプリケーションは、**deployment** ディレクトリーに含まれるマーカーファイルやアプリケーションに影響を与えずにランタイムから削除されます。誤ってリダクションやその他のエラーのリスクを最小限に抑えるには、実稼働環境で管理 **CLI** と管理コンソールを使用します。*

手順10.10 以下の方法の1 つを用いたアプリケーションのアンデプロイ

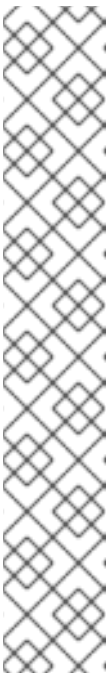
- *アプリケーションのアンデプロイ*
  アプリケーションをデプロイする方法は 2 つありますが、アプリケーションをデプロイメントフォルダーから削除する場合と、デプロイメントの状態のみを変更する場合によって、使用する方法が異なります。

  - *マーカーファイルの削除によるアンデプロイ*
    *デプロイされたアプリケーションの **example.war.deployed** マーカーファイルを削除し、ランタイムからアプリケーションのアンデプロイを開始するためにデプロイメントスキャナーをトリガーします。*

    結果

    *デプロイメントスキャナーはアプリケーションをアンデプロイし、**example.war.undeployed** マーカーファイルを作成します。アプリケーションはデプロイメントフォルダーに残ります。*

  - *アプリケーションの削除によるアンデプロイ*



注記

*この方法を使用した展開形式の **WAR** ファイルのアンデプロイは有効ではありません。マーカーファイルを削除してアンデプロイメントのみが有効になります。展開形式の **WAR** ファイルをアンデプロイしようとすると、以下のメッセージがログに記録されます。*

> ***WARN [org.jboss.as.server.deployment.scanner] (DeploymentScanner-threads - 2) JBAS015006: The deployment scanner found that the content for exploded deployment EXAMPLE.war has been deleted, but auto-deploy/undeploy for exploded deployments is not enabled and the EXAMPLE.war.deployed marker file for this deployment has not been removed. As a result, the deployment is not being undeployed, but resources needed by the deployment may have been deleted and application errors may occur. Deleting the EXAMPLE.war.deployed marker file to trigger undeploy is recommended.***

デプロイメントディレクトリーからアプリケーションを削除して、ランタイムからアプリケーションのアンデプロイを開始するためにデプロイメントスキャナーをトリガーします。

結果

デプロイメントスキャナーはアプリケーションをアンデプロイし、**filename.filetype.undeployed** マーカーファイルを作成します。アプリケーションはデプロイメントフォルダーに存在しません。

結果

アプリケーションファイルはアプリケーションサーバーからアンデプロイされ、管理コンソールの **Deployments** 画面に表示されません。

バグの報告

10.5.4. デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションを再デプロイ

前提条件

- 「JBoss EAP 6 の起動」
- 「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイ」

概要

このタスクでは、デプロイメントスキャナーを使用してデプロイされたスタンドアロンサーバーインスタンスにアプリケーションを再デプロイする方法を説明します。「アプリケーションデプロイメント」トピックで説明されているように、この方法は開発者の利便性のために保持され、本番稼働環境でのアプリケーション管理には管理コンソールと管理 CLI の方法が推奨されます。

手順10.11 アプリケーションをスタンドアロンサーバーへ再デプロイ

- アプリケーションの再デプロイ
  デプロイメントスキャナーを使用してデプロイされたアプリケーションを再デプロイするには、3 つの方法があります。これらの方法は、デプロイメントスキャナーをトリガーしてデプロイメントサイクルを開始し、個人設定に合わせて選択できます。
  - マーカーファイルの変更による再デプロイ
    マーカーファイルのアクセスおよび変更のタイムスタンプを変更して、デプロイメントスキャナーの再デプロイメントをトリガーします。以下の Linux の例では、Unix **touch** コマンドが使用されています。

    例10.5 Unix **touch** コマンドを使用した再デプロイ

    ```
    [user@host bin]$ touch
    EAP_HOME/standalone/deployments/example.war.dodeploy
    ```

    結果

    デプロイメントスキャナーはマーカーファイルの変更を検出し、アプリケーションを再デプロイします。新しい **.deployed** ファイルマーカーが、以前のファイルマーカーに置き換わります。

  - 新しい **.dodeploy** マーカーファイルの作成による再デプロイ
    新しい **.dodeploy** マーカーファイルを作成して、デプロイメントスキャナーの再デプロイメントをトリガーします。「デプロイメントスキャナーを使用してスタンドアロンサー

バーインスタンスにアプリケーションをデプロイ」 の手動デプロイメント手順を参照してください。

- *マーカーファイルの削除による再デプロイ*
  *「デプロイメントスキャナーマーカーファイルのリファレンス」 で説明されているように、デプロイされたアプリケーションの **.deployed** マーカーファイルを削除すると、アンデプロイメントがトリガーされ、**.undeployed** マーカーが作成されます。デプロイメント解除マーカーを削除すると、デプロイメントサイクルが再びトリガーされます。詳細は、「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスからアプリケーションをアンデプロイ」 を参照してください。*

結果

アプリケーションファイルが再デプロイされます。

バグの報告

10.5.5. デプロイメントスキャナーマーカーファイルのリファレンス

マーカーファイル

マーカーファイルは、デプロイメントスキャナーサブシステムの一部です。これらのファイルは、スタンドアロンサーバーインスタンスのデプロイメントディレクトリー内のアプリケーションの状態をマークします。マーカーファイルの名前はアプリケーションと同じで、ファイル接尾辞はアプリケーションのデプロイメントの状態を示します。以下の表は、各マーカーファイルのタイプおよび応答を定義します。

> *例10.6 マーカーファイル例*
>
> *以下の例は、**testapplication.war** という名前のアプリケーションの正常にデプロイされたインスタンスのマーカーファイルを示しています。*
>
> ```
> testapplication.war.deployed
> ```

表10.1 マーカーファイルタイプ定義

ファイル名接尾辞	生成	説明
.dodeploy	ユーザー生成	コンテンツをランタイムにデプロイまたは再デプロイする必要があることを示します。
.skipdeploy	ユーザー生成	アプリケーションの自動デプロイを無効にします。展開形式のコンテンツの自動デプロイメントを一時的にブロックする方法として役に立ち、不完全なコンテンツの編集がライブにプッシュするリスクを防ぎます。スキャナーは zip 形式のコンテンツに対する処理中の変更を検出し、完了するまで待機しますが、zip 形式のコンテンツとともに使用できます。
.isdeploying	システム生成	デプロイメントの開始を示します。デプロイメントプロセスが完了すると、マーカーファイルが削除されます。

ファイル名接尾辞	生成	説明
.deployed	システム生成	コンテンツがデプロイされたことを示します。このファイルが削除された場合、コンテンツはアンデプロイされます。
.failed	システム生成	デプロイメントの失敗を示します。マーカーファイルには、失敗の原因に関する情報が含まれます。マーカーファイルが削除されると、コンテンツは再度自動デプロイメントに表示されます。
.isundeploying	システム生成	**.deployed** ファイルの削除に対する応答を示します。コンテンツはアンデプロイされ、完了時にマーカーは自動的に削除されます。
.undeployed	システム生成	コンテンツがアンデプロイされたことを示します。マーカーファイルを削除しても、コンテンツの再デプロイメントには影響はありません。
.pending	システム生成	デプロイメントの指示が、検出された問題の解決を保留しているサーバーに送信されることを示します。このマーカーはグローバルデプロイメントロードブロックとして機能します。この状態が存在する場合、スキャナーは他のコンテンツをデプロイまたはアンデプロイするようサーバーに指示しません。

バグの報告

10.5.6. デプロイメントスキャナー属性のリファレンス

*デプロイメントスキャナーには、管理 CLI に公開された以下の属性が含まれ、**write-attribute** 操作を使用して設定できます。設定オプションの詳細は、「管理 CLI でのデプロイメントスキャナーの設定」のトピックを参照してください。*

表10.2 デプロイメントスキャナーの属性

名前	詳細	タイプ	デフォルト値
auto-deploy-exploded	**.dodeploy** マーカーファイルなしで、展開形式のコンテンツの自動デプロイメントを許可します。基本的な開発シナリオに対してのみ推奨され、開発者またはオペレーティングシステムによる変更中に、展開形式のアプリケーションデプロイメントが行われないようにします。	ブール値	False
auto-deploy-xml	**.dodeploy** マーカーファイルなしでの XML コンテンツの自動デプロイメントを許可します。	ブール値	True

名前	詳細	タイプ	デフォルト値
auto-deploy-zipped	**.dodeploy** マーカーファイルなしでの zip 形式のコンテンツの自動デプロイメントを許可します。	ブール値	True
deployment-timeout	デプロイメントスキャナーでデプロイメントをキャンセルするまでのデプロイメント試行許可時間 (秒単位)。	Long	600
path	スキャンする実際のファイルシステムパスを定義します。**relative-to** 属性を指定すると、**path** の値は、そのディレクトリーまたはパスに対する相対追加として機能します。	文字列	デプロイメント
relative-to	サーバー設定 XML ファイルの **paths** セクションで定義されたファイルシステムパスへの参照。	文字列	jboss.server.base.dir
scan-enabled	**scan-interval** により起動時にアプリケーションの自動スキャンを許可します。	ブール値	True
scan-interval	リポジトリーのスキャン間隔（ミリ秒単位）。1 未満の値の場合、スキャナーは起動時にのみ動作します。	Int	5000

バグの報告

10.5.7. デプロイメントスキャナーの設定

デプロイメントスキャナーは管理コンソールまたは管理 CLI を使用して設定できます。新しいデプロイメントスキャナーを作成したり、既存のスキャナー属性を管理したりできます。これには、スキャンの間隔、デプロイメントフォルダーの場所、およびデプロイメントをトリガーするアプリケーションファイルタイプが含まれます。

バグの報告

10.5.8. 管理 CLI でのデプロイメントスキャナーの設定

前提条件

- *「管理 CLI の起動」*

概要

*デプロイメントスキャナーを設定する方法は複数ありますが、管理 CLI を使用してバッチスクリプトを使用するか、リアルタイムで属性を公開および変更できます。**read-attribute** および **write-attribute** global コマンドライン操作を使用すると、デプロイメントスキャナーの動作を変更できます。デプロイメントスキャナー属性の詳細は、「デプロイメントスキャナー属性のリファレンス」トピックで定義されています。*

デプロイメントスキャナーは JBoss EAP 6 のサブシステムで、**standalone.xml** で表示できます。

例10.7 Excerpt from **standalone.xml**

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
    <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-
interval="5000"/>
</subsystem>
```

手順10.12 デプロイメントスキャナーの設定

1. 設定するデプロイメントスキャナー属性の決定
   管理 CLI でデプロイメントスキャナーを設定するには、最初に正しい属性名を公開する必要があります。これは、ルートノードのいずれかで **read-resources** 操作を使用するか、**cd** コマンドを使用してサブシステムの子ノードに変更します。このレベルで **ls** コマンドを使用して、属性を表示できます。

   - **read-resource** 操作を使用したデプロイメントスキャナー属性の公開
     read-resource 操作を使用して、デフォルトのデプロイメントスキャナーリソースで定義された属性を公開します。

     例10.8 **read-resource** 出力サンプル

     ```
     [standalone@localhost:9999 /]/subsystem=deployment-
     scanner/scanner=default:read-resource
     {
         "outcome" => "success",
         "result" => {
             "auto-deploy-exploded" => false,
             "auto-deploy-xml" => true,
             "auto-deploy-zipped" => true,
             "deployment-timeout" => 600,
             "path" => "deployments",
             "relative-to" => "jboss.server.base.dir",
             "scan-enabled" => true,
             "scan-interval" => 5000
         }
     }
     ```

   - **ls** コマンドを使用したデプロイメントスキャナー属性の公開
     ls コマンドに **-l** オプションの引数を指定して、サブシステムノード属性、値、およびタイプを含む結果の表を表示します。**ls --help** と入力して、CLI ヘルプエントリーを公開すると、ls コマンドとその引数を確認できます。管理 CLI のヘルプメニューの詳細は、「管理 CLI でのヘルプの取得」 のトピックを参照してください。

     例10.9 **ls -l** の出力例

     ```
     [standalone@localhost:9999 /] ls -l /subsystem=deployment-
     scanner/scanner=default
     ATTRIBUTE           VALUE               TYPE
     auto-deploy-exploded false               BOOLEAN
     ```

```
auto-deploy-xml      true                BOOLEAN
auto-deploy-zipped   true                BOOLEAN
deployment-timeout   600                 LONG
path                 deployments         STRING
relative-to          jboss.server.base.dir STRING
scan-enabled         true                BOOLEAN
scan-interval        5000                INT
```

2. ***write-attribute** 操作を使用したデプロイメントスキャナーの設定*
   *変更する属性の名前を決定したら、**write-attribute** を使用して属性名と、書き込む新しい値を指定します。以下の例はすべて子ノードレベルで実行されます。これは、**cd** コマンドおよびタブ補完を使用してデフォルトのスキャナーノードを公開および変更することでアクセスできます。*

```
[standalone@localhost:9999 /] cd subsystem=deployment-scanner/scanner=default
```

   a. *展開されたコンテンツの自動デプロイメントの有効化*
      ***write-attribute** 操作を使用して、展開形式のアプリケーションコンテンツの自動デプロイメントを有効にします。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-exploded,value=true)
{"outcome" => "success"}
```

   b. ***XML** コンテンツの自動デプロイメントの無効化*
      ***write-attribute** 操作を使用して、**XML** アプリケーションコンテンツの自動デプロイメントを無効にします。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-xml,value=false)
{"outcome" => "success"}
```

   c. ***zip** 形式のコンテンツの自動デプロイメントの無効化*
      ***write-attribute** コマンドを使用して、**zip** 形式のアプリケーションコンテンツの自動デプロイメントを無効にします。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=auto-deploy-zipped,value=false)
{"outcome" => "success"}
```

   d. *パス属性の設定*
      ***write-attribute** 操作を使用してパス属性を変更し、監視するデプロイメントスキャナーの新しいパス名の **newpathname** 値を置き換えます。変更を反映するには、サーバーのリロードが必要なことに注意してください。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=path,value=newpathname)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
```

```
        "process-state" => "reload-required"
    }
}
```

e. *相対パス属性の設定*
write-attribute *操作を使用して、設定 XML ファイルの* **paths** *セクションで定義されたファイルシステムパスへの相対参照を変更します。変更を反映するには、サーバーのリロードが必要なことに注意してください。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=relative-
to,value=new.relative.dir)
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

f. *デプロイメントスキャナーの無効化*
scan -enabled *の値を* **false** *に設定して、* **Write-attribute** *操作を使用してデプロイメントスキャナーを無効にします。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-
enabled,value=false)
{"outcome" => "success"}
```

g. *スキャン間隔の変更*
write-attribute *操作を使用してスキャン間隔を 5000 ミリ秒から 10000 ミリ秒に変更します。*

```
[standalone@localhost:9999 scanner=default] :write-attribute(name=scan-
interval,value=10000)
{"outcome" => "success"}
```

結果

設定の変更が、デプロイメントスキャナーに保存されます。

バグの報告

10.5.9. カスタムデプロイメントスキャナーの定義

前提条件

- *「管理 CLI の起動」*

概要

管理コンソールまたは管理 CLI を使用して新しいデプロイメントスキャナーを追加できます。デプロイメントを確認するためにスキャンする新しいディレクトリーを定義します。デフォルトのデプロイメントスキャナーは **EAP_HOME/standalone/deployments/** *を監視します。既存のデプロイメントスキャナーの設定に関する詳細は、「管理 CLI でのデプロイメントスキャナーの設定」を参照してください。*

手順10.13 管理 CLI でのカスタムデプロイメントスキャナーの定義

1. *管理 CLI の **add** 操作を使用してデプロイメントスキャナーを追加します。*

```
[standalone@localhost:9999 /] /subsystem=deployment-scanner/scanner=new-scanner:add(path=new_deployment_dir,relative-to=jboss.server.base.dir,scan-interval=5000)
{"outcome" => "success"}
```

注記

指定のディレクトリーがすでに存在しないと、このコマンドに失敗し、エラーが発生します。

2. *新しいデプロイメントスキャナーが **standalone.xml** ファイルおよび管理インターフェースに表示されるようになりました。*

*例10.10 からの抜粋 **standalone.xml***

```
<subsystem xmlns="urn:jboss:domain:deployment-scanner:1.1">
 <deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"/>
 <deployment-scanner name="new-scanner" path="new_deployment_dir" relative-to="jboss.server.base.dir" scan-interval="5000"/>
</subsystem>
```

3. *デプロイメント用に指定のディレクトリーがスキャンされるようになりました。デプロイメントスキャナーを使用してアプリケーションをデプロイする方法は、「デプロイメントスキャナーを使用してスタンドアロンサーバーインスタンスにアプリケーションをデプロイ」を参照してください。*

結果

新しいデプロイメントスキャナーが定義され、デプロイメントの監視が行われています。

バグの報告

10.6. MAVEN でのデプロイ

10.6.1. Maven によるアプリケーションデプロイメントの管理

Maven を使用してアプリケーションをデプロイすると、デプロイメントのサイクルを既存の開発ワークフローの一部として取り入れることができます。

バグの報告

10.6.2. Maven によるアプリケーションのデプロイ

前提条件

- *「JBoss EAP 6 の起動」*

概要

*このタスクでは、Maven を使用してアプリケーションをデプロイする方法を説明します。提供されている例では、JBoss EAP 6 のクイックスタートコレクションにある **jboss-helloworld.war** アプリケーションを使用します。**helloworld** プロジェクトには、**jboss-as-maven-plugin** を初期化する POM ファイルが含まれています。このプラグインは、アプリケーションサーバーとの間でアプリケーションをデプロイおよびアンデプロイする簡単な操作を提供します。*

手順10.14 Maven によるアプリケーションのデプロイ

1. *ターミナルセッションを開き、**quickstart** サンプルが含まれるディレクトリーに移動します。*

   例10.11 helloworld アプリケーションディレクトリーへの移動

   ```
   [localhost]$ cd /QUICKSTART_HOME/helloworld
   ```

2. *Maven デプロイコマンドを実行してアプリケーションをデプロイします。アプリケーションがすでに実行されている場合、これは再デプロイされます。*

   ```
   [localhost]$ mvn package jboss-as:deploy
   ```

3. *結果を確認します。*

   - *デプロイメントは、ターミナルウィンドウで操作ログを参照して確認できます。*

     例10.12 Maven での helloworld アプリケーションの確認

     ```
     [INFO] ------------------------------------------------------------------------
     [INFO] BUILD SUCCESS
     [INFO] ------------------------------------------------------------------------
     [INFO] Total time: 32.629s
     [INFO] Finished at: Fri Mar 14 09:09:50 EDT 2014
     [INFO] Final Memory: 23M/204M
     [INFO] ------------------------------------------------------------------------
     ```

   - *また、デプロイメントは、アクティブアプリケーションサーバーインスタンスのステータスストリームで確認することもできます。*

     例10.13 アプリケーションサーバーでの helloworld アプリケーションの確認

     ```
     09:09:49,167 INFO  [org.jboss.as.repository] (management-handler-thread - 1)
     JBAS014900: Content added at location
     /home/username/EAP_HOME/standalone/data/content/32/4b4ef9a4bbe7206d367
     4a89807203a2092fc70/content
     09:09:49,175 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-7)
     JBAS015876: Starting deployment of "jboss-helloworld.war" (runtime-name:
     "jboss-helloworld.war")
     09:09:49,563 INFO  [org.jboss.weld.deployer] (MSC service thread 1-8)
     JBAS016002: Processing weld deployment jboss-helloworld.war
     09:09:49,611 INFO  [org.jboss.weld.deployer] (MSC service thread 1-1)
     JBAS016005: Starting Services for CDI deployment: jboss-helloworld.war
     09:09:49,680 INFO  [org.jboss.weld.Version] (MSC service thread 1-1) WELD-
     ```

```
000900 1.1.17 (redhat)
09:09:49,705 INFO  [org.jboss.weld.deployer] (MSC service thread 1-2)
JBAS016008: Starting weld service for deployment jboss-helloworld.war
09:09:50,080 INFO  [org.jboss.web] (ServerService Thread Pool -- 55)
JBAS018210: Register web context: /jboss-helloworld
09:09:50,425 INFO  [org.jboss.as.server] (management-handler-thread - 1)
JBAS018559: Deployed "jboss-helloworld.war" (runtime-name : "jboss-
helloworld.war")
```

結果

アプリケーションが、アプリケーションサーバーにデプロイされます。

バグの報告

10.6.3. Maven によるアプリケーションのアンデプロイ

前提条件

- *「JBoss EAP 6 の起動」*

概要

*このタスクでは、Maven を使用してアプリケーションをアンデプロイする方法を示します。提供されている例では、JBoss EAP 6 のクイックスタートコレクションにある **jboss-helloworld.war** アプリケーションを使用します。**helloworld** プロジェクトには、**jboss-as-maven-plugin** を初期化する POM ファイルが含まれています。このプラグインは、アプリケーションサーバーとの間でアプリケーションをデプロイおよびアンデプロイする簡単な操作を提供します。*

手順10.15 Maven によるアプリケーションのアンデプロイ

1. *ターミナルセッションを開き、quickstart サンプルが含まれるディレクトリーに移動します。*

   例10.14 helloworld アプリケーションディレクトリーへの移動

   ```
   [localhost]$ cd /QUICKSTART_HOME/helloworld
   ```

2. *Maven アンデプロイコマンドを実行してアプリケーションをアンデプロイします。*

   ```
   [localhost]$ mvn jboss-as:undeploy
   ```

3. *結果を確認します。*

   - *アンデプロイメントは、ターミナルウィンドウの操作ログを参照して確認できます。*

     例10.15 Maven による helloworld アプリケーションのアンデプロイの確定

     ```
     [INFO] ------------------------------------------------------------------------
     [INFO] BUILD SUCCESSFUL
     [INFO] ------------------------------------------------------------------------
     [INFO] Total time: 1 second
     ```

```
[INFO] Finished at: Mon Oct 10 17:33:02 EST 2011
[INFO] Final Memory: 11M/212M
[INFO] ------------------------------------------------------------------------
```

- また、アンデプロイメントは、アクティブアプリケーションサーバーインスタンスのステータスストリームで確認することもできます。

例10.16 アプリケーションサーバーによる helloworld アプリケーションのアンデプロイの確定

```
09:51:40,512 INFO  [org.jboss.web] (ServerService Thread Pool -- 69)
JBAS018224: Unregister web context: /jboss-helloworld
09:51:40,522 INFO  [org.jboss.weld.deployer] (MSC service thread 1-3)
JBAS016009: Stopping weld service for deployment jboss-helloworld.war
09:51:40,536 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-1)
JBAS015877: Stopped deployment jboss-helloworld.war (runtime-name: jboss-
helloworld.war) in 27ms
09:51:40,621 INFO  [org.jboss.as.repository] (management-handler-thread - 10)
JBAS014901: Content removed from location
/home/username/EAP_HOME/jboss-eap-
6.4/standalone/data/content/44/e1f3c55c84b777b0fc201d69451223c09c9da5/con
tent
09:51:40,621 INFO  [org.jboss.as.server] (management-handler-thread - 10)
JBAS018558: Undeployed "jboss-helloworld.war" (runtime-name: "jboss-
helloworld.war")
```

結果

アプリケーションが、アプリケーションサーバーからアンデプロイされます。

バグの報告

10.7. JBOSS EAP 6 にデプロイされたアプリケーションの順序制御

JBoss EAP 6 では、サーバーの起動時にアプリケーションのデプロイメント順序を細かく制御できます。複数の ear ファイルに存在するアプリケーションのデプロイメント順序を厳格に有効にし、再起動後に順序を永続化できます。

手順10.16 EAP 6.0 におけるデプロイメント順序の制御

1. サーバーの起動または停止時に、順番にアプリケーションをデプロイおよびアンデプロイする CLI スクリプトを作成します。
2. CLI はバッチモードの概念もサポートします。バッチモードでは、コマンドおよび操作をグループ化し、それらをアトミックユニットとして一緒に実行できます。少なくとも 1 つのコマンドまたは操作が失敗すると、バッチで正常に実行された他のコマンドおよび操作はすべてロールバックされます。

手順10.17 EAP 6.1 以降におけるデプロイメント順序の制御

EAP 6.1 以降では、Inter Deployment Dependencies を使用すると、トップレベルのデプロイメント間で依存関係を宣言できます。

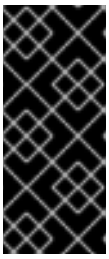
1. *app.ear/META-INF* フォルダーの*jboss-all.xml* ファイルを作成し（存在しない場合）。*app.ear* は、デプロイする別のアプリケーションアーカイブに依存するアプリケーションアーカイブです。

2. 以下のように、このファイルに *jboss-deployment-dependencies* エントリーを作成します。以下の一覧で、*framework.ear* は、*app.ear* アプリケーションアーカイブの前にデプロイされる必要がある依存関係アプリケーションアーカイブであることに注意してください。

```
<jboss umlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>
```

注記

*デプロイメントのランタイム名を **jboss-all.xml** ファイルの依存関係名として使用できます。*



重要

***jboss-all.xml** ファイルを使用すると、サーバーが検出しない依存関係を宣言できますが、厳密な順序付け機能ではありません。JBoss EAP は、**jboss-all.xml** ファイルに指定されたすべての依存関係がすでにデプロイ済みまたは利用できることを前提とします。不足している依存関係がある場合、JBoss EAP はそれらの依存関係を自動的にデプロイせず、デプロイメントは失敗します。*

バグの報告

10.8. デプロイされたコンテンツ用のカスタムディレクトリーの定義

JBoss EAP には、サーバーがデプロイされたコンテンツを格納するために使用する場所を定義するオプションがあります。

スタンドアロンモードでコンテンツをデプロイするカスタムディレクトリーの定義

*デフォルトでは、スタンドアロンモードにデプロイされているコンテンツは **EAP_HOME/standalone/data/content** ディレクトリーに保存されます。*

- *この場所を変更するには、サーバーの起動時に **-Djboss.server.deploy.dir** 引数を渡します。*

```
./standalone.sh -Djboss.server.deploy.dir=/path/to/new_deployed_content
```

- *選択する場所は JBoss EAP インスタンスすべてで一意になる必要があります。*

注記

***jboss.server.deploy.dir** 管理コンソールまたは管理 CLI を使用してデプロイされたコンテンツの格納に使用されるディレクトリーを指定します。デプロイメントスキャナーによって監視されるカスタムデプロイメントディレクトリーを定義する場合は、「カスタムデプロイメントスキャナーの定義」 を参照してください。*

ドメインモードでコンテンツをデプロイするカスタムディレクトリーの定義

デフォルトでは、ドメインモードにデプロイされたコンテンツは **EAP_HOME/domain/data/content** ディレクトリーに保存されます。

- この場所を変更するには、ドメインの起動時に **-Djboss.domain.deployment.dir** 引数を渡します。

```
./domain.sh -Djboss.domain.deployment.dir=/path/to/new_deployed_content
```

- 選択する場所は JBoss EAP インスタンスすべてで一意になる必要があります。

バグの報告

10.9. デプロイメント記述子の上書き

JBoss EAP 6.1 以降では、実行時にデプロイメント記述子、JAR、クラス、JSP ページ、およびその他のファイルをオーバーライドできます。**デプロイメントオーバーレイ** は、アーカイブで上書きする必要のあるファイルのルールセットを表します。また、上書きされたファイルの代わりに使用する必要のある新しいファイルへのリンクも提供します。オーバーライドされるファイルがデプロイメントアーカイブに存在しない場合は、デプロイメントに戻ります。

手順10.18 管理 CLI を使用したデプロイメント記述子の上書き

以下の手順は、**app.war** という名前のデプロイされたアプリケーションがすでにあり、その **WEB-INF/web.xml** ファイルを **/home/user/web.xml** にある別の **web.xml** ファイルで上書きする必要があることを前提としています。

1. デプロイメントオーバーレイを追加し、そこにコンテンツを追加します。これには、以下の 2 つの方法があります。
   - DMR ツリーの使用
     a. `/deployment-overlay=myoverlay:add`
     b. `/deployment-overlay=myoverlay/content=WEB-INF\/web.xml:add(content={url=file:///home/user/web.xml})`

     また、2 番目のステートメントを使用してコンテンツルールを追加できます。
   - 便利なメソッドの使用

     ```
     deployment-overlay add --name=myoverlay --content=WEB-INF/web.xml=/home/user/web.xml
     ```
2. オーバーレイをデプロイメントアーカイブにリンクします。これには、以下の 2 つの方法があります。
   - DMR ツリーの使用

     ```
     /deployment-overlay=myoverlay/deployment=app.war:add
     ```
   - 便利なメソッドの使用

     ```
     deployment-overlay link --name=myoverlay --deployments=app.war
     ```

複数のアーカイブ名を指定するには、コンマで区切ります。

デプロイメントアーカイブ名はサーバーに存在しないことに注意してください。名前を指定しますが、まだ実際のデプロイメントにリンクされていません。

3. アプリケーションの再デプロイ

```
/deployment=app.war:redeploy
```

バグの報告

10.10. ROLLOUT PLAN

10.10.1. ロールアウト計画

ドメインまたはホストレベルのリソースでターゲットとなる操作は、複数のサーバーに影響する可能性があります。このような操作には、サーバーに適用される操作の順序を説明する ロールアウト計画 や、一部のサーバーで実行に失敗した場合に操作を元に戻すかどうかの詳細を示すポリシーなどが含まれます。

例10.17 ロールアウト計画の CLI 形式

```
rollout (id=plan_id | server_group_list) [rollback-across-groups]

server_group_list := server_group [ (sequence_separator | concurrent_separator)
server_group ]
sequence_separator := ','
concurrent_separator := '^'
server_group := server_group_name [group_policy_list]
group_policy_list := '(' policy_property_name=policy_property_value (,
policy_property_name=policy_property_value)* ')'
policy_property_name := 'rolling-to-servers' | 'max-failed-servers' | 'max-failure-percentage'
```

policy_property_value の値はプロパティーによって異なります。ブール値や整数などにすることができます。

ロールアウト計画は長く、複雑になる可能性があります。ただし、ドメイン管理モデルの一部として保存し、名前（上記の定義の ID）を使用してコマンドおよび操作から後で参照することができます。保存されたロールアウト計画は **rollout-plan** コマンドを使用して管理されます。

例10.18 **rollout-plan** コマンドで管理されるロールアウト計画

```
rollout-plan add --name=my-plan --content={rollout main-server-group^other-server-
group}
:write-attribute(name=my-attr,value=my-value){rollout id=my-plan}
```

例10.19 保存されたロールアウト計画の使用

```
rollout-plan add --name=my-plan --content={rollout main-server-group^other-server-
group}
:write-attribute(name=my-attr,value=my-value){rollout id=my-plan}
```

バグの報告

10.10.2. ロールアウト計画を使用した操作

構造は、**rollout-plan** 内の **operation** です。

```
{
"operation" => "write-core-threads",
    "address" => [
        ("profile" => "production"),
        ("subsystem" => "threads"),
        ("bounded-queue-thread-pool" => "pool1")
    ],
    "count" => 0,
    "per-cpu" => 20,
    "operation-headers" => {
        "rollout-plan" => {
            "in-series" => [
                {
                    "concurrent-groups" => {
                        "groupA" => {
                            "rolling-to-servers" => true,
                            "max-failure-percentage" => 20
                        },
                        "groupB" => undefined
                    }
                },
                {
                   "server-group" => {
                        "groupC" => {
                            "rolling-to-servers" => false,
                            "max-failed-servers" => 1
                        }
                    }
                },
                {
                    "concurrent-groups" => {
                        "groupD" => {
                            "rolling-to-servers" => true,
                            "max-failure-percentage" => 20
                        },
                        "groupE" => undefined
                    }
                }
            ],
            "rollback-across-groups" => true
        }
    }
}
```

rollout-plan は、**operation-headers** 構造内で入れ子になっています。構造のルートノードは、2 つの子を許可します。

- **in-series** - シリーズで実行するステップの一覧。各ステップは次のステップが実行される前に完了します。各ステップには、1 つ以上のサーバーグループのサーバーへの操作の適用が含まれます。リストの各要素の詳細は、以下を参照してください。

- **rollback-across-groups** - 1 つのサーバーグループのすべてのサーバーで操作をロールバックする必要があるかどうかを示すブール値。すべてのサーバーグループでロールバックをトリガーします。これは任意の設定で、デフォルトは false に設定されます。

in-series ノード下のリストの各要素には、以下の構造の 1 つまたは他の構造が必要です。

- **concurrent-groups** - サーバーグループに操作を適用する方法を制御するサーバーグループ名のマップ。マップの各サーバーグループに対して、操作を同時に適用することができます。server-group ポリシー設定の詳細は、以下を参照してください。

- **server-group** - サーバーグループ名のキー/値から、そのサーバーグループに操作を適用する方法を制御するポリシーへの単一のキー/値マッピング。ポリシー設定の詳細は、以下を参照してください。（注意：単一のエントリーを持つ、これと「concurrent-groups」マップとの間には、プラン実行の違いはありません。）

サーバーグループのサーバーに操作がどのように適用されるかを制御するポリシーには、以下の要素があります（それぞれは任意です）。

- **rolling-to-servers** - **true** に設定すると、操作は連続してグループ内の各サーバーに適用されます。false を指定した場合、操作はグループのサーバーに同時に適用されます。

- **max-failed-servers** - グループのすべてのサーバーで元に戻す前に、操作の適用に失敗したサーバーの最大数を取る整数。指定がない場合のデフォルト値はゼロです。つまり、サーバーがグループ全体でロールバックをトリガーします。

- **max-failure-percentage** - 0 から 100 までの整数。グループ内のすべてのサーバーで元に戻す前に、操作の適用に失敗したサーバーの合計数の最大パーセンテージを取ります。指定がない場合のデフォルト値はゼロです。つまり、サーバーがグループ全体でロールバックをトリガーします。

max-failed-servers と **max-failure-percentage** の両方がゼロ以外の値に設定された場合、**max-failure-percentage** が優先されます。

上記の（簡単な）の例を見ると、ドメインのサーバーへの操作の適用は **3** つのフェーズで実行されます。あるサーバーグループのポリシーによってサーバーグループ全体で操作のロールバックが引き起こされると、他のサーバーグループもすべてロールバックされます。**3** つのフェーズは次のとおりです。

1. サーバーグループ **groupA** と **groupB** には同時に操作が適用されます。**groupA** のサーバーには操作が順次適用され、**groupB** のすべてのサーバーは操作を同時に処理します。**groupA** で操作の適用に失敗したサーバーが **20 %** を超えると、グループ全体でロールバックが実行されます。**groupB** で操作を適用できなかったサーバーがあると、そのグループ全体でロールバックが実行されます。

2.

groupA および groupB のすべてのサーバーが完了すると、groupC のサーバーに操作が適用されます。これらのサーバーは操作を同時に処理します。groupC で操作を適用できなかったサーバーが複数ある場合は、グループ全体でロールバックが実行されます。

3. *groupC のすべてのサーバーが完了すると、サーバーグループ groupD と groupE の操作は同時に適用されます。groupD のサーバーには操作が順次適用され、groupE のすべてのサーバーは操作を同時に処理します。groupD で操作の適用に失敗したサーバーが 20 % を超えると、グループ全体でロールバックが実行されます。groupE で操作を適用できなかったサーバーがあると、そのグループ全体でロールバックが実行されます。*

デフォルトのロールアウト計画

複数のサーバーに影響する操作はすべてロールアウト計画を使用して実行されます。ただし、実際には操作リクエストでロールアウト計画を指定する必要はありません。rollout-plan が指定されていない場合、デフォルトのプランが生成されます。プランには以下の特徴があります。

- *ハイレベルなフェーズは 1 つのみです。操作に影響するすべてのサーバーグループには同時に操作が適用されます。*
- *各サーバーグループ内では、操作がすべてのサーバーに同時に適用されます。*
- *サーバーグループのいずれかのサーバーに操作を適用できないと、グループ全体でロールバックが実行されます。*
- *あるサーバーグループに操作を適用できないと、他のすべてのサーバーグループでロールバックが実行されます。*

バグの報告

10.10.3. デプロイメント計画の作成

JBoss EAP 6 のクラスター化ドメインにアプリケーションをデプロイするデプロイメントプランの作成方法

1. *rolling-to-servers=true を指定して CLI を使用してロールアウトデプロイメントプランを作成します。パッケージは、サーバーグループの各サーバーに順次デプロイされます。*

*シリアルデプロイメント用の **CLI** デプロイメントプランの例を以下に示します。*

```
deploy ClusterWebApp.war --name=ClusterWebApp.war --runtime-
name=ClusterWebApp.war --server-groups=ha-server-group --headers={rollout ha-
server-group(rolling-to-servers=true)}
```

2. *すべてのサーバーグループにロールアウト計画を適用するには、マスターホストの各 **server-group** の名前を指定する必要があります。*

```
deploy /NotBackedUp/PREVIOUS/ALLWAR/ClusterWebApp.war --
name=ClusterWebApp.war --runtime-name=ClusterWebApp.war --server-
groups=main-server-group,other-server-group --headers={rollout main-server-
group(rolling-to-servers=true),other-server-group(rolling-to-servers=true)}
```

バグの報告

第11章 サブシステムの設定

11.1. サブシステム設定の概要

はじめに

JBoss EAP 6 は簡単な設定を使用し、ドメインごとまたはスタンドアロンサーバーごとに **1** つの設定ファイルを使用します。ドメインモードでは、各ホストコントローラーにも個別のファイルが存在します。設定の変更は自動的に保持されるため、**XML** 設定ファイルは手動で編集しないでください。設定はスキャンされ、管理 **API** によって自動的に上書きされます。コマンドラインベースの管理 **CLI** および **Web** ベースの管理コンソールを使用すると、**JBoss EAP 6** の各側面を設定できます。

JBoss EAP 6 は、モジュラークラスローディングの概念に基づいて構築されます。**Platform** が提供する各 **API** またはサービスはモジュールとして実装され、必要に応じてロードおよびアンロードされます。ほとんどのモジュールには、サブシステムと呼ばれる設定可能な要素が含まれています。サブシステム設定情報は、管理対象ドメインの場合は **EAP_HOME/domain/configuration/domain.xml**、スタンドアロンサーバーの場合は **EAP_HOME/standalone/configuration/standalone.xml** に保存されます。サブシステムの多くには、以前のバージョンの **JBoss EAP** でデプロイメント記述子を介して設定された設定の詳細が含まれます。

サブシステム設定スキーマ

各サブシステムの設定は **XML** スキーマで定義されます。設定スキーマは、インストールの **EAP_HOME/docs/schema/** ディレクトリーにあります。

バグの報告

第12章 ロギングサブシステム

12.1. はじめに

12.1.1. 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両方で設定可能なロギング機能を提供します。**logging** サブシステムは **JBoss LogManager** をベースとしており、**JBoss Logging** だけでなくサードパーティーのアプリケーションロギングフレームワークを複数サポートします。

logging サブシステムはログカテゴリーおよび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（ディスクへの書き込み、コンソールへの送信など）を定義します。

ロギングプロファイルを使用すると、一意に名前付きのロギング設定のセットを作成し、他のロギング設定とは関係なくアプリケーションに割り当てることができます。ロギングプロファイルの設定はメインの **logging** サブシステムとほぼ同じです。

バグの報告

12.1.2. JBoss LogManager でサポートされるアプリケーションロギングフレームワーク

JBoss LogManager は次のロギングフレームワークをサポートします。

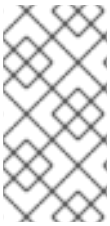
- **JBoss Logging - JBoss EAP 6** に含まれます
- **Apache Commons Logging - http://commons.apache.org/logging/**
- **Simple Logging Facade for Java(SLF4J)- http://www.slf4j.org/**
- **Apache log4j - http://logging.apache.org/log4j/1.2/**
- **Java SE Logging (java.util.logging) - http://download.oracle.com/javase/6/docs/api/java/util/logging/package-summary.html**

JBoss LogManager では以下の API がサポートされます。

- *java.util.logging*
- *JBoss Logging*
- *Log4j*
- *SLF4J*
- *commons-logging*

JBoss LogManager では以下の SPI もサポートされます。

- *java.util.logging Handler*
- *Log4j Appender*



注記

Log4j API と Log4J Appender を使用している場合、オブジェクトは渡される前に string に変換されます。

バグの報告

12.1.3. 起動時のロギング

JBoss EAP の起動中に、Java 環境と各サービスの起動に関するログエントリーが出力されます。ログは、トラブルシューティング時に役に立ちます。デフォルトでは、すべてのログエントリーはファ

イル **server.log** に書き込まれます。これはランタイムモードに依存する場所です。

スタンドアロンモード

EAP_HOME/standalone/log/server.log

ドメインモード

EAP_HOME/domain/servers/SERVER_NAME/log/server.log

起動ロギングの設定は、設定ファイル **logging.properties** に指定されます。これはランタイムモードに依存する場所です。

スタンドアロンモード

EAP_HOME/standalone/configuration/logging.properties

ドメインモード

ドメインモードでは、ドメインコントローラーと各サーバーの **logging.properties** ファイルがあります。

ドメインコントローラー： **EAP_HOME/domain/configuration/logging.properties**

Server: EAP_HOME/domain/servers/SERVER_NAME/data/logging.properties

設定ファイルは **logging** サブシステムが起動し、引き継がるまで **logging.properties** がアクティブになります。

警告

logging.properties ファイルは、直接編集する必要があるユースケースが不明な場合を除き、直接編集しないことが推奨されます。サポートケースを作成する前に、サポートケースを作成することが推奨されます。

logging.properties ファイルに手動で行った変更は起動時に上書きされます。

バグの報告

12.1.4. 起動エラーの表示

JBoss EAP をトラブルシューティングする場合、最初に行う手順の **1** つとなるよう、起動中に発生したエラーをチェックする必要があります。起動時のエラーを表示する方法は **2** つあり、どちらも利点があります。各メソッドにより、起動時に発生したエラーの一覧が表示されます。提供される情報を使用して原因を診断し、解決します。トラブルシューティングに関しては、**Red Hat** カスタマーポータルにお問い合わせください。

- **server.log** ログファイルを確認します。

  この方法では、各エラーメッセージおよび関連するメッセージを確認でき、エラーが発生した**理由**の詳細を知ることができます。また、エラーメッセージをプレーンテキスト形式で表示することもできます。

- **JBoss EAP 6.4** から、管理 **CLI** コマンドの **read-boot-errors** を使用します。

  この方法では、サーバーのファイルシステムにアクセスする必要がありません。これは管理 **CLI** コマンドであるため、スクリプトで使用できます。たとえば、複数の **JBoss EAP** インスタンスを起動し、起動時に発生したエラーをチェックするスクリプトを作成できます。

手順**12.1** **server.log** でエラーの有無を確認する

1. ファイルビューアーで **server.log** ファイルを開きます。

2. ファイルの最後に移動します。

3. 最新の起動シーケンスの開始 を示すメッセージ識別子の **backward015899** を検索します。

4. ログのその位置から **ERROR** を前方検索します。各検索一致箇所には、エラーの説明が示され、関連するモジュールがリストされます。

例**12.1 Error Description from server.log**

以下は、**server.log** ログファイルのエラー説明の例です。

```
13:23:14,281 ERROR [org.apache.coyote.http11.Http11Protocol] (MSC service thread 1-4)
JBWEB003043: Error initializing endpoint: java.net.BindException: Address already in use
/127.0.0.1:8080
```

手順**12.2** 管理 **CLI** より起動エラーの一覧表示

- 以下の管理 **CLI** コマンドを実行します。

```
/core-service=management:read-boot-errors
```

起動中に発生したすべてのエラーがリストされます。

各エラーのタイムスタンプは **Java** メソッド **currentTimeMillis()** を使用します。これは、現在の時間と午前 **0** 時から **1970 UTC**（協定世界時）の間で測定される違いです。

例**12.2 read-boot-errors** コマンドの出力

```
{
"outcome" => "success",
"result" => [{
    "failed-operation" => {
        "operation" => "add",
        "address" => [
            ("subsystem" => "web"),
            ("connector" => "http")
        ]
    },
    "failure-timestamp" => 1417560953245L,
```

```
    "failure-description" => "{\"JBAS014671: Failed services\" => {\"jboss.web.connector.http\" =>
\"org.jboss.msc.service.StartException in service jboss.web.connector.http: JBAS018007: Error
starting web connector
Caused by: LifecycleException:  JBWEB000023: Protocol handler initialization failed\"}}",
    "failed-services" => {"jboss.web.connector.http" => "org.jboss.msc.service.StartException in
service jboss.web.connector.http: JBAS018007: Error starting web connector
Caused by: LifecycleException:  JBWEB000023: Protocol handler initialization failed"}
}]
}
```

バグの報告

12.1.5. ガベッジコレクションロギング

ガベッジコレクションロギングは、すべてのガベッジコレクションのアクティビティーをプレーンテキストのログファイルに記録します。これらのログファイルは診断を行うのに便利です。**JBoss EAP 6** より、ガベッジコレクションのロギングは **IBM Java Development Kit 以外** のすべてのサポートされる構成では、スタンドアロン モードではデフォルトで有効になっています。

logging は **EAP_HOME/standalone/log/gc.log.**桁 のファイルに出力されます。ログローテーションが有効化され、ログファイルの数は **5** に制限され、各ファイルは最大 **3 MiB** に制限されます。

バグの報告

12.1.6. 暗黙的なロギング **API** の依存関係

JBoss EAP 6 の **logging** サブシステムには、コンテナーが暗黙的なロギング **API** 依存関係をデプロイメントに追加するかどうかを制御する **add-logging-api-dependencies** 属性があります。デフォルトではこの属性は **true** に設定されています。これは、暗黙的なロギング **API** 依存関係がすべてデプロイメントに追加されることを意味します。**false** に設定すると、暗黙的なロギング **API** 依存関係が追加されません。

add-logging-api-dependencies 属性は、管理 **CLI** を使用して設定できます。以下に例を示します。

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies, value=false)
```

バグの報告

12.1.7. 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用に作成されたログファイルです。デフォルト設定は **Periodic** ログハンドラーを使用してサーバーログファイルを書き込みます。*

*表**12.1** 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を含む、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。すべてのガベージコレクションの詳細が含まれます。

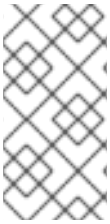
*表**12.2** 管理対象ドメイン用のデフォルトログファイル*

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。

バグの報告

12.1.8. ロギングのフィルター式

*フィルター式は、さまざまな基準に基づいてログメッセージを記録するために使用されます。フィルターチェックは、常に未フォーマットの **raw** メッセージに対して行われます。ロガーまたはハンドラーのフィルターを含めることができます。ロガーフィルターは、ハンドラーに配置されたフィルターよりも優先されます。*



注記

*ルートロガーに指定された **filter-spec** は他のロガーによって継承され **ません**。代わりに、ハンドラーごとに **filter-spec** を指定する必要があります。*

*表**12.3** ロギングのフィルター式*

フィルタータイプ **expression**	説明	パラメーター
Accept **accept**	すべてのログメッセージを許可します。	**accept**
却下 **deny**	すべてのログメッセージを拒否します。	**deny**
Not (否定) **not[filter expression]**	フィルター式の逆の値を返します。	1 つのフィルター式をパラメーターとして取ります。 **not(match("JBAS"))**
All **all[filter expression]**	複数のフィルター式より連結された値を返します。	コンマ区切りの複数のフィルター式を取ります。 **all(match("JBAS"),match("WELD"))**
すべて **any[filter expression]**	複数のフィルター式より 1 つの値を返します。	コンマ区切りの複数のフィルター式を取ります。 **any(match("JBAS"),match("WELD"))**
Level Change **levelChange[level]**	指定のレベルでログレコードを変更します。	1 つの文字列ベースのレベルを引数として取ります。 **levelChange("WARN")**
Levels **levels[levels]**	レベルリストにあるレベルの 1 つでログメッセージをフィルターします。	コンマで区切られた複数の文字列ベースのレベルを取ります。 **levels("DEBUG","INFO","WARN","ERROR")**

フィルタータイプ **expression**	説明	パラメーター
Level Range **levelRange[minLevel,maxLevel]**	指定されたレベル範囲内でログメッセージをフィルターします。	フィルター式は **[** を使用して含まれる最小レベルを示し、a **]** を使用して含まれる最大レベルを示します。または、それぞれ （ または ） を使用して除外を示すこともできます。式の最初の引数は許可される最小レベルで、2 つ目の引数は最大許容レベルです。 以下に例を示します。 ● **levelRange("DEBUG","ERROR")** 最小レベルは **DEBUG** よりも大きく、最大レベルは **ERROR** 未満でなければなりません。 ● **levelRange["DEBUG","ERROR")** 最小レベルは **DEBUG** 以上で、最大レベルは **ERROR** 未満でなければなりません。 ● **levelRange["INFO","ERROR"]** 最小レベルは **INFO** 以上で、最大レベルは **ERROR** 以下でなければなりません。
Match (**match["pattern"]**)	正規表現ベースのフィルター。式に指定されたパターンに対してフォーマットされていないメッセージが使用されます。	正規表現を引数として取ります。 **match("JBAS\d+")**
Substitute (**substitute["pattern","replacement value"]**)	最初にパターンと一致した値を指定の値に置き換えるフィルター。	式の最初の引数はパターンで、2 つ目の引数は置き換えるテキストです。 **substitute("JBAS","EAP")**
Replace All(**substituteAll["pattern","replacement value"]**)	パターンと一致したすべての値を指定の値に置き換えるフィルター。	式の最初の引数はパターンで、2 つ目の引数は置き換えるテキストです。 **substituteAll("JBAS","EAP")**

注記

値のコンマと引用符（'\' 演算子で前）をエスケープし、式全体を引用符で囲み、値が正しく 文字列 として解釈されるようにします。それ以外の場合は、リスト として解析されます。正しい形式の例は次のとおりです。

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=CONSOLE:write-attribute(name=filter-spec, value="substituteAll(\"JBAS\",\"SABJ\")")
```

バグの報告

12.1.9. ログレベル

ログレベルは、ログメッセージの性質と重大度を示す列挙値の順序付けされたセットです。特定のログメッセージのレベルは、開発者がメッセージを送信するために選択したロギングフレームワークの適切なメソッドを使用して指定します。

JBoss EAP 6 は、サポートされるアプリケーションロギングフレームワークによって使用されるすべてのログレベルをサポートします。最も一般的に使用される **6** つのログレベルは（最低から高い順に） **TRACE**、**DEBUG**、**INFO**、**WARN**、**ERROR**、および **FATAL** です。

ログレベルはログカテゴリーとログハンドラーによって使用され、それらが担当するメッセージを限定します。各ログレベルには、他のログレベルに対して相対的な順番を示す数値が割り当てられています。ログカテゴリーとハンドラーにはログレベルが割り当てられ、そのレベル以上のログメッセージのみを処理します。たとえば、**WARN** レベルのログハンドラーは、 **WARN** 、**ERROR**、および **FATAL** レベルのメッセージのみを記録します。

バグの報告

12.1.10. サポート対象のログレベル

表**12.4** サポート対象のログレベル

ログのレベル	Value	説明
FINEST	300	-

ログのレベル	Value	説明
FINER	400	-
TRACE	400	アプリケーションの実行状態に関する詳細情報を提供するメッセージに使用します。通常、**TRACE** のログメッセージはアプリケーションのデバッグ時にのみキャプチャーされます。
DEBUG	500	アプリケーションの個別の要求またはアクティビティーの進捗状況を示すメッセージに使用します。**DEBUG** のログメッセージは通常、アプリケーションをデバッグする場合にのみキャプチャーされます。
FINE	500	-
CONFIG	700	-
INFO	800	アプリケーションの全体的な進捗状況を示すメッセージに使用します。多くの場合、アプリケーションの起動、シャットダウン、およびその他の主要なライフサイクルイベントに使用されます。
WARN	900	エラーではないが、理想的とは見なされない状況を示すために使用されます。今後エラーが発生する可能性のある状況を示す場合があります。
警告	900	-
ERROR	1000	発生したエラーの中で、現在の活動や要求の完了を妨げる可能性があるが、アプリケーション実行の妨げにはならないエラーを表示するために使用されます。
SEVERE	1000	-
FATAL	1100	クリティカルなサービス障害やアプリケーションのシャットダウンをもたらしたり、JBoss EAP 6 のシャットダウンを引き起こす可能性があるイベントを表示するのに使用されます。

バグの報告

12.1.11. ログカテゴリー

*ログカテゴリーは、キャプチャーするログメッセージのセットと、メッセージを処理する **1** つまたは複数のログハンドラーを定義します。*

*キャプチャーするログメッセージは、元の **Java** パッケージとログレベルによって定義されます。そのパッケージのクラスおよびそのログレベル以下のメッセージがログカテゴリーによってキャプチャーされ、指定のログハンドラーに送信されます。*

ログカテゴリーは、独自のハンドラーの代わりにルートロガーのログハンドラーを任意で使用することができます。

バグの報告

12.1.12. ルートロガーについて

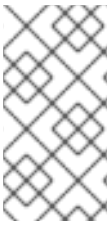
ルートロガーは、ログカテゴリーによってキャプチャーされないサーバー（指定レベルの）に送信されたすべてのログメッセージをキャプチャーします。これらのメッセージは **1** つ以上のログハンドラーに送信されます。

デフォルトでは、ルートロガーはコンソールおよび周期ログハンドラーを使用するように設定されています。**Periodic** ログハンドラーは、ファイル **server.log** に書き込むように設定されます。このファイルはサーバーログと呼ばれることもあります。

バグの報告

12.1.13. ログハンドラー

ログハンドラーはキャプチャーしたメッセージの記録方法を定義します。利用できるログハンドラーは、コンソール、ファイル、周期、サイズ、**Async**、**syslog**、**Periodic Size**、および **Custom** です。



注記

ログハンドラーをアクティブにするには、少なくとも **1** つのロガーに追加する必要があります。

バグの報告

12.1.14. ログハンドラーのタイプ

Console

コンソールログハンドラーは、ログメッセージをホストオペレーティングシステムの標準出力（標準出力）または標準エラー**(stderr)**ストリームに書き込みます。これらのメッセージは、**JBoss EAP 6** がコマンドラインプロンプトから実行された場合に表示されます。オペレーティングシステ

ムが標準出力または標準エラーストリームをキャプチャーするように設定されていない限り、**Console** ログハンドラーからのメッセージは保存されません。

ファイル

File ログハンドラーはログメッセージを指定のファイルに書き込みます。

Periodic

Periodic ログハンドラーは、指定した時間が経過するまで、ログメッセージを指定ファイルに書き込みます。期間が経過したら、指定されたタイムスタンプを追加してファイルの名前が変更され、ハンドラーは元の名前で新規作成されたログファイルに書き込みを継続します。

サイズ

Size ログハンドラーは、指定したファイルが指定したサイズに達するまで、ログメッセージを名前付きファイルに書き込みます。ファイルが指定したサイズに達すると、数値の接尾辞が付いて名前が変更され、ハンドラーは元の名前で新規作成されたログファイルに書き込みを継続します。各サイズログハンドラーは、この方式で保管されるファイルの最大数を指定する必要があります。

Periodic Size

JBoss EAP 6.4 から利用できます。これは **Periodic** および **Size** ハンドラーの組み合わせで、組み合わされた属性をサポートします。

現在のログファイルが指定されたサイズに達する **か**、指定の期間が経過すると、ファイルの名前が変更され、ハンドラーは元の名前で新規作成されたログファイルに書き込みを継続します。

非同期

非同期ログハンドラーは、**1** つ以上のログハンドラーに対して非同期動作を提供するラッパーログハンドラーです。これは、待ち時間が長い可能性があるログハンドラーや、ネットワークファイルシステムへのログファイルの書き込みなど、パフォーマンス上の問題がある場合に役立ちます。

カスタム

Custom ログハンドラーを使用すると、実装された新しいタイプのログハンドラーを設定することができます。カスタムハンドラーは、**java.util.logging.Handler** を拡張し、モジュールに含まれる **Java** クラスとして実装する必要があります。

syslog

syslog ハンドラーは、リモートのロギングサーバーへメッセージを送信するために使用できます。これにより、複数のアプリケーションが同じサーバーにログメッセージを送信でき、そのサーバーですべてのログメッセージを解析できます。

バグの報告

12.1.15. ログフォーマッター

*ログフォーマッターは、ログハンドラーの設定プロパティーで、そのハンドラーからのログメッセージの表示を定義します。これは、**java.util.Formatter** クラスに基づいた構文を使用する文字列です。*

*たとえば、デフォルト設定のログフォーマッター文字列 **%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n** は、以下のようなログメッセージを作成します。*

> *15:53:26,546 INFO [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console listening on http://127.0.0.1:9990*

バグの報告

12.1.16. ログフォーマッター構文

表12.5 ログフォーマッター構文

記号	説明
%c	ロギングイベントのカテゴリー
%p	ログエントリーのレベル (情報やデバッグなど)
%P	ログエントリーのローカライズレベル
%d	現在の日付/時刻 (yyyy-MM-dd HH:mm:ss,SSS 形式)
%r	相対時間 (ログが初期化された以降のミリ秒単位の時間)
%z	タイムゾーン
%k	ログリソースキー (ログメッセージのローカリゼーションに使用)
%m	ログメッセージ (例外トレースを除外)

記号	説明
%s	単純なログメッセージ (例外トレースなし)
%e	例外スタックトレース (拡張モジュール情報なし)
%E	例外スタックトレース (拡張モジュール情報あり)
%t	現在のスレッドの名前
%n	改行文字
%C	ログメソッドを呼び出すコードのクラス (低速)
%F	ログメソッドを呼び出すクラスのファイル名 (低速)
%l	ログメソッドを呼び出すコードのソースロケーション (低速)
%L	ログメソッドを呼び出すコードの行番号 (低速)
%M	ログメソッドを呼び出すコードのメソッド (低速)
%x	ネスト化診断コンテキスト
%X	メッセージ診断コンテキスト
%%	リテラルパーセント記号 (エスケープ)

バグの報告

12.2. 管理コンソールでのロギングの設定

管理コンソールは、ルートロガー、ログハンドラー、およびログカテゴリーの設定用のグラフィカルユーザーインターフェースを提供します。管理コンソールの詳細は、「管理コンソール」 を参照してください。

ロギング設定にアクセスするには、以下の手順に従います。

手順12.3 アクセスロギングの設定

1. *管理コンソールへのログイン*

2. ***logging*** *サブシステム設定に移動します。この手順は、スタンドアロンサーバーとして実行されているサーバーと管理対象ドメインで実行されているサーバーによって異なります。*

   - *スタンドアロンサーバー*

     Configuration *をクリックし、****Subsystems*** *メニューの****Core*** *を展開し、****Logging*** *をクリックします。*

   - *管理対象ドメイン*

     Configuration *をクリックし、ドロップダウンメニューから編集するプロファイルを選択します。****Subsystems*** *メニューで****Core*** *を展開し、****Logging*** *をクリックします。*

ルートロガーを設定するために実行できるタスクは以下のとおりです。

- *ログレベルを編集します。*
- *ログハンドラーを追加および削除します。*

ログカテゴリーを設定するために実行できるタスクは以下のとおりです。

- *ログカテゴリーを追加および削除します。*
- *ログカテゴリープロパティーを編集します。*
- *カテゴリーのログハンドラーを追加および削除します。*

ログハンドラーを設定するために実行できる主なタスクは以下のとおりです。

- 新しいハンドラーを追加します。
- ハンドラーを設定します。

管理コンソールでは、すべてのサポート対象ログハンドラー (カスタムを含む) を設定できます。

バグの報告

12.3. CLI でのロギング設定

前提条件

管理 **CLI** が実行され、関連する **JBoss EAP** インスタンスに接続されている必要があります。詳細はを参照してください。 「管理 **CLI** の起動」

バグの報告

12.3.1. CLI でのルートロガーの設定

ルートロガーの設定は管理 **CLI** を使用して確認および編集することができます。

ルートロガーを設定するために実行する主なタスクは以下のとおりです。

- ルートロガーへのログハンドラーの追加
- ルートロガー設定の表示
- ログレベルの変更

- *ルートロガーからのログハンドラーの削除*

重要

*ルートロガーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。*

ルートロガーへのログハンドラーの追加

*次の構文で **add-handler** 操作を使用します。**HANDLER** は追加するログハンドラーの名前です。*

```
/subsystem=logging/root-logger=ROOT:add-handler(name="HANDLER")
```

ログハンドラーを作成してから、ログハンドラーをルートロガーへ追加する必要があります。

*例**12.3** ルートロガーの **add-handler** 操作*

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:add-
handler(name="FILE")
{"outcome" => "success"}
```

ルートロガーの設定内容の表示

*以下の構文で **read-resource** 操作を使用します。*

```
/subsystem=logging/root-logger=ROOT:read-resource
```

*例**12.4** ルートロガーの **read-resource** 操作*

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:read-resource
{
    "outcome" => "success",
    "result" => {
        "filter" => undefined,
        "filter-spec" => undefined,
```

```
        "handlers" => [
            "CONSOLE",
            "FILE"
        ],
        "level" => "INFO"
    }
}
```

ルートロガーのログレベルの設定

次の構文で ***write-attribute*** *操作を使用します。* ***LEVEL*** *はサポートされているログレベルの* ***1*** *つになります。*

```
/subsystem=logging/root-logger=ROOT:write-attribute(name="level", value="LEVEL")
```

例 ***12.5*** *ルートロガーの* ***write-attribute*** *操作によるログレベルの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

ルートロガーからのログハンドラーの削除

次の構文で ***remove-handler*** *を使用します。* ***HANDLER*** *は削除するログハンドラーの名前です。*

```
/subsystem=logging/root-logger=ROOT:remove-handler(name="HANDLER")
```

例 ***12.6*** *ログハンドラーの削除*

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:remove-
handler(name="FILE")
{"outcome" => "success"}
```

バグの報告

12.3.2. CLI でのログカテゴリー設定

ログカテゴリーは ***CLI*** *で追加、削除、および編集できます。*

ログカテゴリーを設定するために実行する主なタスクは次のとおりです。

- 新しいログカテゴリーの追加
- ログカテゴリーの設定表示
- ログレベルを設定します。
- ログカテゴリーへのログハンドラーの追加
- ログカテゴリーからのログハンドラーの削除
- ログカテゴリーの削除

重要

ログカテゴリーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは */**subsystem=logging**/* ではなく */**subsystem=logging**/**logging-profile=NAME**/* になります。

管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。*/**subsystem=logging**/* は */**profile=NAME**/**subsystem=logging**/* に置き換えます。

ログカテゴリーの追加

以下の構文で **add** 操作を使用します。**CATEGORY** を、追加するカテゴリーに置き換えます。

```
/subsystem=logging/logger=CATEGORY:add
```

例**12.7** 新規カテゴリーの追加

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:add
{"outcome" => "success"}
```

ログカテゴリーの設定の表示

*以下の構文で **read-resource** 操作を使用します。**CATEGORY** をカテゴリーの名前に置き換えます。*

```
/subsystem=logging/logger=CATEGORY:read-resource
```

*例**12.8** ログカテゴリーの **read-resource** 操作*

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=org.apache.tomcat.util.modeler:read-resource
{
    "outcome" => "success",
    "result" => {
        "category" => "org.apache.tomcat.util.modeler",
        "filter" => undefined,
        "filter-spec" => undefined,
        "handlers" => undefined,
        "level" => "WARN",
        "use-parent-handlers" => true
    }
}
```

ログレベルの設定

*次の構文で **write-attribute** 操作を使用します。**CATEGORY** はログカテゴリーの名前に、**LEVEL** は設定するログレベルに置き換えます。*

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="level", value="LEVEL")
```

*例**12.9** ログレベルの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:write-
attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

ルートロガーのログハンドラーを使用するためのログカテゴリの設定

*次の構文で **write-attribute** 操作を使用します。**CATEGORY** はログカテゴリーの名前に置き換えます。ルートロガーのハンドラーを使用するために、このログカテゴリーの **BOOLEAN** を **true** に置き換えます。独自の割り当てられたハンドラーのみを使用する場合は **false** に置き換えます。*

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="use-parent-handlers",
value="BOOLEAN")
```

*例**12.10 use-parent-handlers** の設定*

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:write-attribute(name="use-parent-handlers", value="true")
{"outcome" => "success"}
```

ログカテゴリへのログハンドラ追加

*次の構文で **add-handler** 操作を使用します。**CATEGORY** をカテゴリーの名前に置き換え、**HANDLER** を追加するハンドラーの名前に置き換えます。*

```
/subsystem=logging/logger=CATEGORY:add-handler(name="HANDLER")
```

ログハンドラーを作成してから、ログハンドラーをルートロガーへ追加する必要があります。

*例**12.11** ログハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:add-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
```

ログカテゴリからのログハンドラの削除

*次の構文で **remove-handler** 操作を使用します。**CATEGORY** をカテゴリーの名前に置き換え、**HANDLER** を削除するログハンドラーの名前に置き換えます。*

```
/subsystem=logging/logger=CATEGORY:remove-handler(name="HANDLER")
```

*例**12.12** ログハンドラーの削除*

```
[standalone@localhost:9999 /] /subsystem=logging/logger=jacorb:remove-handler(name="AccountsNFSAsync")
{"outcome" => "success"}
```

カテゴリーの削除

*以下の構文で **remove** 操作を使用します。**CATEGORY** は削除するカテゴリーの名前に置き換えます。*

```
/subsystem=logging/logger=CATEGORY:remove
```

*例**12.13** ログカテゴリの削除*

```
[standalone@localhost:9999 /]
/subsystem=logging/logger=com.company.accounts.rec:remove
{"outcome" => "success"}
```

バグの報告

12.3.3. CLI でのコンソールログハンドラーの設定

コンソールログハンドラーは **CLI** で追加、削除、および編集できます。

Console ログハンドラーを設定するために実行する主なタスクは次のとおりです。

- 新しいコンソールログハンドラーの追加
- コンソールログハンドラーの設定表示
- ハンドラーのログレベルの設定
- ハンドラーの出力のターゲットの設定
- ハンドラーの出力に使用されるエンコーディングの設定
- ハンドラーの出力に使用されるフォーマッターの設定
- ハンドラーによる自動フラッシュ使用の有無を設定
- コンソールログハンドラーの削除

重要

*ログハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。*

***Console** ログハンドラーの追加*

*以下の構文で **add** 操作を使用します。**HANDLER** を、追加するコンソールログハンドラーに置き換えます。*

```
/subsystem=logging/console-handler=HANDLER:add
```

*例**12.14 Console** ログハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:add
{"outcome" => "success"}
```

コンソールログハンドラーの設定表示

*以下の構文で **read-resource** 操作を使用します。**HANDLER** をコンソールログハンドラーの名前に置き換えます。*

```
/subsystem=logging/console-handler=HANDLER:read-resource
```

*例**12.15** コンソールログハンドラーの設定表示*

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=CONSOLE:read-resource
{
    "outcome" => "success",
    "result" => {
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "INFO",
        "name" => "CONSOLE",
        "named-formatter" => "COLOR-PATTERN",
```

```
        "target" => "System.out"
    }
}
```

ログレベルの設定

次の構文で **write-attribute** 操作を使用します。**HANDLER** はコンソールログハンドラーの名前に、**LEVEL** は設定するログレベルに置き換えます。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="level",
value="INFO")
```

例**12.16** ログレベルの設定

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

ターゲットの設定

次の構文で **write-attribute** 操作を使用します。**HANDLER** をコンソールログハンドラーの名前に置き換えます。**TARGET** を、システムエラーストリームまたは標準出力ストリームの **System.err** または **System.out** に置き換えます。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="target",
value="TARGET")
```

例**12.17** ターゲットの設定

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-
attribute(name="target", value="System.err")
{"outcome" => "success"}
```

エンコーディングの設定

次の構文で **write-attribute** 操作を使用します。**HANDLER** をコンソールログハンドラーの名前に置き換えます。**ENCODING** を、必要な文字エンコーディングシステムの名前に置き換えます。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

例**12.18** エンコーディングの設定

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

フォーマッターの設定

*次の構文で **write-attribute** 操作を使用します。**HANDLER** をコンソールログハンドラーの名前に置き換えます。**FORMAT** は必要なフォーマッター文字列に置き換えます。*

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

*例**12.19** フォーマッターの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

自動フラッシュの設定

*次の構文で **write-attribute** 操作を使用します。**HANDLER** をコンソールログハンドラーの名前に置き換えます。このハンドラーが出力をすぐに書き込む場合は、**BOOLEAN** を **true** に置き換えます。*

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="autoflush", value="BOOLEAN")
```

*例**12.20** 自動フラッシュの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

***Console** ログハンドラーの削除*

*以下の構文で **remove** 操作を使用します。**HANDLER** を、削除するコンソールログハンドラーの名前に置き換えます。*

```
/subsystem=logging/console-handler=HANDLER:remove
```

*例**12.21 Console** ログハンドラーの削除*

```
[standalone@localhost:9999 /] /subsystem=logging/console-
handler=ERRORCONSOLE:remove
{"outcome" => "success"}
```

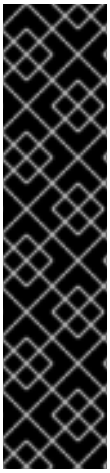
バグの報告

12.3.4. CLI でのファイルログハンドラーの設定

ファイルログハンドラーは **CLI** で追加、削除、および編集できます。

File ログハンドラーを設定するために実行する主なタスクは以下のとおりです。

- 新しいファイルログハンドラーの追加
- ファイルログハンドラーの設定表示
- ハンドラーのログレベルの設定
- ハンドラーの追加動作の設定
- ハンドラーによる自動フラッシュ使用の有無を設定
- ハンドラーの出力に使用されるエンコーディングの設定
- ログハンドラーが書き込むファイルの指定
- ハンドラーの出力に使用されるフォーマッターの設定
- ファイルログハンドラーの削除



重要

*ログハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。*

ファイルログハンドラーの追加

*以下の構文で **add** 操作を使用します。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。*

```
/subsystem=logging/file-handler=HANDLER:add(file={"path"=>"PATH", "relative-
to"=>"DIR"})
```

*例**12.22** ファイルログハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:add(file=
{"path"=>"accounts.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

ファイルログハンドラー設定の表示

*以下の構文で **read-resource** 操作を使用します。**HANDLER** をファイルログハンドラーの名前に置き換えます。*

```
/subsystem=logging/file-handler=HANDLER:read-resource
```

*例**12.23 read-resource** 操作の使用*

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "accounts.log",
            "relative-to" => "jboss.server.log.dir"
```

```
        },
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "ALL",
        "name" => "accounts_log",
        "named-formatter" => undefined
    }
}
```

ログレベルの設定

*次の構文で **write-attribute** 操作を使用します。**HANDLER** をファイルログハンドラーの名前に置き換えます。**LOG_LEVEL_VALUE** を設定するログレベルに置き換えます。*

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

*例**12.24** ログレベルの変更*

```
/subsystem=logging/file-handler=accounts_log:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

追加動作の設定

*次の構文で **write-attribute** 操作を使用します。**HANDLER** をファイルログハンドラーの名前に置き換えます。アプリケーションサーバーが起動されるたびに新しいログファイルを作成する必要がある場合は、**BOOLEAN** を **false** に置き換えます。アプリケーションサーバーが同じファイルを使用し続ける必要がある場合は、**BOOLEAN** を **true** に置き換えます。*

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")
```

*例**12.25** 追加プロパティーの変更*

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

この変更を反映するには、**JBoss EAP 6** を再起動する必要があります。

自動フラッシュの設定

次の構文で **write-attribute** 操作を使用します。**HANDLER** をファイルログハンドラーの名前に置き換えます。このハンドラーが出力をすぐに書き込む場合は、**BOOLEAN** を **true** に置き換えます。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

例**12.26** 自動フラッシュプロパティーの変更

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
attribute(name="autoflush", value="false")
{"outcome" => "success"}
```

エンコーディングの設定

次の構文で **write-attribute** 操作を使用します。**HANDLER** をファイルログハンドラーの名前に置き換えます。**ENCODING** を、必要な文字エンコーディングシステムの名前に置き換えます。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="encoding",
value="ENCODING")
```

例**12.27** エンコーディングの設定

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

ログハンドラーが書き込むファイルの変更

次の構文で **write-attribute** 操作を使用します。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="file", value=
{"path"=>"PATH", "relative-to"=>"DIR"})
```

例**12.28** ログハンドラーが書き込むファイルの変更

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
```

```
attribute(name="file", value={"path"=>"accounts-debug.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

フォーマッターの設定

次の構文で ***write-attribute*** *操作を使用します。* ***HANDLER*** *をファイルログハンドラーの名前に置き換えます。* ***FORMAT*** *は必要なフォーマッター文字列に置き換えます。*

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

例 ***12.29*** *フォーマッターの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

File *ログハンドラーの削除*

以下の構文で ***remove*** *操作を使用します。* ***HANDLER*** *を、削除するファイルログハンドラーの名前に置き換えます。*

```
/subsystem=logging/file-handler=HANDLER:remove
```

例 ***12.30 File*** *ログハンドラーの削除*

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:remove
{"outcome" => "success"}
```

ログハンドラーは、ログカテゴリーまたは非同期ログハンドラーによって参照されていない場合にのみ削除できます。

バグの報告

12.3.5. CLI での周期ログハンドラーの設定

周期ログハンドラーは、 ***CLI*** *で追加、削除、および編集できます。*

***Periodic** ログハンドラーを設定するために実行する主なタスクは以下のとおりです。*

- *新しい周期ログハンドラーの追加*
- *周期ログハンドラーの設定表示*
- *ハンドラーのログレベルの設定*
- *ハンドラーの追加動作の設定*
- *ハンドラーによる自動フラッシュ使用の有無を設定し ます。*
- *ハンドラーの出力に使用されるエンコーディングの設定*
- *ログハンドラーが書き込むファイルの指定*
- *ハンドラーの出力に使用されるフォーマッターの設定*
- *ローテーションされるログの接尾辞を設定します。*
- *周期ログハンドラーの削除*

これらの各タスクについては以下で説明します。

重要

*ログハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。*

新しい周期ローテーションファイルログハンドラーの追加

*以下の構文で **add** 操作を使用します。*

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:add(file={"path"=>"PATH",
"relative-to"=>"DIR"}, suffix="SUFFIX")
```

***HANDLER** をログハンドラーの名前に置き換えます。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。**SUFFIX** を、使用されるファイルローテーション接尾辞に置き換えます。*

*例**12.31** 新しいハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:add(file={"path"=>"daily-debug.log", "relative-
to"=>"jboss.server.log.dir"}, suffix=".yyyy.MM.dd")
{"outcome" => "success"}
```

周期ローテーションファイルログハンドラー設定の表示

*以下の構文で **read-resource** 操作を使用します。*

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:read-resource
```

***HANDLER** をログハンドラーの名前に置き換えます。*

*例**12.32 read-resource** 操作の使用*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:read-resource
```

```
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "daily-debug.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "ALL",
        "name" => "HOURLY_DEBUG",
        "named-formatter" => undefined,
        "suffix" => ".yyyy.MM.dd"
    },
    "response-headers" => {"process-state" => "reload-required"}
}
```

ログレベルの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="level". value="LOG_LEVEL_VALUE")
```

HANDLER *を周期ログハンドラーの名前に置き換えます。* ***LOG_LEVEL_VALUE*** *を設定するログレベルに置き換えます。*

例 ***12.33*** *ログレベルの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
```

追加動作の設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/periodic-rotating-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")
```

HANDLER を周期ログハンドラーの名前に置き換えます。アプリケーションサーバーが起動されるたびに新しいログファイルを作成する必要がある場合は、**BOOLEAN** を **false** に置き換えます。アプリケーションサーバーが同じファイルを使用し続ける必要がある場合は、**BOOLEAN** を **true** に置き換えます。

この変更を反映するには、**JBoss EAP 6** を再起動する必要があります。

例**12.34** 追加動作の設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

自動フラッシュの設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

HANDLER を周期ログハンドラーの名前に置き換えます。このハンドラーが出力をすぐに書き込む場合は、**BOOLEAN** を **true** に置き換えます。

例**12.35** 自動フラッシュ動作の設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="autoflush", value="false")
{
    "outcome" => "success",
    "response-headers" => {"process-state" => "reload-required"}
}
```

エンコーディングの設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="encoding", value="ENCODING")
```

HANDLER を周期ログハンドラーの名前に置き換えます。**ENCODING** を、必要な文字エンコーディングシステムの名前に置き換えます。

例**12.36** エンコーディングの設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

ログハンドラーが書き込むファイルの変更

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

HANDLER を周期ログハンドラーの名前に置き換えます。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。

例**12.37** ログハンドラーが書き込むファイルの変更

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="file", value={"path"=>"daily-debug.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

フォーマッターの設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

HANDLER を周期ログハンドラーの名前に置き換えます。**FORMAT** は必要なフォーマッター文字列に置き換えます。

例**12.38** フォーマッターの設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n")
{"outcome" => "success"}
```

ローテーションされるログの接尾辞の設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:write-attribute(name="suffix", value="SUFFIX")
```

HANDLER *をログハンドラーの名前に置き換えます。* ***SUFFIX*** *を、必要な接尾辞の文字列に置き換えます。*

例 ***12.39*** *ローテーションされるログの接尾辞の設定*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")
{"outcome" => "success"}
```

周期ログハンドラーの削除

以下の構文で ***remove*** *操作を使用します。*

```
/subsystem=logging/periodic-rotating-file-handler=HANDLER:remove
```

HANDLER *を周期ログハンドラーの名前に置き換えます。*

例 ***12.40*** *周期ログハンドラーの削除*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-handler=HOURLY_DEBUG:remove
{"outcome" => "success"}
```

バグの報告

12.3.6. CLI でのサイズログハンドラーの設定

*サイズローテーションファイルログハンドラーは、**CLI** で追加、削除、および編集できます。*

サイズローテーションファイルログハンドラーを設定するために実行するタスクは以下のとおりです。

- *新しいログハンドラーの追加*
- *ログハンドラーの設定表示*
- *ハンドラーのログレベルの設定*
- *ハンドラーの追加動作の設定*
- *ハンドラーによる自動フラッシュ使用の有無を設定*
- *ハンドラーの出力に使用されるエンコーディングの設定*
- *ログハンドラーが書き込むファイルの指定*
- *ハンドラーの出力に使用されるフォーマッターの設定*
- *各ログファイルの最大サイズを設定します。*
- *保持するバックアップログの最大数の設定*
- *サイズローテーションファイルハンドラーにブート時のローテーションオプションを設定します。*
- *ローテーションされるログの接尾辞を設定します。*

- *ログハンドラーの削除*

これらの各タスクについては以下で説明されています。

重要

*ログハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。*

新しいログハンドラーの追加

*以下の構文で **add** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:add(file={"path"=>"PATH",
"relative-to"=>"DIR"})
```

***HANDLER** をログハンドラーの名前に置き換えます。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。*

*例**12.41** 新しいログハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:add(file={"path"=>"accounts_trace.log", "relative-
to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

ログハンドラーの設定表示

*以下の構文で **read-resource** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:read-resource
```

HANDLER *をログハンドラーの名前に置き換えます。*

*例**12.42** ログハンドラーの設定表示*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "file" => {
            "path" => "accounts_trace.log",
            "relative-to" => "jboss.server.log.dir"
        },
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "ALL",
        "max-backup-index" => 1,
        "name" => "ACCOUNTS_TRACE",
        "named-formatter" => undefined,
        "rotate-on-boot" => false,
        "rotate-size" => "2m",
        "suffix" => undefined
    }
}
```

ハンドラーのログレベルの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attributel(name="level",
value="LOG_LEVEL_VALUE")
```

HANDLER *をログハンドラーの名前に置き換えます。****LOG_LEVEL_VALUE*** *を設定するログレベルに置き換えます。*

*例**12.43** ハンドラーのログレベルの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

ハンドラーの追加動作の設定

*次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="append",
value="BOOLEAN")
```

***HANDLER** をログハンドラーの名前に置き換えます。アプリケーションサーバーが起動されるたびに新しいログファイルを作成する必要がある場合は、**BOOLEAN** を **false** に置き換えます。アプリケーションサーバーが同じファイルを使用し続ける必要がある場合は、**BOOLEAN** を **true** に置き換えます。*

*この変更を反映するには、**JBoss EAP 6** を再起動する必要があります。*

*例**12.44** ハンドラーの追加動作の設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

ハンドラーによる自動フラッシュ使用の有無を設定

*次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="autoflush", value="BOOLEAN")
```

***HANDLER** をログハンドラーの名前に置き換えます。このハンドラーが出力をすぐに書き込む場合は、**BOOLEAN** を **true** に置き換えます。*

*例**12.45** ハンドラーによる自動フラッシュ使用の有無を設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

ハンドラーの出力に使用されるエンコーディングの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="encoding", value="ENCODING")
```

HANDLER *をログハンドラーの名前に置き換えます。* ***ENCODING*** *を、必要な文字エンコーディングシステムの名前に置き換えます。*

例 ***12.46*** *ハンドラーの出力に使用されるエンコーディングの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

ログハンドラーが書き込むファイルの指定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="file",
value={"path"=>"PATH", "relative-to"=>"DIR"})
```

HANDLER *をログハンドラーの名前に置き換えます。* ***PATH*** *を、ログが書き込まれるファイルのファイル名に置き換えます。* ***DIR*** *は、ファイルが存在するディレクトリー名に置き換えます。* ***DIR*** *の値はパス変数になります。*

例 ***12.47*** *ログハンドラーが書き込むファイルの指定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:write-attribute(name="file", value=
{"path"=>"accounts_trace.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

ハンドラーの出力に使用されるフォーマッターの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-
attribute(name="formatter", value="FORMATTER")
```

HANDLER *をログハンドラーの名前に置き換えます。****FORMAT*** *は必要なフォーマッター文字列に置き換えます。*

*例****12.48*** *ハンドラーの出力に使用されるフォーマッターの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p (%c) [%t] %s%E%n")
{"outcome" => "success"}
```

各ログファイルの最大サイズの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-size", value="SIZE")
```

HANDLER *をログハンドラーの名前に置き換えます。****SIZE*** *はファイルの最大サイズに置き換えます。*

*例****12.49*** *各ログファイルの最大サイズの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="rotate-size", value="50m")
{"outcome" => "success"}
```

保持するバックアップログの最大数の設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="max-backup-index", value="NUMBER")
```

HANDLER *をログハンドラーの名前に置き換えます。****NUMBER*** *を、保持するログファイルの数に置き換えます。*

*例****12.50*** *保持するバックアップログの最大数の設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="max-backup-index", value="5")
{"outcome" => "success"}
```

***size-rotating-file-handler*での *rotate-on-boot* オプションの設定**

*このオプションは、**size-rotating-file-handler** ファイルハンドラーでのみ利用できます。デフォルトは **false** で、サーバーの再起動時に新しいログファイルは作成されません。*

*これを変更するには、次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="rotate-on-boot", value="BOOLEAN")
```

***HANDLER** を **size-rotating-file-handler** ログハンドラーの名前に置き換えます。再起動時に新しい **size-rotating-file-handler** ログファイルを作成する場合は、**BOOLEAN** を **true** に置き換えます。*

*例**12.51** サーバーの再起動時に新しい **size-rotating-file-handler** ログファイルを作成するよう指定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="rotate-on-boot", value="true")
{"outcome" => "success"}
```

ローテーションされるログの接尾辞の設定

*次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/size-rotating-file-handler=HANDLER:write-attribute(name="suffix", value="SUFFIX")
```

***HANDLER** をログハンドラーの名前に置き換えます。**SUFFIX** を、必要な接尾辞の文字列に置き換えます。*

*例**12.52** ローテーションされるログの接尾辞の設定*

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-handler=ACCOUNTS_TRACE:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")
{"outcome" => "success"}
```

ログハンドラーの削除

*以下の構文で **remove** 操作を使用します。*

-

> ***/subsystem=logging/size-rotating-file-handler=HANDLER:remove***
>
> ***HANDLER*** *をログハンドラーの名前に置き換えます。*

> *例**12.53** ログハンドラーの削除*
>
> ```
> [standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
> handler=ACCOUNTS_TRACE:remove
> {"outcome" => "success"}
> ```

バグの報告

*12.3.7. CLI での **Periodic Size rotating** ログハンドラーの設定*

Periodic Size Rotating *ログハンドラーは **CLI** で追加、削除、および編集できます。*

Periodic Size rotating *ハンドラーを設定するために実行する主なタスクは以下のとおりです。*

- *新しい **Periodic Size rotating** ハンドラーを追加します。*
- ***Periodic Size rotating** ハンドラーの設定を表示します。*
- *ハンドラーのログレベルの設定*
- *ハンドラーの追加動作の設定*
- *ハンドラーによる自動フラッシュ使用の有無を設定し ます。*
- *ハンドラーの出力に使用されるエンコーディングの設定*
- *ログハンドラーが書き込むファイルの指定*

- ハンドラーの出力に使用されるフォーマッターの設定
- 各ログファイルの最大サイズを設定します。
- 保持するバックアップログの最大数の設定
- **Periodic Size rotating** ハンドラーのブート時のローテーションオプションを設定します。
- ローテーションされるログの接尾辞を設定します。
- **Periodic Size rotating** ハンドラーの削除。

これらの各タスクについては以下で説明します。

重要

スタンドアロンシステムのロギングプロファイルで **Periodic Size rotating** ハンドラーを設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。

管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。

新しい **Periodic Size** ローテーションハンドラーの追加

以下の構文で **add** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:add(file=
{"path"=>"PATH", "relative-to"=>"DIR"})
```

HANDLER をログハンドラーの名前に置き換えます。**PATH** を、ログが書き込まれるファイルのファイル名に置き換えます。**DIR** は、ファイルが存在するディレクトリー名に置き換えます。**DIR** の値はパス変数になります。

例**12.54** 新しいログハンドラーの追加

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:add(file={"path"=>"periodic_size.log","relative-
to"=>"jboss.server.log.dir"},suffix=".yyyy.MM.dd")
{"outcome" => "success"}
```

ログハンドラーの設定表示

以下の構文で **read-resource** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:read-resource
```

HANDLER をログハンドラーの名前に置き換えます。

例**12.55** ログハンドラーの設定表示

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:read-resource
{
    "outcome" => "success",
    "result" => {
        "append" => true,
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "file" => {
            "relative-to" => "jboss.server.log.dir",
            "path" => "periodic_size.log"
        },
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "ALL",
        "max-backup-index" => 1,
        "name" => "PERIODIC_SIZE",
        "named-formatter" => undefined,
        "rotate-on-boot" => false,
        "rotate-size" => "2m",
        "suffix" => ".yyyy.MM.dd"
    }
}
```

ハンドラーのログレベルの設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="level", value="LOG_LEVEL_VALUE")
```

HANDLER をログハンドラーの名前に置き換えます。*LOG_LEVEL_VALUE* を設定するログレベルに置き換えます。

例*12.56* ハンドラーのログレベルの設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

ハンドラーの追加動作の設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="append", value="BOOLEAN")
```

HANDLER をログハンドラーの名前に置き換えます。アプリケーションサーバーが起動されるたびに新しいログファイルを作成する必要がある場合は、*BOOLEAN* を *false* に置き換えます。アプリケーションサーバーが同じファイルを使用し続ける必要がある場合は、*BOOLEAN* を *true* に置き換えます。

この変更を反映するには、*JBoss EAP 6* を再起動する必要があります。

例*12.57* ハンドラーの追加動作の設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="append", value="true")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

ハンドラーによる自動フラッシュ使用の有無を設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="autoflush", value="BOOLEAN")
```

HANDLER をログハンドラーの名前に置き換えます。このハンドラーが出力をすぐに書き込む場合は、*BOOLEAN* を *true* に置き換えます。

例**12.58** ハンドラーによる自動フラッシュ使用の有無を設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="autoflush", value="true")
{"outcome" => "success"}
```

ハンドラーの出力に使用されるエンコーディングの設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="encoding", value="ENCODING")
```

HANDLER をログハンドラーの名前に置き換えます。*ENCODING* を、必要な文字エンコーディングシステムの名前に置き換えます。

例**12.59** ハンドラーの出力に使用されるエンコーディングの設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="encoding", value="utf-8")
{"outcome" => "success"}
```

ログハンドラーが書き込むファイルの指定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH", "relative-to"=>"DIR"})
```

HANDLER をログハンドラーの名前に置き換えます。*PATH* を、ログが書き込まれるファイルのファイル名に置き換えます。*DIR* は、ファイルが存在するディレクトリー名に置き換えます。*DIR* の値はパス変数になります。

例**12.60** ログハンドラーが書き込むファイルの指定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="file", value={"path"=>"accounts_trace.log",
"relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}
```

ハンドラーの出力に使用されるフォーマッターの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-
attribute(name="formatter", value="FORMAT")
```

HANDLER *をログハンドラーの名前に置き換えます。* ***FORMAT*** *は、設定ファイルで指定されているフォーマッター文字列またはフォーマッターの名前に置き換えます。*

例 ***12.61*** *ハンドラーの出力に使用されるフォーマッターの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="formatter", value="%d{HH:mm:ss,SSS} %-5p
(%c) [%t] %s%E%n")
{"outcome" => "success"}
```

各ログファイルの最大サイズの設定

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-
attribute(name="rotate-size", value="SIZE")
```

HANDLER *をログハンドラーの名前に置き換えます。* ***SIZE*** *はファイルの最大サイズに置き換えます。*

例 ***12.62*** *各ログファイルの最大サイズの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-
handler=PERIODIC_SIZE:write-attribute(name="rotate-size", value="50m")
{"outcome" => "success"}
```

保持するバックアップログの最大数の設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="max-backup-index", value="NUMBER")
```

HANDLER をログハンドラーの名前に置き換えます。**NUMBER** を、保持するログファイルの数に置き換えます。

例**12.63** 保持するバックアップログの最大数の設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="max-backup-index", value="5")
{"outcome" => "success"}
```

periodic-size-rotating-file-handler で **rotate-on-boot** オプションを設定します。

デフォルトは **false** で、サーバーの再起動時に新しいログファイルは作成されません。

これを変更するには、次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="rotate-on-boot", value="BOOLEAN")
```

HANDLER を **Periodic-size-rotating-file-handler** ログハンドラー の名前に置き換えます。再起動時に新しい **Periodic-size-rotating-file-handler** ログファイル を作成する場合は、**BOOLEAN** を **true** に置き換えます。

例**12.64** サーバーの再起動時に新しい **Periodic-size-rotating-file-handler** ログファイルを作成するよう指定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="rotate-on-boot", value="true")
{"outcome" => "success"}
```

ローテーションされるログの接尾辞の設定

次の構文で **write-attribute** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:write-attribute(name="suffix", value="SUFFIX")
```

HANDLER をログハンドラーの名前に置き換えます。**SUFFIX** を、必要な接尾辞の文字列に置き換えます。

例**12.65** ローテーションされるログの接尾辞の設定

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")
{"outcome" => "success"}
```

ログハンドラーの削除

以下の構文で **remove** 操作を使用します。

```
/subsystem=logging/periodic-size-rotating-file-handler=HANDLER:remove
```

HANDLER をログハンドラーの名前に置き換えます。

例**12.66** ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-size-rotating-file-handler=PERIODIC_SIZE:remove
{"outcome" => "success"}
```

バグの報告

12.3.8. CLI での非同期ログハンドラーの設定

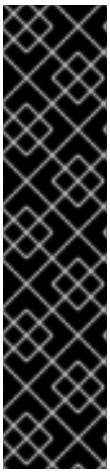
非同期ログハンドラーは、**CLI** で追加、削除、および編集できます。

非同期ログハンドラーを設定するために実行するタスクは以下のとおりです。

- 新しい非同期ログハンドラーの追加
- 非同期ログハンドラーの設定表示

- *ログレベルの変更*
- *キューの長さの設定*
- *オーバーフローアクションの設定*
- *サブハンドラーの追加*
- *サブハンドラーの削除*
- *非同期ログハンドラーの削除*

これらの各タスクについては以下で説明されています。



重要

*ログハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは /**subsystem=logging**/ ではなく /**subsystem=logging/logging-profile=NAME**/ になります。*

*管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。/**subsystem=logging**/ は /**profile=NAME/subsystem=logging**/ に置き換えます。*

新しい非同期ログハンドラーの追加

*以下の構文で **add** 操作を使用します。*

```
/subsystem=logging/async-handler=HANDLER:add(queue-length="LENGTH")
```

***HANDLER** をログハンドラーの名前に置き換えます。**LENGTH** を、キューに保持できるログリクエストの最大数に置き換えます。*

*例**12.67** 新しい非同期ログハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add(queue-
length="10")
{"outcome" => "success"}
```

非同期ログハンドラーの設定表示

以下の構文で ***read-resource*** *操作を使用します。*

```
/subsystem=logging/async-handler=HANDLER:read-resource
```

HANDLER *をログハンドラーの名前に置き換えます。*

例 ***12.68*** *非同期ログハンドラーの設定表示*

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:read-resource
{
    "outcome" => "success",
    "result" => {
        "enabled" => true,
        "encoding" => undefined,
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "ALL",
        "name" => "NFS_LOGS",
        "overflow-action" => "BLOCK",
        "queue-length" => "10",
        "subhandlers" => undefined
    },
    "response-headers" => {"process-state" => "reload-required"}
}
```

ログレベルの変更

次の構文で ***write-attribute*** *操作を使用します。*

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="level",
value="LOG_LEVEL_VALUE")
```

HANDLER *をログハンドラーの名前に置き換えます。* ***LOG_LEVEL_VALUE*** *を設定するログレベルに置き換えます。*

例 ***12.69*** *ログレベルの変更*

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="level", value="INFO")
{"outcome" => "success"}
```

キューの長さの設定

*次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="queue-length", value="LENGTH")
```

***HANDLER** をログハンドラーの名前に置き換えます。**LENGTH** を、キューに保持できるログリクエストの最大数に置き換えます。*

*この変更を反映するには、**JBoss EAP 6** を再起動する必要があります。*

*例**12.70** キューの長さの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="queue-length", value="150")
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

オーバーフローアクションの設定

*次の構文で **write-attribute** 操作を使用します。*

```
/subsystem=logging/async-handler=HANDLER:write-attribute(name="overflow-action", value="ACTION")
```

***HANDLER** をログハンドラーの名前に置き換えます。**ACTION** は、**DISCARD** または **BLOCK** に置き換えます。*

*例**12.71** オーバーフローアクションの設定*

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="overflow-action", value="DISCARD")
{"outcome" => "success"}
```

サブハンドラーの追加

次の構文で **add-handler** 操作を使用します。

```
/subsystem=logging/async-handler=HANDLER:add-handler(name="SUBHANDLER")
```

HANDLER をログハンドラーの名前に置き換えます。**SUBHANDLER** を、この非同期ハンドラーのサブハンドラーとして追加するログハンドラーの名前に置き換えます。

例**12.72** サブハンドラーの追加

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add-handler(name="NFS_FILE")
{"outcome" => "success"}
```

サブハンドラーの削除

次の構文で **remove-handler** 操作を使用します。

```
/subsystem=logging/async-handler=HANDLER:remove-handler(name="SUBHANDLER")
```

HANDLER をログハンドラーの名前に置き換えます。**SUBHANDLER** を、削除するサブハンドラーの名前に置き換えます。

例**12.73** サブハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove-handler(name="NFS_FILE")
{"outcome" => "success"}
```

非同期ログハンドラーの削除

以下の構文で **remove** 操作を使用します。

```
/subsystem=logging/async-handler=HANDLER:remove
```

HANDLER をログハンドラーの名前に置き換えます。

例**12.74** 非同期ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove
{"outcome" => "success"}
```

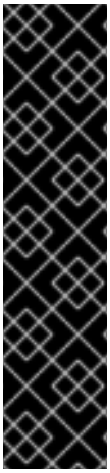
バグの報告

12.3.9. CLI でのカスタムハンドラーの設定

カスタムハンドラーは **CLI** で追加、削除、および編集できます。

カスタムハンドラーを設定するために実行する主なタスクは以下のとおりです。

- 新しいカスタムハンドラーを追加します。
- カスタムハンドラーの設定表示
- ログレベルを設定します。
- カスタムハンドラーの削除



重要

カスタムハンドラーをスタンドアロンシステムのロギングプロファイルに設定する場合、設定パスのルートは **/subsystem=logging/** ではなく **/subsystem=logging/logging-profile=NAME/** になります。

管理対象ドメインでは、使用するプロファイルを指定する必要があります。管理対象ドメインの設定パスの先頭にプロファイル名を追加する必要があります。**/subsystem=logging/** は **/profile=NAME/subsystem=logging/** に置き換えます。

新しいカスタムハンドラーの追加

*以下の構文で **add** 操作を使用します。*

```
/subsystem=logging/custom-handler="MyCustomHandler":add
```

*例**12.75** 新しいカスタムハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=logging/custom-
handler="MyCustomHandler":add(class="JdbcLogger",module="com.MyModule",formatter="%d{
HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",properties=
{"maxNumberOfDays"=>"90","fileName"=>"custom.log","compressBackups"=>"true"})
```

カスタムハンドラーの表示

*以下の構文で **read-resource** 操作を使用します。**CUSTOMHANDLER** をカスタムハンドラーの名前に置き換えます。*

```
/subsystem=logging/custom-handler=CUSTOMHANDLER:read-resource
```

*例**12.76** カスタムハンドラー設定の表示*

```
[standalone@localhost:9999 /] /subsystem=logging/custom-handler="MyCustomHandler":read-
resource
{
    "outcome" => "success",
    "result" => {
        "autoflush" => true,
        "enabled" => true,
        "encoding" => undefined,
        "filter" => undefined,
        "filter-spec" => undefined,
        "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
        "level" => "INFO",
        "name" => "CONSOLE",
        "named-formatter" => "COLOR-PATTERN",
        "target" => "System.out"
    }
}
```

ログレベルの設定

*次の構文で **write-attribute** 操作を使用します。**CUSTOMHANDLER** はログカテゴリーの名前に、**LEVEL** は設定するログレベルに置き換えます。*

```
/subsystem=logging/custom-handler=CUSTOMHANDLER:write-attribute(name="level",
value="INFO")
```

例**12.77** ログレベルの設定

```
[standalone@localhost:9999 /] /subsystem=logging/custom-
handler="MyCustomHandler":write-attribute(name="level", value="TRACE")
{"outcome" => "success"}
```

カスタムハンドラーの削除

以下の構文で **remove** 操作を使用します。**CUSTOMHANDLER** を、削除するカスタムハンドラーの名前に置き換えます。

```
/subsystem=logging/custom-handler=CUSTOMHANDLER:remove
```

例**12.78** カスタムハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/custom-
handler="MyCustomHandler":remove
{"outcome" => "success"}
```

バグの報告

12.3.10. CLI での Syslog ハンドラーの設定

JBoss EAP 6 のログマネージャーに **Syslog** ハンドラーが含まれるようになりました。**Syslog** ハンドラーは、**Syslog** プロトコル（**RFC-3164** または **RFC-5424**）をサポートするリモートロギングサーバーにメッセージを送信するために使用できます。これにより、複数のアプリケーションのログメッセージを同じサーバーに送信できます。これにより、複数のアプリケーションのログメッセージを解析できます。このトピックでは、管理 **CLI** を使用して **Syslog** ハンドラーを作成および設定する方法と、利用可能な設定オプションについて説明します。

Syslog ハンドラー属性の詳細は、「管理インターフェース監査ロギングリファレンス」 を参照してください。

前提条件

- 管理 **CLI** のアクセスおよび適切なパーミッション。

手順12.4 Syslog ハンドラーの追加

- 以下のコマンドを実行して **syslog** ハンドラーを追加します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:add
```

手順12.5 Syslog ハンドラーの設定

- 以下のコマンドを実行して **Syslog** ハンドラー属性を設定します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:write-
attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

手順12.6 Syslog ハンドラーの削除

- 以下のコマンドを実行して既存の **syslog** ハンドラーを削除します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:remove
```

バグの報告

12.3.11. CLI でのカスタムログフォーマッターの設定

概要

*「ログフォーマッター構文」 で指定したログフォーマッター構文の他に、任意のログハンドラーで使用するカスタムログフォーマッターを作成できます。この手順では、コンソールログハンドラーの **XML** フォーマッターを作成して説明します。*

前提条件

- ***JBoss EAP 6** サーバーの管理 **CLI** へのアクセス。*
- *以前設定されたログハンドラー。この手順例では、**Console** ログハンドラーを使用します。*

*手順12.7 ログハンドラーのカスタム **XML** フォーマッターの設定*

1. カスタムフォーマッターを作成します。

   この例では、以下のコマンドで **java.util.logging.XMLFormatter** クラスを使用する **XML_FORMATTER** という名前のカスタムフォーマッターを作成します。

   ```
   [standalone@localhost:9999 /] /subsystem=logging/custom-formatter=XML_FORMATTER:add(class=java.util.logging.XMLFormatter, module=org.jboss.logmanager)
   ```

2. 使用したいログハンドラーのカスタムフォーマッターを登録します。

   この例では、前の手順のフォーマッターがコンソールログハンドラーに追加されます。

   ```
   [standalone@localhost:9999 /] /subsystem=logging/console-handler=HANDLER:write-attribute(name=named-formatter, value=XML_FORMATTER)
   ```

3. **JBoss EAP 6** サーバーを再起動し、変更を反映します。

   ```
   [standalone@localhost:9999 /] shutdown --restart=true
   ```

結果

カスタム **XML** フォーマッターがコンソールログハンドラーに追加されます。コンソールログの出力は **XML** でフォーマットされます。以下に例を示します。

```
<record>
  <date>2014-03-11T13:02:53</date>
  <millis>1394539373833</millis>
  <sequence>116</sequence>
  <logger>org.jboss.as</logger>
  <level>INFO</level>
  <class>org.jboss.as.server.BootstrapListener</class>
  <method>logAdminConsole</method>
  <thread>282</thread>
  <message>JBAS015951: Admin console listening on http://%s:%d</message>
  <param>127.0.0.1</param>
  <param>9990</param>
</record>
```

バグの報告

12.4. デプロイメントごとのロギング

12.4.1. デプロイメントごとのロギング

デプロイメントごとのロギングを使用すると、開発者はアプリケーションのロギング設定を事前に設定できます。アプリケーションがデプロイされると、定義された設定に従ってロギングが開始されます。この設定によって作成されたログファイルにはアプリケーションの動作に関する情報のみが含まれます。

この方法では、システム全体のロギングを使用する利点と欠点があります。利点は、**JBoss EAP** インスタンスの管理者がロギングを設定する必要がないことです。欠点は、デプロイメントごとのロギング設定は起動時に読み取り専用であるため、実行時に変更できないことです。

バグの報告

12.4.2. デプロイメントごとのロギングの無効化

手順**12.8** デプロイメントごとのロギングの無効化

- デプロイメントごとのロギングを無効にする方法は **2** つあります。**1** つはすべてのバージョンの **JBoss EAP 6** で動作しますが、もう **1** つは **JBoss EAP 6.3** 以降でのみ動作します。

  - **JBoss EAP 6** (全バージョン)

    システムプロパティーを追加します。

    ```
    org.jboss.as.logging.per-deployment=false
    ```

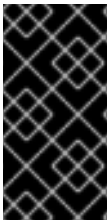
  - **JBoss EAP 6.3** (およびそれ以降のバージョン)

    jboss-deployment-structure.xml ファイルを使用して **logging** サブシステムを除外します。詳細は、『**開発ガイド**』の「『**サブシステムをデプロイメントから除外**」を』参照してください。

バグの報告

12.5. ロギングプロファイル

12.5.1. ロギングプロファイル



重要

ロギングプロファイルは、**6.1.0** 以降のバージョンでのみ利用できます。管理コンソールを使用して設定することはできません。

ロギングプロファイルは、デプロイされたアプリケーションに割り当てることができる独立したロギング設定のセットです。通常の **logging** サブシステムと同様に、ロギングプロファイルはハンドラー、カテゴリー、ルートロガーを定義できますが、他のプロファイルやメインの **logging** サブシステムを参照できません。設定を容易にするため、ロギングプロファイルは **logging** サブシステムと似ています。

ロギングプロファイルを使用すると、管理者は他のロギング設定に影響を与えずに **1** つ以上のアプリケーションに固有するロギング設定を作成できます。各プロファイルはサーバー設定で定義されるため、ロギング設定を変更しても影響を受けるアプリケーションを再デプロイする必要はありません。

各ロギングプロファイルに含めることができる設定は次のとおりです。

- 一意な名前これは必須です。
- 任意の数のログハンドラー。
- 任意の数のログカテゴリー。
- 最大 **1** つのルートロガー。

アプリケーションは **logging-profile** 属性を使用して **MANIFEST.MF** ファイルで使用するロギングプロファイルを指定できます。

バグの報告

12.5.2. CLI を使用した新しいロギングプロファイルの作成

以下の **CLI** コマンドを使用して新しいロギングプロファイルを作成できます。**NAME** は必要なプロファイル名に置き換えます。

```
/subsystem=logging/logging-profile=NAME:add
```

これにより、ハンドラー、カテゴリー、およびルートロガーを追加できる新しい空のプロファイルが作成されます。

バグの報告

12.5.3. CLI を使用したロギングプロファイルの設定

メインロギングシステムを使用した場合とほぼ同じ構文を用いて、ログハンドラー、カテゴリーおよびルートロガーを使用してロギングプロファイルを設定できます。

メインロギングサブシステムの設定とロギングプロファイルの設定で異なる点は以下の **2** つのみです。

1. ルート設定パスは以下のとおりです。 **/subsystem=logging/logging-profile=NAME**
2. ロギングプロファイルに他のロギングプロファイルを追加できません。

以下のロギング管理タスクを参照してください。

- 「**CLI** でのルートロガーの設定」
- 「**CLI** でのログカテゴリー設定」

- 「CLI でのコンソールログハンドラーの設定」
- 「CLI でのファイルログハンドラーの設定」
- 「CLI での周期ログハンドラーの設定」
- 「CLI でのサイズログハンドラーの設定」
- 「CLI での非同期ログハンドラーの設定」

例12.79 ロギングプロファイルの作成および設定

ロギングプロファイルの作成およびカテゴリーとファイルログハンドラーの追加。

1. プロファイルを作成します。

```
/subsystem=logging/logging-profile=accounts-app-profile:add
```

2. ファイルハンドラーの作成

```
/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})
```

```
/subsystem=logging/logging-profile=accounts-app-profile/file-handler=ejb-trace-file:write-attribute(name="level", value="DEBUG")
```

3. ロガーカテゴリーの作成

```
/subsystem=logging/logging-profile=accounts-app-profile/logger=com.company.accounts.ejbs:add(level=TRACE)
```

4. ファイルハンドラーをカテゴリーへ割り当て

```
/subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-trace-file")
```

バグの報告

12.5.4. アプリケーションでのロギングプロファイルの指定

アプリケーションは使用するロギングプロファイルを **MANIFEST.MF** ファイルに指定します。

要件:

1. このアプリケーションが使用するサーバー上に設定されたロギングプロファイルの名前を知っている必要があります。サーバー管理者に対し、使用するプロファイルの名前を要求します。

手順12.9 ロギングプロファイル設定のアプリケーションへの追加

- **MANIFEST.MF**の編集

  アプリケーションに **MANIFEST.MF** ファイルがない場合： 以下の内容が含まれるファイルを作成します。**NAME** は必要なプロファイル名に置き換えます。

  ```
  Manifest-Version: 1.0
  Logging-Profile: NAME
  ```

  アプリケーションに **MANIFEST.MF** ファイルがすでにある場合は、以下の行をそのファイルに追加します。**NAME** は必要なプロファイル名に置き換えます。

  ```
  Logging-Profile: NAME
  ```

注記

Maven および **maven-war-plugin** を使用している場合は、**MANIFEST.MF** ファイルを **src/main/resources/META-INF/** に置き、以下の設定を **pom.xml** ファイルに追加します。

```
<plugin>
  <artifactId>maven-war-plugin</artifactId>
  <configuration>
    <archive>
      <manifestFile>src/main/resources/META-INF/MANIFEST.MF</manifestFile>
    </archive>
  </configuration>
</plugin>
```

アプリケーションがデプロイされると、ログメッセージに対して指定されたロギングプロファイルの設定を使用します。

バグの報告

12.5.5. ロギングプロファイル設定の例

この例は、ロギングプロファイルの設定と、そのプロファイルを使用するアプリケーションを示しています。**CLI** セッション、生成された **XML** 設定、およびアプリケーションの **MANIFEST.MF** ファイルが表示されます。

ロギングプロファイルの例には以下が指定されています。

- **Name** は **accounts-app-profile** です。
- ログカテゴリーは **com.company.accounts.ejbs** です。
- ログレベル **TRACE**
- ログハンドラーは、**ejb-trace.log** ファイルを使用するファイルハンドラーです。

例**12.80 CLI** セッション

```
localhost:bin user$ ./jboss-cli.sh -c
[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile:add
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile/file-
handler=ejb-trace-file:add(file={path=>"ejb-trace.log", "relative-to"=>"jboss.server.log.dir"})
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-profile/file-
handler=ejb-trace-file:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add(level=TRACE)
{"outcome" => "success"}

[standalone@localhost:9999 /] /subsystem=logging/logging-profile=accounts-app-
profile/logger=com.company.accounts.ejbs:add-handler(name="ejb-trace-file")
{"outcome" => "success"}
```

例**12.81 XML** の設定

```
<logging-profiles>
  <logging-profile name="accounts-app-profile">
    <file-handler name="ejb-trace-file">
      <level name="DEBUG"/>
      <file relative-to="jboss.server.log.dir" path="ejb-trace.log"/>
    </file-handler>
    <logger category="com.company.accounts.ejbs">
      <level name="TRACE"/>
      <handlers>
        <handler name="ejb-trace-file"/>
      </handlers>
    </logger>
  </logging-profile>
</logging-profiles>
```

例**12.82** アプリケーションの **MANIFEST.MF** ファイル

```
Manifest-Version: 1.0
Logging-Profile: accounts-app-profile
```

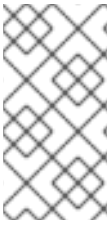
バグの報告

12.6. ロギング設定プロパティー

12.6.1. ルートロガーのプロパティー

表12.6 ルートロガーのプロパティー

プロパティー	データタイプ	説明
level	文字列	ルートロガーが記録するログメッセージの最大レベル。
handlers	String[]	ルートロガーによって使用されるログハンドラーの一覧。
filter-spec	文字列	フィルターを定義する式の値。式 **not（match("JBAS.*")）** は、パターンに一致しないログエントリーを除外するフィルターを定義します。



注記

*ルートロガーに指定された **filter-spec** は他のハンドラーによって継承され **ません**。代わりに、ハンドラーごとに **filter-spec** を指定する必要があります。*

バグの報告

12.6.2. ログカテゴリーのプロパティー

表12.7 ログカテゴリーのプロパティー

プロパティー	データタイプ	説明
level	文字列	ログカテゴリーが記録するログメッセージの最大レベル。
handlers	String[]	ロガーに関連付けられたログハンドラーのリスト。
use-parent-handlers	ブール値	true に設定した場合、割り当てられた他のハンドラーだけでなく、ルートロガーのログハンドラーを使用します。
category	文字列	ログメッセージがキャプチャーされるログカテゴリー。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**

バグの報告

12.6.3. コンソールログハンドラーのプロパティー

表12.8 コンソールログハンドラーのプロパティー

プロパティー	データタイプ	説明
level	文字列	ログハンドラーが記録するログメッセージの最大レベル。
encoding	文字列	出力に使用する文字エンコーディングスキーム。
formatter	文字列	このログハンドラーで使用するログフォーマッター。
named-formatter	文字列	ハンドラーで使用する定義されたフォーマッターの名前。
target	文字列	ログハンドラーの出力があるシステム出力ストリーム。システムエラーストリームの場合は System.err または System.out のいずれかになります。
autoflush	ブール値	true に設定すると、ログメッセージは受信直後にハンドラーのターゲットに送信されます。
name	文字列	このログハンドラーの一意の ID。
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**

バグの報告

12.6.4. ファイルログハンドラープロパティー

表12.9 ファイルログハンドラープロパティー

プロパティー	データタイプ	説明
level	文字列	ログハンドラーが記録するログメッセージの最大レベル。
encoding	文字列	出力に使用する文字エンコーディングスキーム。
formatter	文字列	このログハンドラーで使用するログフォーマッター。
named-formatter	文字列	ハンドラーで使用する定義されたフォーマッターの名前。
append	ブール値	true に設定された場合、このハンドラーが書き込んだすべてのメッセージがファイル（すでに存在する場合）に追加されます。false に設定すると、アプリケーションサーバーが起動されるたびに、新しいファイルが作成されます。**append** を変更するには、サーバーを再起動する必要があります。

プロパティー	データタイプ	説明
autoflush	ブール値	true に設定すると、ログメッセージは受信直後にハンドラー割り当てファイルに送信されます。
name	文字列	このログハンドラーの一意の ID。
file	オブジェクト	このログハンドラーの出力が書き込まれるファイルを表すオブジェクト。これには、**relative-to** と **path** の 2 つの設定プロパティーがあります。
relative-to	文字列	これは、ファイルオブジェクトのプロパティーであり、ログファイルが書き込まれるディレクトリーです。ここでは JBoss EAP 6 のファイルパス変数を指定できます。**jboss.server.log.dir** 変数は、サーバーの **log/** ディレクトリーを参照します。
path	文字列	ファイルオブジェクトのプロパティーであり、ログメッセージが書き込まれるファイルの名前です。完全パスを決定するために、**relative-to** プロパティーの値に追加される相対パス名です。
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**

バグの報告

12.6.5. 周期ログハンドラープロパティー

表12.10 周期ログハンドラープロパティー

プロパティー	データタイプ	説明
append	ブール値	**true** に設定された場合、このハンドラーが書き込んだすべてのメッセージがファイル（すでに存在する場合）に追加されます。**false** に設定すると、アプリケーションサーバーが起動されるたびに、新しいファイルが作成されます。append への変更を反映するには、サーバーを再起動する必要があります。
autoflush	ブール値	**true** に設定すると、ログメッセージは受信直後にハンドラー割り当てファイルに送信されます。
encoding	文字列	出力に使用する文字エンコーディングスキーム。
formatter	文字列	このログハンドラーで使用するログフォーマッター。

プロパティー	データタイプ	説明
named-formatter	文字列	ハンドラーで使用する定義されたフォーマッターの名前。
level	文字列	ログハンドラーが記録するログメッセージの最大レベル。
name	文字列	このログハンドラーの一意の ID。
file	オブジェクト	このログハンドラーの出力が書き込まれるファイルを表すオブジェクト。これには、**relative-to** と **path** の 2 つの設定プロパティーがあります。
relative-to	文字列	これは、ファイルオブジェクトのプロパティーで、ログファイルが含まれるディレクトリーです。ここでは、ファイルパス変数を指定できます。**jboss.server.log.dir** 変数は、サーバーの **log/** ディレクトリーを参照します。
path	文字列	ファイルオブジェクトのプロパティーであり、ログメッセージが書き込まれるファイルの名前です。完全パスを決定するために、**relative-to** プロパティーの値に追加される相対パス名です。
接尾辞	文字列	この文字列はローテーションされたログのファイル名に追加され、ローテーションの頻度を決定するために使用されます。接尾辞の形式はドット(.)の後に **date String** が続きます。これは、**SimpleDateFormat** クラスによって解析できます。ログは、接尾辞で定義される最小時間単位に基づいてローテーションされます。**suffix** 属性で許可される最小時間の単位は分であることに注意してください。 たとえば、**接尾辞** の値が **.YYYY-MM-dd** の場合は、ログが毎日ローテーションされます。**server.log** という名前のログファイルと **接尾辞** の値が **.YYYY-MM-dd** の場合、2014 年 10 月 20 日にローテーションされたログファイルの名前は **server.log.2014-10-19 に** なります。Periodic ログハンドラーの場合、接尾辞には最小時間単位の以前の値が含まれます。この例では、**dd** の値は **19**（前の日）です。
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 **not（match("JBAS.*")）** はパターンに一致しないフィルターを定義します。

バグの報告

12.6.6. サイズログハンドラープロパティー

表12.11 サイズログハンドラープロパティー

プロパティー	データタイプ	説明
append	ブール値	**true** に設定された場合、このハンドラーが書き込んだすべてのメッセージがファイル（すでに存在する場合）に追加されます。**false** に設定すると、アプリケーションサーバーが起動されるたびに、新しいファイルが作成されます。append への変更を反映するには、サーバーを再起動する必要があります。
autoflush	ブール値	**true** に設定すると、ログメッセージは受信直後にハンドラー割り当てファイルに送信されます。
encoding	文字列	出力に使用する文字エンコーディングスキーム。
formatter	文字列	このログハンドラーで使用するログフォーマッター。
named-formatter	文字列	ハンドラーで使用する定義されたフォーマッターの名前。
level	文字列	ログハンドラーが記録するログメッセージの最大レベル。
name	文字列	このログハンドラーの一意の ID。
file	オブジェクト	このログハンドラーの出力が書き込まれるファイルを表すオブジェクト。これには、**relative-to** と **path** の 2 つの設定プロパティーがあります。
relative-to	文字列	これは、ファイルオブジェクトのプロパティーで、ログファイルが含まれるディレクトリーです。ここでは、ファイルパス変数を指定できます。**jboss.server.log.dir** 変数は、サーバーの **log/** ディレクトリーを参照します。
path	文字列	ファイルオブジェクトのプロパティーであり、ログメッセージが書き込まれるファイルの名前です。完全パスを決定するために、**relative-to** プロパティーの値に追加される相対パス名です。
rotate-size	整数	ログファイルがローテーションされる前に到達できる最大サイズです。数字に追加された単一の文字はサイズ単位を示します。バイトの場合は **b**、キロバイトの場合は **k**、メガバイトの場合は **m**、ギガバイトの場合は **g** になります。たとえば、50 メガバイトの場合は **50m** になります。

プロパティー	データタイプ	説明
max-backup-index	整数	保持するローテーションログの最大数。この数に達すると、古いログが再使用されます。デフォルト：1 JBoss EAP 6.4 から利用できます。**suffix** 属性が使用された場合は、ローテーションログファイルの接尾辞がローテーションアルゴリズムに含まれます。ログファイルがローテーションされると、名前が **name**+**suffix** で始まる最も古いファイルが削除され、残りのローテーションログファイルの数値接尾辞が増分され、新しくローテーションされたログファイルに数値接尾辞 **1** が提供されます。 ログファイル名 **server.log**、**接尾辞** の値が **.YYYY-mm**、および **max-backup-index** の値が 3 を想定します。ログファイルが 2 回ローテーションされると、ログファイル名は server.log、**server.log.2014-10.1、および server.log.2014-10.2 の可能性があります**。次回のログファイルがローテーションされると、**server.log.2014- 10. 2 が削除され、server.log.2014-10.1 の名前が server.log.2014 - 10.** 2 に変更され、新たにローテーションされたログファイルの名前は **server.log.2014-10.** 1 になります。
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**
rotate-on-boot	ブール値	**true** に設定されると、新しいログファイルがサーバーの起動時に作成されます。デフォルトは **false** です。
接尾辞	文字列	JBoss EAP 6.4 から利用できます。この文字列はローテーションログに追加される接尾辞に含まれます。*接尾辞* の形式はドット(.)の後に **SimpleDateFormat** クラスが解析できる日付文字列です。 たとえば、**server.log** という名前のログファイルと **接尾辞** の値が **.YYYY-mm** の場合、1 年 10 月 1 日にローテーションされる最初のログファイルの名前は **server.log.2014-10.1 になります**。この例では、接尾辞の **2014-10** 部分は接尾辞の値から派生しています。

バグの報告

12.6.7. Periodic Size rotating ログハンドラープロパティー

表12.12 Periodic Size ログハンドラープロパティー

プロパティー	データタイプ	説明
append	ブール値	**true** に設定された場合、このハンドラーが書き込んだすべてのメッセージがファイル（すでに存在する場合）に追加されます。**false** に設定すると、アプリケーションサーバーが起動されるたびに、新しいファイルが作成されます。append への変更を反映するには、サーバーを再起動する必要があります。
rotate-size	文字列	ログファイルがローテーションされる前に到達できる最大サイズです。
max-backup-index	整数	保持されるサイズローテーションログの最大数。この数に達すると、古いログが再使用されます。デフォルト：1 この設定は、ファイルサイズに基づいてローテーションされるログにのみ適用されます。 ファイルが日付形式のサフィックスでローテーションされている場合は、**max-backup-index** オプションを使用してパージされないことに注意してください。
接尾辞	文字列	この文字列はローテーションされたログのファイル名に追加され、ローテーションの頻度を決定するために使用されます。接尾辞の形式はドット(.)の後に **date String** が続きます。これは、**SimpleDateFormat** クラスによって解析できます。ログは、接尾辞で定義される最小時間単位に基づいてローテーションされます。**suffix** 属性で許可される最小時間の単位は分であることに注意してください。 たとえば、**接尾辞** の値が **.YYYY-MM-dd** の場合は、ログが毎日ローテーションされます。**server.log** という名前のログファイルと **接尾辞** の値が **.YYYY-MM-dd** の場合、2014 年 10 月 20 日にローテーションされたログファイルの名前は **server.log.2014-10-19 に** なります。Periodic ログハンドラーの場合、接尾辞には最小時間単位の以前の値が含まれます。この例では、**dd** の値は **19**（前の日）です。
autoflush	ブール値	**true** に設定すると、ログメッセージは受信直後にハンドラー割り当てファイルに送信されます。
file	オブジェクト	このログハンドラーの出力が書き込まれるファイルを表すオブジェクト。これには、**relative-to** と **path** の 2 つの設定プロパティーがあります。
relative-to	文字列	これは、ファイルオブジェクトのプロパティーで、ログファイルが含まれるディレクトリーです。ここでは、ファイルパス変数を指定できます。**jboss.server.log.dir** 変数は、サーバーの **log/** ディレクトリーを参照します。
path	文字列	ファイルオブジェクトのプロパティーであり、ログメッセージが書き込まれるファイルの名前です。完全パスを決定するために、**relative-to** プロパティーの値に追加される相対パス名です。

プロパティー	データタイプ	説明
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**
rotate-on-boot	ブール値	**true** に設定されると、新しいログファイルがサーバーの起動時に作成されます。デフォルトは **false** です。
formatter	文字列	このログハンドラーで使用するログフォーマッター。
level	文字列	ログハンドラーが記録するログメッセージの最小レベル。
name	文字列	このログハンドラーの一意の ID。
named-formatter	文字列	ハンドラーで使用する定義されたフォーマッターの名前。
encoding	文字列	出力に使用する文字エンコーディングスキーム。

バグの報告

12.6.8. 同期ログハンドラープロパティー

表12.13 同期ログハンドラープロパティー

プロパティー	データタイプ	説明
level	文字列	ログハンドラーが記録するログメッセージの最大レベル。
name	文字列	このログハンドラーの一意の ID。
encoding	文字列	出力に使用する文字エンコーディングスキーム。
formatter	文字列	このログハンドラーで使用するログフォーマッター。
queue-length	Integer	サブハンドラーが応答するときに、このハンドラーが保持するログメッセージの最大数。
overflow-action	文字列	キューの長さを超えたときにこのハンドラーがどのように応答するかを示します。これは **BLOCK** または **DISCARD** に設定できます。**BLOCK** により、キューでスペースが利用可能になるまでロギングアプリケーションが待機します。これは、非同期でないログハンドラーと同じ動作です。**DISCARD** により、ロギングアプリケーションは動作し続けますが、ログメッセージは削除されます。

プロパティー	データタイプ	説明
subhandlers	String[]	これは、この非同期ハンドラーがログメッセージを渡すログハンドラーの一覧です。
enabled	ブール値	**true** に設定された場合、ハンドラーが有効になり、通常とおり機能します。**false** に設定された場合、ログメッセージの処理時にハンドラーが無視されます。
filter-spec	文字列	フィルターを定義する式の値。式 not（match **("JBAS.*")はパターンに一致しないフィルターを定義します。**

バグの報告

12.7. ロギング用 XML 設定例

12.7.1. ルートロガーの XML 設定例

```
<root-logger>
  <level name="INFO"/>
  <handlers>
    <handler name="CONSOLE"/>
      <handler name="FILE"/>
  </handlers>
</root-logger>
```

バグの報告

12.7.2. ログカテゴリーの XML 設定例

```
<logger category="com.company.accounts.rec">
  <handlers>
    <handler name="accounts-rec"/>
  </handlers>
</logger>
```

バグの報告

12.7.3. コンソールログハンドラーの XML 設定例

```
<console-handler name="CONSOLE">
  <level name="INFO"/>
  <formatter>
```

```
    <pattern-formatter pattern="%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n"/>
  </formatter>
</console-handler>
```

バグの報告

12.7.4. ファイルログハンドラーの XML 設定例

```
<file-handler name="accounts-rec-trail" autoflush="true">
    <level name="INFO"/>
    <file relative-to="jboss.server.log.dir" path="accounts-rec-trail.log"/>
    <append value="true"/>
</file-handler>
```

バグの報告

12.7.5. 定期ログハンドラーの XML 設定例

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

バグの報告

12.7.6. サイズログハンドラーの XML 設定例

```
<size-rotating-file-handler name="accounts_debug" autoflush="false">
  <level name="DEBUG"/>
  <file relative-to="jboss.server.log.dir" path="accounts-debug.log"/>
  <rotate-size value="500k"/>
  <max-backup-index value="5"/>
  <append value="true"/>
</size-rotating-file-handler>
```

バグの報告

12.7.7. Periodic Size Rotating ログハンドラーの XML 設定例

```
<periodic-size-rotating-file-handler name="Periodic_size_rotating_Handler" autoflush="false">
```

```
  <level name="INFO"/>
  <file relative-to="jboss.server.log.dir" path="Sample.log"/>
  <max-backup-index value="1"/>
  <suffix value=".yyyy.MM.dd"/>
  <append value="false"/>
<periodic-size-rotating-file-handler>
```

バグの報告

12.7.8. 非同期ログハンドラーの XML 設定例

```
<async-handler name="Async_NFS_handlers">
  <level name="INFO"/>
  <queue-length value="512"/>
  <overflow-action value="block"/>
  <subhandlers>
    <handler name="FILE"/>
    <handler name="accounts-record"/>
  </subhandlers>
</async-handler>
```

バグの報告

第13章 INFINISPAN

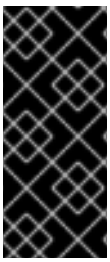
13.1. INFINISPAN

Infinispan は **Java** グリッドプラットフォームです。キャッシュされたデータを管理するための **JSR-107** と互換性のあるキャッシュインターフェースを提供します。

JBoss Enterprise Application Platform 6 では以下の **Infinispan** キャッシュコンテナが使用されます。

- **Web** （**Web** セッションクラスタリング）
- **EJB** （ステートフルセッション **Bean** クラスタリング）
- **hibernate (**エンティティーキャッシング**)**
- シングルトン キャッシングのシングルトン

各キャッシュコンテナーは「**repl**」と「**dist**」キャッシュを定義します。これらのキャッシュは、ユーザーアプリケーションで直接使用しないでください。



重要

ユーザーはキャッシュコンテナーおよびキャッシュをさらに追加し、**JNDI** 経由で参照できます。ただし、これは **JBoss Enterprise Application Platform 6** ではサポートされません。

Infinispan の機能や設定オプションの詳細は **Infinispan** のドキュメント を参照して『ください』。

バグの報告

13.2. クラスタリングモード

JBoss EAP 6 では Infinispan を使用する 2 つの方法でクラスタリングを設定できます。アプリケーションの正しい方法は要件によって異なります。各モードでは可用性、一貫性、信頼性、スケーラビリティーのトレードオフが発生します。クラスタリングモードを選択する前に、ネットワークで最も重要な点を特定し、これらの要件のバランスを取ることが必要となります。

レプリケートモード

レプリケートモードはクラスターの新規インスタンスを自動的に検出し、追加します。これらのインスタンスに加えられた変更は、クラスター上のすべてのノードにレプリケートされます。ネットワークでレプリケートされる情報量が多いため、通常 Replicated モードは小型のクラスターでの使用に最も適しています。UDP マルチキャストを使用するよう Infinispan を設定すると、ネットワークトラフィックの輻輳をある程度軽減できます。

ディストリビューションモード

Distribution (分散) モードでは、Infinispan はクラスターを線形にスケールできます。Distribution モードは一貫性のあるハッシュアルゴリズムを使用して、クラスター内で新しいノードを配置する場所を決定します。保持する情報のコピー数は設定可能です。保持するコピー数、データの持続性、およびパフォーマンスの持続性にはトレードオフが生じます。たとえば、保持するコピー数が多いほど、パフォーマンスへの影響が大きくなりますが、サーバーの障害時にデータが失われる可能性は低くなります。ハッシュアルゴリズムは、メタデータのマルチキャストや保存を行わずにエントリーを配置し、ネットワークトラフィックを軽減します。

クラスターサイズが 6-8 ノードを超える場合は Distribution(dist)モードをキャッシュストラテジーとして使用することを検討する必要があります。Distribution モードでは、データはクラスター内のすべてのノード（デフォルトの Replicated モード）ではなくノードのサブセットのみに分散されます。

同期および非同期のレプリケーション

レプリケーションは同期または非同期モードで実行でき、選択するモードは要件やアプリケーションモードによって異なります。同期のレプリケーションでは、ユーザーリクエストを処理するスレッドはレプリケーションが正常に実行されるまでブロックされます。レプリケーションが正常に行われる場合にのみ、応答がクライアントに返され、スレッドがリリースされます。同期レプリケーションはクラスターの各ノードからの応答を必要とするため、ネットワークトラフィックに影響を与えます。ただし、クラスターのすべてのノードへ確実に変更が加えられる利点があります。

非同期レプリケーションはバックグラウンドで実行されます。Infinispan はレプリケーションを実行するバックグラウンドスレッドによって使用されるレプリケーションキューを実装します。レプリケーションは時間ベースまたはキューのサイズのいずれかでトリガーされます。レプリケーションキューは、クラスターノード間の対話が行われないため、パフォーマンスを向上させることができます。非同期レプリケーションとのトレードオフは、非常に正確ではないことです。失敗したレプリケーションはログに書き込まれ、リアルタイムで通知されません。

バグの報告

13.3. キャッシュコンテナー

キャッシュコンテナー

キャッシュコンテナーはサブシステムによって使用されるキャッシュのリポジトリーです。**Infinispan** のデフォルトのキャッシュコンテナーは設定 **xml** ファイル（**standalone-ha.xml**、**standalone-full-ha.xml**、**domain.xml**）で定義されます。**1** つのキャッシュがデフォルトのキャッシュとして定義されます。これは、クラスタリングに使用されるキャッシュです。

例**13.1 standalone-ha.xml** 設定ファイルのキャッシュコンテナ定義

```
<subsystem xmlns="urn:jboss:domain:infinispan:1.5">
      <cache-container name="singleton" aliases="cluster ha-partition" default-cache="default">
        <transport lock-timeout="60000"/>
        <replicated-cache name="default" mode="SYNC" batching="true">
          <locking isolation="REPEATABLE_READ"/>
        </replicated-cache>
      </cache-container>
      <cache-container name="web" aliases="standard-session-cache" default-cache="repl"
module="org.jboss.as.clustering.web.infinispan">
       <transport lock-timeout="60000"/>
       <replicated-cache name="repl" mode="ASYNC" batching="true">
        <file-store/>
       </replicated-cache>
       <replicated-cache name="sso" mode="SYNC" batching="true"/>
        <distributed-cache name="dist" l1-lifespan="0" mode="ASYNC" batching="true">
         <file-store/>
        </distributed-cache>
       </cache-container>
```

各キャッシュコンテナーで定義されたデフォルトのキャッシュに注目してください。この例では、**Web** キャッシュコンテナーで **repl** キャッシュがデフォルトとして定義されます。そのため、**Web** セッションのクラスタリングでは **repl** キャッシュが使用されます。

キャッシュコンテナとキャッシュ属性は、管理コンソールまたは **CLI** コマンドを使用して設定できますが、キャッシュコンテナまたはキャッシュの名前を変更することは推奨されません。

キャッシュコンテナーの設定

Infinispan のキャッシュコンテナは **CLI** または管理コンソールを使用して設定できます。

手順**13.1** 管理コンソールでの **Infinispan** キャッシュコンテナの設定

1. *画面上部の **Configuration** タブを選択します。*

2. *ドメインモードの場合は、左上のドロップダウンメニューから **ha** または **full-ha** のいずれかを選択します。*

3. ***Subsystems** メニューを展開し、**Infinispan** メニューを展開します。**Cache Containers** を選択します。*

4. ***Cache Containers** テーブルからキャッシュコンテナーを選択します。*

5. *デフォルトのキャッシュコンテナの追加、削除、および設定*

   a. *新しいキャッシュコンテナーを作成するには、**Cache Containers** テーブルから **Add** をクリックします。*

   b. *キャッシュコンテナーを削除するには、**Cache Containers** 表のキャッシュコンテナーを選択します。削除 をクリックし、**OK** をクリックして確定します。*

   c. *キャッシュコンテナーをデフォルトとして設定するには、**Set Default** をクリックし、ドロップダウンリストからキャッシュコンテナー名を入力し、**Save** をクリックして確定します。*

6. *キャッシュコンテナーの属性を追加または更新するには、キャッシュコンテナー の表にあるキャッシュコンテナーを選択 し ます。画面の **Details** エリアの **Attributes**、**Transport**、および **Aliases** タブから **1** つを選択し、**Edit** をクリックします。**Attributes**、**Transport**、および **Aliases** タブの内容のヘルプは、**Need Help?** をクリックします。*

*手順**13.2** 管理 **CLI** での **Infinispan** キャッシュコンテナの設定*

1. *設定可能な属性のリストを取得するには、以下の **CLI** コマンドを入力します。*

```
/profile=profile name/subsystem=infinispan/cache-container=container name:read-resource
```

2.

管理 **CLI** を使用してキャッシュコンテナーを追加、削除、および更新できます。キャッシュコンテナーで使用するコマンドを実行する前に、管理 **CLI** コマンドで正しいプロファイルを使用するようにしてください。

a. キャッシュコンテナの追加

キャッシュコンテナを追加するには、以下の例に従ってコマンドを入力します。

```
/profile=profile-name/subsystem=infinispan/cache-container="cache container name":add
```

b. キャッシュコンテナの削除

キャッシュコンテナを削除するには、以下の例に従ってコマンドを入力します。

```
/profile=profile-name/subsystem=infinispan/cache-container="cache container name":remove
```

c. キャッシュコンテナ属性の更新

write-attribute 操作を使用して、新しい値を属性に書き込みます。タブ補完を使用すると、入力時のコマンド文字列の補完に役立つほか、利用可能な属性を明らかにできます。以下の例は、**statistics-enabled** を **true** に更新します。

```
/profile=profile name/subsystem=infinispan/cache-container=cache container name:write-attribute(name=statistics-enabled,value=true)
```

バグの報告

13.4. INFINISPAN の統計

監視目的で、**Infinispan** キャッシュおよびキャッシュコンテナーオブジェクトに関するランタイム統計を有効にできます。パフォーマンス上の理由で、統計の収集はデフォルトでは無効になっています。

警告

***Infinispan** の統計を有効にすると、**Infinispan** サブシステムのパフォーマンスに影響を及ぼす可能性があります。統計は必要な場合のみ有効にしてください。*

*統計収集は、各キャッシュコンテナー、キャッシュ、またはその両方に対して有効にできます。各キャッシュの統計オプションは **cache-container** のオプションを上書きします。キャッシュコンテナーの統計収集を無効または有効にすると、独自の設定が明示的に指定されている場合以外はそのコンテナーのすべてのキャッシュが設定を継承します。**cache-container** のみが統計に対して有効になっている場合、有用な統計を使用できます。*

バグの報告

13.5. INFINISPAN 統計収集の有効化

*統計収集は、起動設定ファイル（**standalone.xml**、**standalone-ha.xml**、**domain.xml**など）または管理 **CLI** から有効にできます。*

バグの報告

13.5.1. 起動設定ファイルでの Infinispan 統計収集の有効化

*手順**13.3** 起動設定ファイルでの **Infinispan** 統計の有効化*

- *属性 **statistics-enabled=VALUE** を **Infinispan** サブシステム下の必要な **cache-container** または **cache XML** タグに追加します。*

*例**13.2** キャッシュの統計収集の有効化*

```
<replicated-cache name="sso" mode="SYNC" batching="true" statistics-enabled="true"/>
```

*例**13.3** キャッシュコンテナーの統計収集の有効化*

```
<cache-container name="singleton" aliases="cluster ha-partition" default-cache="default"
statistics-enabled="true">
```

バグの報告

13.5.2. 管理 CLI での Infinispan 統計収集の有効化

手順13.4 管理 CLI での Infinispan 統計収集の有効化

以下の手順では、

- CACHE_CONTAINER が推奨される キャッシュコンテナー （例： web）です。
- CACHE_TYPE が推奨されるキャッシュタイプです（例： distributed-cache）。
- CACHE はキャッシュ名です（例： dist）。

1. 以下のコマンドを入力します。

例13.4 キャッシュの統計収集の有効化

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:write-
attribute(name=statistics-enabled,value=true)
```

例13.5 キャッシュコンテナーの統計収集の有効化

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:write-
attribute(name=statistics-enabled,value=true)
```

2. *以下のコマンドを実行し、サーバーをリロードします。*

```
reload
```

注記

属性を未定義にするには、以下のコマンドを実行します。

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:undefine-attribute(name=statistics-enabled)
```

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:undefine-attribute(name=statistics-enabled)
```

バグの報告

13.5.3. Infinispan 統計収集の有効化を検証

手順13.5 Infinispan 統計収集の有効化を検証

統計収集がキャッシュまたは キャッシュ コンテナー で有効になっているかどうかを確認するかどうかに応じて、以下の管理 CLI コマンドのいずれかを使用します。

-

- キャッシュの場合

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER/CACHE_TYPE=CACHE:read-attribute(name=statistics-enabled)
```

- キャッシュコンテナーの場合

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:read-attribute(name=statistics-enabled)
```

バグの報告

13.6. WEB セッションレプリケーションの分散キャッシュモードへの切り替え

Infinispan サブシステムの **web** キャッシュコンテナを **dist** から **repl** に変更します。**owners** 属性は、セッションデータを保持するクラスターノードの数を定義します。その **1** つはプライマリー所有者と呼ばれ、もう **1** つはバックアップ所有者です。

キャッシュモードを **dist** プロファイルが使用されていると仮定して、**3** つの所有者でキャッシュモードを更新する **CLI** コマンドは次のとおりです。**ha**

```
/subsystem=infinispan/cache-container=web/:write-attribute(name=default-cache,value=dist)
/subsystem=infinispan/cache-container=web/distributed-cache=dist/:write-attribute(name=owners,value=3)
```

上記の **CLI** を実行すると、以下の出力が表示されます。

```
<cache-container name="web" aliases="standard-session-cache" default-cache="dist"
module="org.jboss.as.clustering.web.infinispan">
  <transport lock-timeout="60000"/>
  ...
  <distributed-cache name="dist" owners="3" l1-lifespan="0" mode="ASYNC"
batching="true">
    <file-store/>
  </distributed-cache>
</cache-container>
```

バグの報告

13.7. JGROUPS

13.7.1. JGroups

JGroups は、システムの信頼性に問題がある場合に開発者が信頼できるメッセージングアプリケーションを作成できるメッセージングツールキットです。**JGroups** を使用すると、ノードがお互いにメッセージを送信できるクラスターを作成できます。

JGroups サブシステムは、クラスター内のサーバーが相互に対話するためのすべての通信メカニズムを提供します。**EAP** には **2** つの **JGroups** スタックが事前設定されています。

- **udp -** クラスターのノードは **UDP(User Datagram Protocol)**マルチキャストを使用してお互いに通信します。通常、**UDP** は **TCP** よりも高速ですが、信頼性は低くなります。
- **tcp -** クラスターのノードは **TCP(Transmission Control Protocol)**を使用してお互いに通信します。**TCP** は **UDP** よりも遅い傾向がありますが、データを宛先に確実に配信します。

事前定義されたスタックを使用できますが、システムの特定要件に見合うよう独自に定義することもできます。

バグの報告

13.8. JGROUPS トラブルシューティング

13.8.1. ノードがクラスターを形成しない

マシンで **IP** マルチキャストが正しくセットアップされていることを確認します。**McastReceiverTest** と **McastSenderTest** の検出に使用できるテストプログラムは **2** つあります。以下に例を示します。

```
java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastReceiverTest -mcast_addr
230.11.11.11 -port 5555 -bind_addr YOUR_BIND_ADDRESS
```

次に、別のウィンドウで **McastSenderTest** を開始します。

```
java -cp EAP_HOME/bin/client/jboss-client.jar org.jgroups.tests.McastSenderTest -mcast_addr
230.11.11.11 -port 5555 -bind_addr YOUR_BIND_ADDRESS
```

特定のネットワークインターフェースカード**(NIC)**にバインドする場合は、**-bind_addr 192.168.0.2** を使用します。**192.168.0.2** は、バインドする **NIC** の **IP** アドレスに置き換えます。送信側と受信側の両方にこのパラメーターを使用します。

McastSenderTest ウィンドウに入力でき、**McastReceiverTest** ウィンドウで出力を確認できるはずです。そうでない場合は、送信者で **-ttl 32** を使用します。それでも失敗する場合は、システム管理者を参照して、**IP** マルチキャストを正しく設定できるように管理者に連絡し、選択したインターフェースでマルチキャストが機能するか、またはマシンに複数のインターフェースがある場合は、正しいインターフェースを要求するように管理者に依頼してください。クラスターの各マシンでマルチキャストが適切に動作したら、送信側と受信側を別々のマシンに配置し、テストを繰り返します。

バグの報告

13.8.2. FD にハートビートが欠落する原因

ハートビートが一定時間 **T**（**timeout** および **max_tries** によって定義）を受け取っていないため、**FD** がメンバーに疑われる場合もあります。これには、**A**、**B**、**C**、**D** のクラスターなど、複数の理由が考えられます（**A** に **B**、**B** が **C**、**C** が **D** および **D** に **ping** される可能性があることに注意してください）。

- **B** または **C** が **CPU** の **100%** で稼働している（**T** 秒以上）。そのため、**C** がハートビート認証を **B** に送信しても、**B** が **100%** であるため処理できない場合があります。

- *B または C は、上記と同様にガベッジコレクションです。*

- *上記の 2 つのケースの組み合わせ*

- *ネットワークによるパケットの損失が発生する場合。これは通常、ネットワーク上に大量のトラフィックがあり、スイッチがパケットドロップを開始します（通常は最初にブロードキャスト、次に IP マルチキャスト、TCP パケットが最後にブロードキャストされます）。*

- *B または C がコールバックを処理する場合。C がチャンネル上でリモートメソッド呼び出しを受信し、T+1 秒かかるとします。この間、C はハートビートなどの他のメッセージを処理しません。そのため、B はハートビート認証を受け取らず、C は疑われます。*

バグの報告

第14章 JVM

14.1. JVM

14.1.1. JVM 設定

Java Virtual Machine(JVM)設定の設定は、管理対象ドメインとスタンドアロンサーバーインスタンスによって異なります。管理対象ドメインでは、**JVM** 設定は **host.xml** および **domain.xml** 設定ファイルで宣言され、サーバープロセスを起動および停止するドメインコントローラーコンポーネントによって決定されます。スタンドアロンサーバーインスタンスでは、起動時にサーバーの起動プロセスがコマンドライン設定を渡すことができます。これらはコマンドラインまたは管理コンソールの システムプロパティー画面を 使用して宣言できます。

注記

JAVA_OPTS で設定されていないシステムプロパティーは、起動時に **JBoss** モジュールによって使用されるプロパティーには使用できません。また、**java.util.logging.manager** などの他の **JVM** プロパティーには使用できません。

管理対象ドメイン

管理対象ドメインの重要な機能は、複数のレベルで **JVM** 設定を定義する機能です。カスタム **JVM** 設定は、ホストレベル、サーバーグループ、またはサーバーインスタンスにより設定できます。より特殊な子要素は親設定を上書きし、グループまたはホストレベルで除外せずに特定のサーバー設定を宣言できます。これにより、設定が設定ファイルで宣言されるか、実行時に渡されるまで、親設定を他のレベルで継承できます。

例**14.1** ドメイン設定ファイルの **JVM** 設定

以下の例は、**domain.xml** 設定ファイルのサーバーグループに対する **JVM** 宣言を示しています。

```
<server-groups>
    <server-group name="main-server-group" profile="default">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="standard-sockets"/>
    </server-group>
    <server-group name="other-server-group" profile="default">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
```

```
        <socket-binding-group ref="standard-sockets"/>
    </server-group>
</server-groups>
```

このインスタンスでは、**main-server-group** という名前のサーバーグループが **64** メガバイトのヒープサイズと最大ヒープサイズ **512** メガバイトを宣言します。このグループに属するサーバーは、この設定を継承します。グループ全体、ホスト、または個々のサーバーの設定を変更できます。

例**14.2** ホスト設定ファイルのドメイン設定

以下の例は、**host.xml** 設定ファイルのサーバーグループの **JVM** 宣言を示しています。

```
<servers>
  <server name="server-one" group="main-server-group" auto-start="true">
    <jvm name="default"/>
  </server>
  <server name="server-two" group="main-server-group" auto-start="true">
    <jvm name="default">
      <heap size="64m" max-size="256m"/>
    </jvm>
    <socket-bindings port-offset="150"/>
  </server>
  <server name="server-three" group="other-server-group" auto-start="false">
    <socket-bindings port-offset="250"/>
  </server>
</servers>
```

この場合、**server-two** という名前のサーバーは **main-server-group** という名前のサーバーグループに属し、**default JVM** グループから **JVM** 設定を継承します。上記の例では、**main-server-group** のメインヒープサイズは **512** メガバイトに設定されています。最大ヒープサイズ **256** メガバイトを宣言すると、**server-two** は **domain.xml** 設定を上書きし、必要に応じてパフォーマンスを微調整できます。

実行時のスタンドアロンサーバー設定

スタンドアロンサーバーインスタンスの **JVM** 設定は、サーバーを起動する前に **JAVA_OPTS** 環境変数を設定することでランタイム時に宣言できます。**Linux** コマンドラインで **JAVA_OPTS** 環境変数を設定する例は次のとおりです。

```
[user@host bin]$ export JAVA_OPTS="-Xmx1024M"
```

次のように、同じ設定を **Microsoft Windows** 環境で使用できます。

```
C:\> set JAVA_OPTS="Xmx1024M"
```

または、**JVM** に渡すオプションの例が含まれる **EAP_HOME/bin** フォルダーにある **standalone.conf** ファイルに **JVM** 設定を追加することもできます。

警告

JAVA_OPTS 環境変数を設定すると、**JAVA_OPTS** 環境変数のデフォルト値が再定義されます。これにより、**JBoss EAP** の起動が中断または中断される可能性があります。

バグの報告

14.1.2. 管理コンソールでの JVM 状態の表示

前提条件

- 「**JBoss EAP 6** をスタンドアロンサーバーとして起動」
- 「**JBoss EAP 6** を管理対象ドメインとして起動」
- 「管理コンソールへのログイン」

スタンドアロンサーバーまたは管理対象ドメインの管理コンソールに **Java** 仮想マシン**(JVM)**ステータスを表示できます。コンソールは、サーバーのヒープ使用量、非ヒープ使用、およびスレッドの使用状況を表示します。統計はリアルタイムでは表示されませんが、コンソール表示を更新して **JVM** リソースの最新の概要を提供できます。

JVM の状態には次の値が表示されます。

表**14.1 JVM** 状態属性

タイプ	説明
Max	メモリー管理に使用できるメモリーの最大量。最大エバラブメモリーは、軽量グレイのバーで表示されます。
Used	使用されたメモリーの量。使用中のメモリー容量は、ダクグレーのバーで表示されます。
Committed	Java 仮想マシンが使用するために確保されるメモリー量。コミットされたメモリーは dark grey バーで表示されます。

*手順**14.1** 管理コンソールでの **JVM** 状態の表示*

1.

- *スタンドアロンサーバーインスタンスの **JVM** 状態 の表示*

  *画面の上部にある **Runtime** タブを選択します。**Status** メニューを展開し、**Platform** メニューを展開します。**JVM** を選択します。*

- *管理対象ドメインの **JVM** 状態の表示*

  *画面の上部にある **Runtime** タブを選択します。**Server Status** メニューを展開し、**Platform** メニューを展開します。**JVM** を選択します。*

2.

*管理対象ドメインはサーバーグループのすべてのサーバーインスタンスを表示できますが、サーバーメニューから選択すると一度に **1** つのサーバーのみを表示できます。サーバーグループ内の他のサーバーの状態を表示するには、画面の左側にある **Server** の変更 をクリックして、グループに表示されるホストおよびサーバーから選択します。必要なサーバーまたはホストを選択し、**JVM** の詳細が変更されます。**Close** をクリックして終了します。*

結果

*サーバーインスタンスに対する **JVM** 設定の状態が表示されます。*

バグの報告

14.1.3. JVM の設定

<jvm></jvm> タグは、**<option value="VALUE"/>** タグを使用して **JVM** 設定にパラメーターを追加するために使用できる **<jvm-options></jvm-options>** の使用をサポートします。

例**14.3 <jvm-options>** の使用

```
<jvm name="default">
  <heap size="1303m" max-size="1303m"/>
  <permgen max-size="256m"/>
  <jvm-options>
    <option value="-XX:+UseCompressedOops"/>
  </jvm-options>
</jvm>
```

CLI を使用した **JVM** の設定

CLI を使用して **JVM** を設定するには、以下の構文を使用します。

```
# cd /server-group=main-server-group/jvm=default

# :add-jvm-option(jvm-option="-XX:+UseCompressedOops")
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {
            "outcome" => "success",
            "response-headers" => {
                "operation-requires-restart" => true,
                "process-state" => "restart-required"
            }
        }},
        "server-two" => {"response" => {
            "outcome" => "success",
            "response-headers" => {
                "operation-requires-restart" => true,
                "process-state" => "restart-required"
            }
        }}
    }}}}
}

# :read-resource

# Expected Result:

[domain@localhost:9999 jvm=default] :read-resource
  {
    "outcome" => "success",
    "result" => {
    "agent-lib" => undefined,
    "agent-path" => undefined,
```

```
    "env-classpath-ignored" => undefined,
    "environment-variables" => undefined,
    "heap-size" => "1303m",
    "java-agent" => undefined,
    "java-home" => undefined,
    "jvm-options" => ["-XX:+UseCompressedOops"],
    "max-heap-size" => "1303m",
    "max-permgen-size" => "256m",
    "permgen-size" => undefined,
    "stack-size" => undefined,
    "type" => undefined
  }
}
```

jvm-options エントリーの削除

jvm-options エントリーを削除するには、次の構文を使用します。

```
# cd /server-group=main-server-group/jvm=default

# :remove-jvm-option(jvm-option="-XX:+UseCompressedOops")

# Expected Result:

[domain@localhost:9999 jvm=default] :remove-jvm-option(jvm-option="-XX:+UseCompressedOops")
{
    "outcome" => "success",
    "result" => undefined,
    "server-groups" => {"main-server-group" => {"host" => {"master" => {
        "server-one" => {"response" => {
            "outcome" => "success",
            "response-headers" => {
                "operation-requires-restart" => true,
                "process-state" => "restart-required"
            }
        }},
        "server-two" => {"response" => {
            "outcome" => "success",
            "response-headers" => {
                "operation-requires-restart" => true,
                "process-state" => "restart-required"
            }
        }}
    }}}}
}
```

32 ビットまたは 64 ビットアーキテクチャーの指定

Hewlett-Packard HP-UX および *Solaris* などの一部の環境では、*32* ビットまたは *64* ビット *JVM* で実行されるかどうかを指定するために *-d32* または *-d64* スイッチが使用されます。指定がない場合はデフォルトで *32* ビットが使用されます。

手順14.2 スタンドアロンサーバーでの 64 ビットアーキテクチャーの指定

1. *EAP_HOME/bin/standalone.conf* ファイルを開きます。

2. 以下の行を追加して、*-d64* オプションを *JAVA_OPTS* に追加します。

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

手順14.3 管理対象ドメインでの 64 ビットアーキテクチャーの指定

管理対象ドメインで実行している場合は、サーバーインスタンスの他にホストとプロセスコントローラーの *64* ビット環境を指定できます。

1. *64* ビット *JVM* で実行するよう、ホストおよびプロセスコントローラーを設定します。

   a. *EAP_HOME/bin/domain.conf* ファイルを開きます。

   b. 以下の行を追加して、*-d64* オプションを *JAVA_OPTS* に追加します。*PROCESS_CONTROLLER_JAVA_OPTS* および *HOST_CONTROLLER_JAVA_OPTS* が設定される前に挿入されていることを確認します。

```
JAVA_OPTS="$JAVA_OPTS -d64"
```

例14.4 domain.confの JVM オプション

```
#
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
  JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxPermSize=256m -
```

```
Djava.net.preferIPv4Stack=true"
 JAVA_OPTS="$JAVA_OPTS -
Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -
Djava.awt.headless=true"
 JAVA_OPTS="$JAVA_OPTS -Djboss.modules.policy-permissions=true"
 JAVA_OPTS="$JAVA_OPTS -d64"
else
 echo "JAVA_OPTS already set in environment; overriding default settings
with values: $JAVA_OPTS"
fi
```

2. **64** ビット **JVM** で実行するよう、 サーバーインスタンスを設定します。

a. 適切な **host.xml** 設定ファイルに **-d64** を **JVM** オプションとして追加します。

例**14.5 host.xml**の **JVM** オプション

```
<jvms>
 <jvm name="default">
  <heap size="64m" max-size="256m"/>
  <permgen size="256m" max-size="256m"/>
  <jvm-options>
   <option value="-server"/>
   <option value="-d64"/>
  </jvm-options>
 </jvm>
</jvms>
```

注記

管理 **CLI** では以下のコマンドで、適切なホストに **-d64** オプションを追加できます。

```
/host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-d64")
```

バグの報告

14.1.4. Java Security Manager について

Java Security Manager は、**Java Virtual Machine (JVM)** サンドボックスの外部境界を管理するクラスで、**JVM** 内で実行されるコードが **JVM** 外のリソースと対話する方法を制御します。**Java Security Manager** を有効にすると、**Java API** はさまざまな潜在的に危険な操作を実行する前に、セキュリティーマネージャーで承認を確認します。**Java Security Manager** はセキュリティーポリシーを使用して、特定のアクションが許可または拒否されるかどうかを判断します。

バグの報告

14.1.5. Java セキュリティーポリシー

Java Security ポリシーは、さまざまなクラスのコードに対する定義されたパーミッションのセットです。**Java Security Manager** は、アプリケーションによって要求されたアクションをセキュリティーポリシーと比較します。ポリシーがアクションを許可した場合、**Security Manager** はそのアクションの実行を許可します。ポリシーによりアクションが許可されない場合、セキュリティーマネージャーはそのアクションを拒否します。セキュリティーポリシーは、コードの場所、コードの署名、またはサブジェクトのプリンシパルに基づいてパーミッションを定義できます。

セキュリティーポリシー付与エントリーは、以下の設定要素で構成されます。

CodeBase

コードの起点となる **URL** の場所（ホストとドメイン情報を除く）。このパラメーターは任意です。

SignedBy

コードの署名に秘密鍵が使用された署名者を参照するキーストアで使用されるエイリアス。これは、単一の値またはコンマ区切りの値のリストになります。このパラメーターは任意です。省略した場合には、署名の有無に関わらず、**Java Security Manager** には影響がありません。

Principal

実行中のスレッドのプリンシパルセットに存在する必要がある **principal_type/principal_name** ペアの一覧。**Principals** エントリーは任意です。これを省略すると、実行中のスレッドのプリンシパルが **Java Security Manager** には影響しません。

Permissions

パーミッションは、コードに付与されるアクセスです。多くのパーミッションは、**Java Enterprise Edition 6(Java EE 6)**仕様の一部として提供されます。

バグの報告

14.1.6. Java セキュリティーポリシーの作成

ほとんどの **JDK** および **JRE** ディストリビューションに **policytool** と呼ばれるアプリケーションが含まれており、**Java** セキュリティーポリシーを作成および編集することを目的としています。**policytool** に関する詳細情報は、にリンクされてい **http://docs.oracle.com/javase/6/docs/technotes/tools/** ます。テキストエディターを使用してセキュリティーポリシーを作成することもできます。

手順**14.4** 新しい **Java Security Manager** ポリシーの設定

1. **policytool** を起動します。

   policytool ツールを以下のいずれかの方法で起動します。

   - **Red Hat Enterprise Linux**

     GUI またはコマンドプロンプトで、**/usr/bin/policytool** を実行します。

   - **Microsoft Windows Server**

     Start メニューまたは **Java** インストールの **bin** から **policytool.exe** を実行します。ロケーションは異なる場合があります。

2. ポリシーを作成します。

   ポリシーを作成するには、**Add Policy Entry** を選択します。必要なパラメーターを追加してから、完了 をクリック し ます。

注記

VFS を使用して、**JBoss EAP** にデプロイされたアプリケーションのパスを指定します。**Linux** では、パスは **vfs:/content/application.war** になります。**Microsoft Windows** では、**vfs:/${user.dir}/content/application.war** です。

以下に例を示します。

```
grant codeBase "vfs:/content/application.war/-" {
permission java.util.PropertyPermission "*", "read";
};
```

3. 既存のポリシーを編集します。

既存のポリシーの一覧からポリシーを選択し、**Edit Policy Entry** ボタンを選択します。必要に応じてパラメーターを編集します。

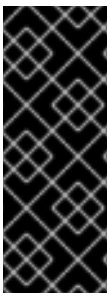
4. 既存のポリシーを削除します。

既存のポリシーの一覧からポリシーを選択し、**Remove Policy Entry** ボタンを選択します。

バグの報告

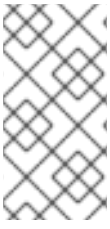
14.1.7. Java Security Manager 内での JBoss EAP 6 の実行

JBoss EAP 6.4 以降、**Java Security Manager(JSM)**内で **JBoss EAP 6** を実行すると、**secmgr** オプションを使用します。



重要

-Djava.security.manager Java システムプロパティーの直接使用ができなくなりました。以前の **JBoss EAP 6** で **Java Security Manager** を有効にするために使用されていたこの方法は、**JBoss EAP** 起動スクリプトのフォールバックメカニズムとしてのみサポートされています。



注記

JBoss EAP 6.4 以降では、カスタムセキュリティーマネージャーは使用できません。

以下の手順では、指定のセキュリティーポリシーを使用して **Java Security Manager** 内で実行するように **JBoss EAP 6** インスタンスを設定する手順を説明します。

前提条件

- この手順を実行する前に、**Java Development Kit(JDK)**または **Java SE Runtime Environment(JRE)**に含まれる **policytool** アプリケーションを使用してセキュリティーポリシーを記述する必要があります。テキストエディターを使用してセキュリティーポリシーを作成することもできます。

  パーミッションが必要なユーザーデプロイメントには、セキュリティーポリシーが必要です。この手順では、ポリシーが **EAP_HOME/bin/server.policy** にあることを前提としています。

- 設定ファイルを編集する前に、ドメインまたはスタンドアロンサーバーを完全に停止する必要があります。

管理対象ドメインで **JBoss EAP 6** を使用している場合は、ドメインの各物理ホストまたはインスタンスで以下の手順を実行する必要があります。

手順**14.5 JBoss EAP 6** の **Java Security Manager** の設定

1. 設定ファイルを開く

   設定ファイルを開いて編集します。編集が必要な設定ファイルは、管理対象ドメインまたはスタンドアロンサーバーとオペレーティングシステムのどちらを使用するかによって異なります。

   - 管理対象ドメイン
     - **For Linux: EAP_HOME/bin/domain.conf**
     - **Windows** の場合： **EAP_HOME\bin\domain.conf.bat**
   - スタンドアロンサーバー

- ***For Linux: EAP_HOME/bin/standalone.conf***
- ***Windows の場合： EAP_HOME\bin\standalone.conf.bat***

2. ***Java Security Manager** の有効化*

   *以下のいずれかの方法を使用して **Java Security Manager** を有効にします。*

   - ***JBoss EAP 6** のサーバー起動スクリプトで **-secmgr** オプションを使用します。*
   - *設定ファイルの **secmgr="true"** 行のコメントを解除します。*

     - ***Linux** の場合:*

       ```
       # Uncomment this to run with a security manager enabled
       SECMGR="true"
       ```

     - ***Windows** の場合:*

       ```
       rem # Uncomment this to run with a security manager enabled
       set "SECMGR=true"
       ```

3. ***Java** セキュリティーポリシーの指定*

-Djava.security.policy を使用して、セキュリティーポリシーの場所を指定できます。これは、改行なしで 1 行のみに置く必要があります。-Djava.security.policy を設定する場合は == を使用すると、セキュリティーマネージャーは指定されたポリシーファイル のみ を使用するように指定します。= を使用すると、セキュリティーマネージャーは JAVA_HOME/jre/lib/security/java.security の policy.url セクションに設定されたポリシー と 組み合わせ て指定されたポリシーを使用するように指定します。

関連する JBoss EAP 6 設定ファイルでセキュリティーポリシー Java オプションを追加します。管理対象ドメインを使用している場合は、PROCESS_CONTROLLER_JAVA_OPTS および HOST_CONTROLLER_JAVA_OPTS が設定される前に挿入されていることを確認します。

- *Linux の場合:*

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.policy==EAP_HOME/bin/server.policy
-Djboss.home.dir=EAP_HOME"
```

- *Windows の場合:*

```
set "JAVA_OPTS=%JAVA_OPTS% -
Djava.security.policy==EAP_HOME\bin\server.policy -
Djboss.home.dir=EAP_HOME"
```

4. *ドメインまたはサーバーの起動*

   ドメインまたはサーバーを通常どおり起動します。

バグの報告

14.1.8. IBM JDK および Java Security Manager

IBM JDK のバージョンによっては、JBoss EAP セキュリティーポリシーと適切に動作しないデフォルトのポリシープロバイダーを使用します。IBM JDK を使用して Java Security Manager を有効にして JBoss EAP をホストする際に問題が発生した場合は、JRE 設定を変更して標準のポリシープロバイダーを使用する必要があります。

IBM JDK の **JRE** 設定を変更するには、**JAVA_HOME/jre/lib/security/java.security** ファイルを編集し、**policy.provider** の値を **sun.security.provider.PolicyFile** に設定します。

```
policy.provider=sun.security.provider.PolicyFile
```

バグの報告

14.1.9. Security Manager ポリシーのデバッグ

デバッグ情報を有効にして、セキュリティーポリシー関連の問題のトラブルシューティングに役立ちます。**java.security.debug** オプションは、報告されたセキュリティー関連情報のレベルを設定します。**java -Djava.security.debug=help** コマンドは、デバッグオプションの全範囲でヘルプ出力を生成します。デバッグレベルを **all** に設定すると、原因が完全に不明ですが、一般的に使用する場合に、情報が過剰に生成されるセキュリティー関連の障害をトラブルシューティングするのに便利です。適切な一般的なデフォルトは **access:failure** です。

手順**14.6** 一般的なデバッグの有効化

- この手順を実行すると、セキュリティー関連デバッグ情報の一般的な機密レベルを有効にすることができます。

  次の行をサーバー設定ファイルに追加します。

  - **JBoss EAP 6** インスタンスが管理対象ドメインで実行されている場合は、**Linux** の場合は **bin/domain.conf** ファイル、**Windows** の場合は **bin\domain.conf.bat** ファイルに追加されます。

  - **JBoss EAP 6** インスタンスがスタンドアロンサーバーとして実行されている場合、行は **Linux** の場合は **bin/standalone.conf** ファイル、**Windows** の場合は **bin\standalone.conf.bat** ファイルに追加されます。

Linux

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.debug=access:failure"
```

Windows

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.debug=access:failure"
```

結果

セキュリティー関連デバッグ情報の一般的なレベルが有効になります。

バグの報告

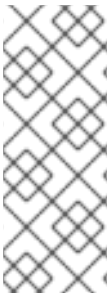
第15章 WEB サブシステム

15.1. WEB サブシステムの設定

Web サブシステムは管理コンソールまたは管理 CLI を使用して設定できます。

管理コンソールで、画面上部の Configuration タブをクリックして Subsystems メニューを展開し、Web メニューを展開します。Servlet/HTTP メニュー項目をクリックし、JSP、connector、および Virtual Servers 設定も Global を設定します。Web Services メニュー項目をクリックし、Web サービスサブシステムを設定します。

管理 CLI では、すべての Web サブシステムパラメーターは標準のコマンド形式を使用して設定されます。



注記

mod_cluster コンポーネントは、プロファイルが管理対象ドメインで ha または full-ha であるか、または standalone-ha または standalone-full-ha プロファイルでスタンドアロンサーバーを起動する場合にのみ利用できます。mod_cluster 設定の詳細は、「mod_cluster サブシステムの設定」 を参照してください。

バグの報告

15.2. HTTP セッションタイムアウトの設定

HTTP セッションタイムアウトは、指定された期間内にアクティビティーがないため、HTTP セッションが無効とみなされる期間を定義します。HTTP セッションタイムアウトを変更するには、影響を受けるすべての JBoss EAP インスタンスを再起動する必要があります。これが完了するまで、元の HTTP セッションタイムアウト値が適用されます。

HTTP セッションタイムアウトは、複数の場所で設定できます。以下に優先順に記載します。

- application - アプリケーションの web.xml 設定ファイルで定義されます。詳細は、『『開発ガイド』』『の「アプリケーションごとの HTTP タイムアウトの設定』」を参照してください。
-

server: default-session-timeout 属性で指定します。この設定は **JBoss EAP 6.4** からのみ利用できます。

- デフォルトは **30** 分です。

手順**15.1** 管理コンソールを使用した **HTTP** セッションタイムアウトの設定

1. **Configuration** タブをクリックし、**Subsystems,Web** に移動し、**Servlet/HTTP** メニュー項目をクリックします。
2. **Servlet/HTTP Configuration** パネルの **Global** タブをクリックします。
3. **Edit** オプションをクリックします。
4. **Default session timeout** の新しい値を入力します。
5. **Save** ボタンをクリックします。
6. **JBoss EAP** サーバーをリロードします。

手順**15.2** 管理 **CLI** を使用した **HTTP** セッションタイムアウトの設定

注記

管理対象ドメインのコマンドに、プレフィックス */host=HOST_NAME* を追加します。

1. 必要な **HTTP** セッションタイムアウト値を指定します。

```
/subsystem=web:write-attribute(name=default-session-timeout, value=timeout)
```

2. **JBoss EAP** サーバーをリロードします。

```
reload
```

バグの報告

15.3. サーブレット/HTTP 設定

手順15.3 サーブレット/HTTP 設定

1. 管理コンソールで、画面上部の **Configuration** タブをクリックして **Subsystems** メニューを展開し、**Web** メニューを展開します。

2. **Servlet/HTTP** メニュー項目をクリックします。

3. コンポーネントの名前をクリックし、**Edit** をクリックします。

4. **Advanced** ボタンをクリックして高度なオプションを表示します。

以下は、高度なサーブレット/HTTP 設定オプションです。

管理 **CLI** コマンドがプロファイルに適用される場合は、プレフィックス **/profile=PROFILE** を追加します。

例15.1 このインスタンスの名前の設定

```
[domain@localhost:9999 /] /profile=full-ha/subsystem=web:write-attribute(name=instance-id,value=worker1)
```

グローバル設定オプション

デフォルトのセッションタイムアウト

JBoss EAP 6.4 から利用できます。Web コンテナーのデフォルトセッションタイムアウト。

管理 CLI 属性： default-session-timeout

デフォルトの仮想サーバー

Web コンテナーのデフォルト仮想サーバー。

管理 CLI 属性： default-virtual-server

インスタンス ID

ロードバランシングのシナリオでセッションアフィニティーを有効にするのに使用される ID。識別子はクラスター内のすべての JBoss EAP サーバーで一意である必要があり、生成されたセッション識別子に追加されます。これにより、フロントエンドプロキシーは特定のセッションを同じ JBoss EAP インスタンスに転送できます。インスタンス ID はデフォルトとして設定されていません。

管理 CLI 属性： instance-id

ネイティブ

ネイティブ初期化リスナーを Web コンテナーに追加します。

管理 CLI 属性： ネイティブ値： true または falseデフォルト： true

***JSP** 設定オプション*

チェック間隔

バックグラウンドスレッドを使用して JSP 更新を確認する間隔（秒単位）。デフォルト： 0 （無効）

管理 CLI 属性： check-interval

開発

true の場合、開発モード が有効化されます。これにより、より詳細なデバッグ情報が生成されます。デフォルト**: false**。

管理 **CLI** 属性： 開発

無効

true の場合は、**Java ServerPages(JSP)**コンテナーが無効になります。**JSP** を使用しない場合は便利です。デフォルトは **false** です。

管理 **CLI** 属性： 無効

ソースの追加表示

true の場合、ランタイムエラーが発生すると **JSP** ソースフラグメントが表示されます。デフォルト： **true**

管理 **CLI** 属性： **display-source-fragment**

SMAP のダンプ

true の場合、**JSR 045 SMAP** データはファイルに書き込まれます。デフォルト**: false**

管理 **CLI** 属性： **dump-smap**

文字列を **Char** アレイとして生成

true の場合、文字列定数は **char** 配列として生成されます。デフォルト**: false**

管理 **CLI** 属性： **generate-strings-as-char-arrays**

Bean の無効なクラス属性の使用時のエラー

true の場合、**useBean** で不適切なクラスが使用される場合にエラーの出力を有効にします。デフォルト**: false**

管理 **CLI** 属性： **error-on-use-bean-invalid-class-attribute**

Java Encoding

Java ソースに使用されるエンコーディングを指定します。デフォルト： **UTF8**

管理 **CLI** 属性： **java-encoding**

生成されているままにする

true の場合、生成されたサーブレットを保持します。デフォルト： **true**

管理 **CLI** 属性： **keep-generated**

マップされたファイル

true の場合、デバッグを容易にするために、入力行ごとに **1** つの印刷ステートメントを使用して静的コンテンツが生成されます。デフォルト： **true**

管理 **CLI** 属性： **true**

変更テスト間隔

更新の **2** つのテストの最小時間（秒単位）。デフォルト： **4**

管理 **CLI** 属性： **modification-test-interval**

Fail での再コンパイル

true の場合、リクエストごとに失敗した **JSP** コンパイルが再試行されます。デフォルト**:** **false**。

管理 **CLI** 属性： **recompile-on-fail**

スクラッチディレクトリー

別のワークディレクトリーの場所を指定します。

*管理 **CLI** 属性： **scratch-dir***

SMAP

true** の場合、**JSR 045 SMAP** が有効になります。デフォルト**: true

*管理 **CLI** 属性： **smap***

ソース仮想マシン

*ソースファイルと互換性のある **Java Development Kit** バージョン。デフォルト： **1.5***

*管理 **CLI** 属性： **source-vm***

タグプール

true** の場合、タグプーリングは有効になります。デフォルト**: true

*管理 **CLI** 属性： **tag-pooling***

ターゲット仮想マシン

*クラスファイルと互換性のある **Java Development Kit** バージョン。デフォルト： **1.5***

*管理 **CLI** 属性： **target-vm***

トリミング領域

*生成されたサーブレットから一部の領域をトリミングします。デフォルト**: false**。*

*管理 **CLI** 属性： **trim-spaces***

x Powered By

true** の場合、**JSP** エンジンは **x-powered-by HTTP** ヘッダーを使用してアドバタイズされます。デフォルト： **true

*管理 **CLI** 属性： **x-powered-by***

以下のコマンドのいずれかを使用して、これらのコネクターのいずれかの設定を表示します。

```
[standalone@localhost:9999 /] /subsystem=web/connector=http:read-resource-description
```

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:read-resource-description
```

*コネクターを設定するには、**Connectors** タブを選択し、**Add** をクリックします。コネクターを削除するには、そのエントリーを選択し、**Remove** をクリックします。コネクターを編集するには、エントリーを選択し、**Edit** をクリックします。*

*管理 **CLI** を使用して新しいコネクターを作成する場合、以下のコマンドなどでオプションがすべて設定されます。*

*例**15.2** 新しいコネクターの作成*

```
[domain@localhost:9999 /] /profile=full-ha/subsystem=web/connector=ajp/:add(socket-
binding=ajp,scheme=http,protocol=AJP/1.3,secure=false,name=ajp,max-post-
size=2097152,enabled=true,enable-lookups=false,redirect-port=8433,max-save-post-size=4096)
```

デフォルトのコネクター属性

bytes Sent

コネクターによって送信されるバイト数。

*管理 **CLI** 属性： **bytesSent***

プロキシーポート

リダイレクトの送信時に使用されるポート。

*管理 **CLI** 属性： **proxy-port***

Secure

*コネクターによって送受信されたコンテンツがユーザーパースペクティブからセキュア化されるかどうかを示します。デフォルト**: false**。*

*管理 **CLI** 属性： **secure***

仮想サーバー

このコネクターからアクセス可能な仮想サーバーの一覧。デフォルト： すべての仮想サーバーを許可します。

*管理 **CLI** 属性： **virtual-server***

エラー数

コネクターによるリクエストの処理時に発生するエラーの数。

*管理 **CLI** 属性： **errorCount***

最大時間

要求の処理に費やされた最大時間。

*管理 **CLI** 属性： **maxTime***

ソケットバインディング

コネクターがバインドされる名前付きソケットバインディング。ソケットバインディングは、ソケット名とネットワークポート間のマッピングです。ソケットバインディングは各スタンドアロンサーバーまたは管理対象ドメインのソケットバインディンググループで設定されます。ソケットバインディンググループはサーバーグループに適用されます。

*管理 **CLI** 属性： **socket-binding***

Scheme

***Web** コネクタースキーム（**HTTP** や **HTTPS** など）。*

*管理 **CLI** 属性： **scheme***

名前

コネクターの一意の名前。

*管理 **CLI** 属性： **name***

*最大 **POST** サイズ*

*コンテナーで解析できる **POST** リクエストの最大サイズ（バイト単位）。デフォルト： **2097152***

*管理 **CLI** 属性： **max-post-size***

リクエスト数

コネクターによって処理されるリクエストの数。

*管理 **CLI** 属性： **requestCount***

プロキシー名

*リダイレクトの送信時に使用されるホスト名。デフォルト： **null***

*管理 **CLI** 属性： **proxy-name***

有効

*コネクターが起動時に開始されるかどうかを定義します。デフォルト**: true***

*管理 CLI 属性： **enabled***

プロトコル

***AJP** または **HTTP** のいずれかを使用する **Web** コネクタープロトコル。それぞれのプロトコルについて **API** を指定し、使用する実装を判別するためにサーバーに残すか、または完全修飾クラス名を指定できます。*

*管理 CLI 属性： **protocol***

HTTP

API** オプション： **HTTP/1.1** または **HTTP/1.0

*クライアントが **HTTP/1.1** に対応していない場合、コネクターは最初の応答で **HTTP/1.1** を返し、**HTTP/1.0** にフォールバックします。*

完全修飾コネクター名 オプション：

- ***jio: org.apache.coyote.http11.Http11Protocol***
- ***NIO2: org.apache.coyote.http11.Http11NioProtocol***
- ***APR: org.apache.coyote.http11.Http11AprProtocol***

AJP

API** オプション： **AJP/1.3

完全修飾コネクター名 オプション：

- ***jio: org.apache.coyote.ajp.AjpProtocol***
- ***APR: org.apache.coyote.ajp.AjpAprProtocol***

注記

***APR** では、ネイティブコンポーネントパッケージをインストールして有効にしておく必要があります。詳細な手順は、**JBoss EAP** 『**インストールガイド**』 』を参照してください。*

ルックアップの有効化

***Servlet API** の **DNS** ルックアップを有効にします。*

*管理 **CLI** 属性： **enable-lookups***

*エグゼキューター **(Executor)***

このコネクターの処理スレッドに使用されるエグゼキューターの名前。定義されていない場合は、デフォルトで内部プールを使用します。

*管理 **CLI** 属性： **executor***

処理時間

コネクターによって使用される処理時間（ミリ秒単位）。

*管理 **CLI** 属性： **processingTime***

プロキシーバインディング

*リダイレクトの送信時に使用されるホストおよびポートを定義するソケットバインディング。デフォルト： **null***

*管理 **CLI** 属性： **proxy-binding***

Redirect Port

*セキュアなコネクターへリダイレクトするためのポート。デフォルト： **443***

同時に設定されている場合に ***redirect-binding*** *に遅延します。*

管理 ***CLI*** *属性：* ***redirect-port***

受信バイト数

*コネクター（****POST*** *データ）によって受信されるバイト数。*

管理 ***CLI*** *属性：* ***bytesReceived***

リダイレクトバインディング

リダイレクトバインディングは動作においてポートをリダイレクトするのと似ていますが、ポート番号の代わりにソケットバインディング名 ***を*** *指定する必要があります。****redirect-binding*** *オプションでは、事前定義されたソケットバインディング（****https****、****AJP*** *など）をリダイレクト用に特定のポートに使用できるようにするため、設定の柔軟性が高くなります。****redirect-port*** *オプションと同じ結果が提供されます。*

同時に設定されている場合は、リダイレクトポート よりも優先されます。

管理 ***CLI*** *属性：* ***redirect-binding***

最大接続

パフォーマンスを最適化してコネクターが処理できる同時接続の最大数。デフォルト値は、使用されるコネクターによって異なります。

管理 ***CLI*** *属性：* ***max-connections***

最大保存後サイズ

特定の認証スキーム時に保存される ***POST*** *リクエストの最大サイズ（バイト単位）。デフォルトは* ***4096*** *です。*

管理 ***CLI*** *属性：* ***max-save-post-size***

仮想サーバーを設定するには、**Virtual Servers** タブをクリックします。**Add** ボタンを使用して新しい仮想サーバーを追加します。仮想サーバーを編集または削除するには、そのエントリーを選択し、編集 または 削除 ボタンをクリックします。

管理 **CLI** を使用して新しい仮想サーバーを追加する場合、以下のコマンドのようにすべての必須オプションが一度に設定されます。

例**15.3** 新しい仮想サーバーの追加

```
/profile=full-ha/subsystem=web/virtual-server=default-host/:add(enable-welcome-root=true,default-web-module=ROOT.war,alias=["localhost","example.com"],name=default-host)
```

仮想サーバーオプション

アクセスログ

アクセスログ情報をログに記録する方法を記述する要素。

管理 **CLI** 属性： **access-log**

- **access-log** 属性を追加するには、以下の管理 **CLI** コマンドを入力します。

```
/subsystem=web/virtual-server=default-host/configuration=access-log:add()
```

- **access-log** 属性には複数の子属性があります。これらの子とそれらの設定オプションを一覧表示するには、以下の管理 **CLI** コマンドを入力します。

```
/subsystem=web/virtual-server=default-host/configuration=access-log:read-resource-description()
```

pattern

リクエストおよび応答からのさまざまな情報フィールドを特定するフォーマット

*レイアウト、または標準形式を選択する **common** または **combined** という単語を指定します。**pattern** 属性の値は、フォーマットトークンで構成されます。指定されない場合は、デフォルトの **common** が使用されます。*

rotate

***ouput** をローテーションする必要がある場合にバルブに指示します。指定されていない場合は、デフォルトの **true** が使用されます。*

プレフィックス

*ログファイルの名前を作成するために使用されるプレフィックスを定義します。指定されていない場合は、デフォルトの **access_log** が使用されます。*

拡張

***AccessLogValve** の代わりに **ExtendedAccessLogValve** を使用します。指定されていない場合は、デフォルトの **false** が使用されます。*

resolve-hosts

*ホスト名を解決するかどうかをバルブに指示します。指定されていない場合は、デフォルトの **false** が使用されます。**resolve-host** のコネクターの ルックアップ が有効になっていない限り、このオプションは動作しません。これは、**resolve-host** を **true** に設定すると有効にできます。*

Alias

*この仮想サーバーでサポートされるホスト名の一覧。管理コンソールでは、**1** 行あたり **1** つのホスト名を使用します。*

*管理 **CLI** 属性： エイリアス*

*デフォルトの **Web** モジュール*

Web アプリケーションがこの仮想サーバーのルートノードにデプロイされるモジュールで、HTTP リクエストにディレクトリーが指定されていない場合に表示されます。デフォルト：ROOT.war

管理 CLI 属性： default-web-module

Welcome ルートの有効化

この要素は、バンドルされた welcome ディレクトリーがルート Web コンテキストとして使用されるかどうかを定義します。デフォルト: false

管理 CLI 属性： enable-welcome-root

名前

表示目的の仮想サーバーの一意な名前。

管理 CLI 属性： name

rewrite

リライトバルブが仮想ホストに対応するリクエストと実行する必要があるものを記述する要素。rewrite は、処理前にリクエストが書き換えられる方法を記述します。RewriteValve を virtual-server で定義される仮想ホストに追加します。

flag

RewriteRule は、1 つ以上のフラグでその動作を変更できます。フラグはルールの最後に角括弧に含まれており、複数のフラグはコンマで区切ります。

pattern

pattern は、リクエストの URL に適用される perl と互換性のある正規表現です。

置換

書き換えルールの置き換えは、パターンが一致した元の **URL** に置換（または置換）される文字列です。

管理 **CLI** 属性： **rewrite**

SSO

この仮想サーバーの **SSO** 設定。デフォルト（**http** の場合）**: true**

管理 **CLI** 属性： **sso**

- **SSO** 設定を追加するには、以下の管理 **CLI** コマンドを入力します。

```
/subsystem=web/virtual-server=default-host/configuration=sso:add()
```

- **SSO** 属性には複数の子属性があります。これらの子とそれらの設定オプションを一覧表示するには、以下の管理 **CLI** コマンドを入力します。

```
/subsystem=web/virtual-server=default-host/configuration=sso:read-resource-
description()
```

http-only

cookie http-only フラグが設定されます。

cache-container

指定されたクラスター化されたキャッシュコンテナーを使用してクラスター化された **SSO** を有効にします。

reauthenticate

SSO の使用時にレルムと再認証を有効にします。

domain

使用されるクッキードメイン。

cache-name

キャッシュコンテナーで使用するキャッシュの名前。

バグの報告

15.4. デフォルトの WELCOME WEB アプリケーションの置き換え

JBoss EAP 6 には Welcome アプリケーションが含まれており、これはポート 8080 でサーバーの URL を開く際に表示されます。以下の手順に従って、このアプリケーションを独自の Web アプリケーションに置き換えることができます。

手順15.4 デフォルトの Welcome Web アプリケーションを独自の Web アプリケーションに置き換える

1. *Welcome アプリケーションを無効にします。*

   管理 CLI スクリプト EAP_HOME/bin/jboss-cli.sh を使用して以下のコマンドを実行します。プロファイルを変更して別の管理対象ドメインプロファイルを変更するか、スタンドアロンサーバーのコマンドの /profile=default 部分を削除しなければならない場合があります。

   ```
   /profile=default/subsystem=web/virtual-server=default-host:write-attribute(name=enable-welcome-root,value=false)
   ```

2. *ルートコンテキストを使用するよう Web アプリケーションを設定します。*

   ルートコンテキスト(/)を URL アドレスとして使用するように Web アプリケーションを設定するには、META-INF/ または WEB-INF/ ディレクトリーにある jboss-web.xml を変更しま

す。**< context-root>** ディレクティブを、以下のようなディレクティブに置き換えます。

```
<jboss-web>
    <context-root>/</context-root>
</jboss-web>
```

3. アプリケーションをデプロイします。

    最初の手順で変更したサーバーグループまたはサーバーにアプリケーションをデプロイします。アプリケーションが **http://SERVER_URL:PORT/** で利用できるようになりました。

バグの報告

15.5. JBOSSWEB のシステムプロパティー

ここでは、デフォルトの **JBossWeb** 動作の変更に使用できるシステムプロパティーを示します。**system-properties** は **JBoss Enterprise Web** アプリケーション設定で設定できます。再起動して、**Web** サブシステムに適用する必要があります。

以下の例は、**JBossWeb** で **system-properties** を変更する方法の例になります。

```
standalone@localhost:9999 /] ./system-
property=org.apache.catalina.JSESSIONID:add(value="MYID")
{"outcome" => "success"}
standalone@localhost:9999 /] shutdown
Communication error: Channel closed
Closed connection to localhost:9999
```

プロパティーによっては、**reload** コマンドを使用して再起動できます。

以下は、**reload** コマンドを使用して再起動する例です。

```
[standalone@localhost:9999 /] reload
{
    "outcome" => "success",
    "response-headers" => {
        "operation-requires-reload" => true,
        "process-state" => "reload-required"
    }
}
```

表**15.1** サーブレットコンテナーおよびコネクター

属性	説明
jvmRoute	***jvmRoute*** 属性のデフォルト値を提供します。standalone-ha.xml などの設定を使用して ha read を使用する場合に、自動生成された値を上書きしません。 **reload** をサポートします。
org.apache.tomcat.util.buf.StringCache.byte.enabled	true の場合、String キャッシュは ByteChunk に対して有効になります。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.tomcat.util.buf.StringCache.char.enabled	true の場合、String キャッシュは CharChunk に対して有効になります。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.tomcat.util.buf.StringCache.cacheSize	String キャッシュのサイズ。値の指定がない場合は、デフォルト値の 5000 が使用されます。
org.apache.tomcat.util.buf.StringCache.maxStringSize	キャッシュされる String の最大長。値の指定がない場合は、デフォルト値の 128 が使用されます。
org.apache.tomcat.util.http.FastHttpDateFormat.CACHE_SIZE	解析およびフォーマットされた日付値を使用するキャッシュのサイズ。値の指定がない場合は、デフォルト値の 1000 が使用されます。
org.apache.catalina.core.StandardService.DELAY_CONNECTOR_STARTUP	true の場合、コネクターは自動的に起動されません。これは埋め込みモードで便利です。
org.apache.catalina.connector.Request.SESSION_ID_CHECK	true の場合、Servlet コンテナーは、ID でセッションを作成する前に、指定されたセッション ID のコンテキストにセッションが存在することを確認します。
org.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER	true の場合、HTTP ヘッダー内でカスタムの HTTP ステータスメッセージが使用されます。XSS の脆弱性を防ぐために、このようなメッセージがメッセージに特に入力されている場合は ISO-8859-1 でエンコードされていることを確認する必要があります。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.tomcat.util.http.Parameters.MAX_COUNT	ポストボディーで解析できるパラメーターの最大量です。この値を超えると、IllegalStateException によって解析に失敗します。デフォルト値は 512 パラメーターです。
org.apache.tomcat.util.http.MimeHeaders.MAX_COUNT	HTTP リクエストで送信できるヘッダーの最大量。この値を超えると、IllegalStateException によって解析に失敗します。デフォルト値は 128 ヘッダーです。

属性	説明
org.apache.tomcat.util.net.MAX_THREADS	コネクターがリクエストの処理に使用するスレッドの最大数。デフォルト値は 32 x Runtime.getRuntime（）.availableProcessors（）です。（JIO コネクターの場合は 512 x Runtime.getRuntime（）.availableProcessors（））
org.apache.coyote.http11.Http11Protocol.MAX_HEADER_SIZE	HTTP ヘッダーのバイト単位の最大サイズ。この値を超えると、ArrayOutOfBoundsExceptions を使用して解析に失敗します。デフォルト値は 8192 バイトです。
org.apache.coyote.http11.Http11Protocol.COMPRESSION	HTTP コネクターでの簡単な圧縮の使用を許可します。デフォルト値は off で、on 値を使用して圧縮を有効にし、条件付きで有効にするか、常に強制的に有効にすることができます。
org.apache.coyote.http11.Http11Protocol.COMPRESSION_RESTRICTED_UA	圧縮されたコンテンツを受け取らないユーザーエージェント regexps。デフォルト値は空です。
org.apache.coyote.http11.Http11Protocol.COMPRESSION_MIME_TYPES	圧縮可能なコンテンツのコンテンツタイプ接頭辞。デフォルト値は text/html,text/xml,text/plain です。
org.apache.coyote.http11.Http11Protocol.COMPRESSION_MIN_SIZE	圧縮されるコンテンツの最小サイズ。デフォルト値は 2048 バイトです。
org.apache.coyote.http11.DEFAULT_CONNECTION_TIMEOUT	デフォルトのソケットタイムアウト。デフォルト値は 60000 ミリ秒です。
org.jboss.as.web.deployment.DELETE_WORK_DIR_ONCONTEXTDESTROY	このプロパティーを使用して **.java** および **.class** ファイルを削除し、JSP ソースが再コンパイルされるようにします。デフォルト値は **false** です。keep alive のデフォルトのソケットタイムアウト。デフォルト値は -1 ミリ秒で、デフォルトのソケットタイムアウトを使用します。
org.apache.tomcat.util.buf.StringCache.trainThreshold	キャッシュをアクティベートする前に **toString（）** を呼び出す必要のある回数を指定します。デフォルト値は **100000** です。

表15.2 EL

属性	説明
org.apache.el.parser.COERCE_TO_ZERO	true の場合、式を数字 "" に変換し、null は仕様で必要なようにゼロに強制されます。値の指定がない場合は、デフォルト値の true が使用されます。

表15.3 JSP

属性	説明
org.apache.jasper.compiler.Generator.VAR_EXPRESSIONFACTORY	式言語の式ファクトリーに使用される変数の名前。値の指定がない場合は、デフォルト値の _el_expressionfactory が使用されます。
org.apache.jasper.compiler.Generator.VAR_INSTANCEMANAGER	インスタンスマネージャーファクトリーに使用する変数の名前。値の指定がない場合は、デフォルト値の _jsp_instancemanager が使用されます。
org.apache.jasper.compiler.Parser.STRICT_QUOTE_ESCAPING	false の場合、JSP 属性の引用符をエスケープする要件が緩和され、必要な引用符が欠落しているとエラーが発生しません。値の指定がない場合は、仕様に準拠するデフォルトの true が使用されます。
org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE	org.apache.jasper.Constants.DEFAULT_TAG_BUFFER_SIZE を超えるタグバッファーは破棄され、デフォルトサイズの新しいバッファーが作成されます。値の指定がない場合は、デフォルト値の 512 が使用されます。
org.apache.jasper.runtime.JspFactoryImpl.USE_POOL	true の場合、ThreadLocal PageContext プールが使用されます。値の指定がない場合は、デフォルト値の true が使用されます。
org.apache.jasper.runtime.JspFactoryImpl.POOL_SIZE	ThreadLocal PageContext のサイズ。値の指定がない場合は、デフォルト値の 8 が使用されます。
org.apache.jasper.Constants.JSP_SERVLET_BASE	JSP から生成されたサーブレットのベースクラス。値の指定がない場合は、デフォルト値の org.apache.jasper.runtime.HttpJspBase が使用されます。
org.apache.jasper.Constants.SERVICE_METHOD_NAME	ベースクラスによって呼び出されるサービスメソッドの名前。値の指定がない場合は、デフォルト値の _jspService が使用されます。
org.apache.jasper.Constants.SERVLET_CLASSPATH	JSP のクラスパスを提供する ServletContext 属性の名前。値の指定がない場合は、デフォルト値の org.apache.catalina.jsp_classpath が使用されます。
org.apache.jasper.Constants.JSP_FILE	サーブレット定義の <jsp-file> 要素の request 属性の名前。リクエストに存在する場合は、request.getServletPath（）によって返される値を上書きし、実行される JSP ページを選択します。値の指定がない場合は、デフォルト値の org.apache.catalina.jsp_file が使用されます。

属性	説明
org.apache.jasper.Constants.PRECOMPILE	JSP エンジンがサーブレットを事前生成し、呼び出しは行わないようにするクエリーパラメーターの名前。値の指定がない場合は、デフォルト値の org.apache.catalina.jsp_precompile が使用されます。
org.apache.jasper.Constants.JSP_PACKAGE_NAME	コンパイルされた jsp ページのデフォルトのパッケージ名。値の指定がない場合は、デフォルト値の org.apache.jsp が使用されます。
org.apache.jasper.Constants.TAG_FILE_PACKAGE_NAME	タグファイルから生成されたタグハンドラーのデフォルトのパッケージ名。値の指定がない場合は、デフォルト値の org.apache.jsp.tag が使用されます。
org.apache.jasper.Constants.TEMP_VARIABLE_NAME_PREFIX	生成された一時的な変数名に使用する接頭辞。値の指定がない場合は、デフォルト値の _jspx_temp が使用されます。
org.apache.jasper.Constants.USE_INSTANCE_MANAGER_FOR_TAGS	true の場合、インスタンスマネージャーはタグハンドラーインスタンスの取得に使用されます。値の指定がない場合は true が使用されます。
org.apache.jasper.Constants.INJECT_TAGS	true の場合、タグに指定されたアノテーションは処理およびインジェクトされます。簡単なタグを使用する場合やタグプーリングが無効になっている場合はパフォーマンスに影響することがあります。値の指定がない場合は false が使用されます。

*表**15.4** セキュリティー*

属性	説明
org.apache.catalina.connector.RECYCLE_FACADES	true の場合や、セキュリティーマネージャーが使用中の場合は、リクエストごとに新しいファサードオブジェクトが作成されます。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH	true の場合、「\」文字はパス区切り文字として使用できます。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH	true の場合、「%2F」と「%5C」はパス区切り文字として使用できます。値の指定がない場合は、デフォルト値の false が使用されます。

*表**15.5** 仕様*

属性	説明
org.apache.catalina.STRICT_SERVLET_COMPLIANCE	値の指定がない場合は true が使用されます。true の場合、以下のアクションが発生します。 ● アプリケーションディスパッチャーに渡されるラップされた要求または応答オブジェクトは、元の要求または応答がラップされるようにチェックされます。(SRV.8.2 / SRV.14.2.5.1) ● 文字エンコーディングが指定されていない場合に Response.getWriter（）を呼び出すと、後続の Response.getCharacterEncoding（）が ISO-8859-1 を返し、Content-Type 応答ヘッダーに charset=ISO-8859-1 コンポーネントが含まれます。(SRV.15.2.22.1) ● セッションに関連するすべてのリクエストにより、リクエストが明示的にセッションにアクセスしたかどうかに関わらず、セッションの最終アクセス時間が更新されます。(SRV.7.6)
org.apache.catalina.core.StandardWrapperValve.SERVLET_STATS	true を指定した場合または org.apache.catalina.STRICT_SERVLET_COMPLIANCE が true の場合、ラッパーは各サーブレットの JSR-77 統計を収集します。値の指定がない場合は、デフォルト値の false が使用されます。
org.apache.catalina.session.StandardSession.ACTIVITY_CHECK	true の場合や、org.apache.catalina.STRICT_SERVLET_COMPLIANCE が true の場合、Tomcat は各セッションのアクティブなリクエストの数を追跡します。セッションが有効であるかを判断するとき、アクティブなセッションが 1 つ以上あるセッションは常に有効であると見なされます。値の指定がない場合は、デフォルト値の false が使用されます。

バグの報告

15.6. HTTP のみのセッション管理クッキー

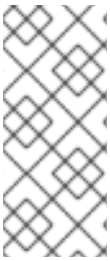
*セッション管理クッキーの **http-only** 属性は、**HTTP** 以外の **API**（**JavaScript** など）からのアクセスを制限することで、セキュリティーの脆弱性のリスクを軽減します。この制限は、クロスサイトスクリプティング攻撃によるセッションクッキーの脅威を軽減するのに役立ちます。クライアント側では、**JavaScript** またはその他のスクリプトメソッドを使用して **Cookie** にアクセスできません。これはセッション管理クッキーにのみ適用され、他のブラウザークッキーには適用されません。デフォルトでは、**http-only** 属性は有効になっています。*

まだ行われていない場合は、**http-only** 属性を使用するには、**Web** サブシステムの仮想サーバーに **SSO** を追加する必要があります。

例**15.4** 仮想サーバーへの **SSO** の追加

以下の管理 **CLI** コマンドを入力して、**Web** サブシステムの仮想サーバーに **SSO** を追加します。

```
/subsystem=web/virtual-server=default-host/configuration=sso:add
```



注記

このコマンドで「**JBAS014803: Duplicate resource**」が失敗すると、**SSO** がすでに仮想サーバー設定に追加されていることを意味します。このエラーは無視して、続行することができます。

注記

JSESSIONID および **JSESSIONIDSSO** はセッション追跡クッキーです。デフォルトでは、**http** 専用 であるため、スクリプトでアクセスすることはできません。

例**15.5 http-only** 属性の検証

以下の管理 **CLI** コマンドを入力して、**http-only** 属性の値を確認します。

```
/subsystem=web/virtual-server=default-host/configuration=sso:read-resource
{
    "outcome" => "success",
    "result" => {
        "cache-container" => undefined,
        "cache-name" => undefined,
        "domain" => undefined,
        "http-only" => true,
        "reauthenticate" => undefined
    },
    "response-headers" => {"process-state" => "reload-required"}
}
```

例**15.6 http-only** 属性の有効化

*以下の管理 **CLI** コマンドを入力して **http-only** 属性を有効にします。*

```
/subsystem=web/virtual-server=default-host/configuration=sso:write-attribute(name=http-only,value=true)
```

バグの報告

第16章 WEB サービスサブシステム

16.1. WEB サービスオプションの設定

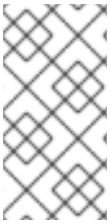
Web サービスオプションは、管理コンソールまたは管理 **CLI** を使用して設定できます。

管理コンソールで、画面上部の **Configuration** タブをクリックして **Subsystems** メニューを展開し、**Web** メニューを展開します。**Web Services** メニュー項目をクリックし、**Web** サービスサブシステムを設定します。オプションは以下の表で説明されています。

表16.1 Web サービス設定オプション

オプション	説明	CLI コマンド
Modify WSDL Address	WSDL アドレスをアプリケーションで変更できるかどうか。デフォルトは **true** です。	/profile=full-ha/subsystem=webservices/:write-attribute(name=modify-wsdl-address,value=true)
WSDL Host	JAX-WS Web サービスの WSDL コントラクトには、エンドポイントの場所を示す <soap:address> 要素が含まれます。<soap:address> の値が有効な URL の場合、**modify-wsdl-address** が **true** に設定されていない限り上書きされません。<soap:address> の値が有効な URL ではない場合、**wsdl-host** の値と **wsdl-port** または **wsdl-secure-port** のいずれかを使用して上書きされます。**wsdl-host** が **jbossws.undefined.host** に設定された場合、リクエスターのホストアドレスは <soap-address> の書き換え時に使用されます。デフォルトは **${jboss.bind.address:127.0.0.1}** で、JBoss EAP 6 の起動時にバインドアドレスが指定されていない場合は **127.0.0.1** を使用します。	/profile=full-ha/subsystem=webservices/:write-attribute(name=wsdl-host,value=127.0.0.1)
WSDL Port	SOAP アドレスを書き換えるために使用されるセキュアではないポート。これが設定されない場合（デフォルト）、ポートはインストールされたコネクターのリストを問い合わせることにより識別されます。	/profile=full-ha/subsystem=webservices/:write-attribute(name=wsdl-port,value=80)

オプション	説明	CLI コマンド
WSDL Secure Port	SOAP アドレスを書き換えるために使用されるセキュアなポート。これが設定されない場合（デフォルト）、ポートはインストールされたコネクターのリストを問い合わせることにより識別されます。	/profile=full-ha/subsystem=webservices/:write-attribute(name=wsdl-secure-port,value=443)



注記

*プロファイルを変更して別の管理対象ドメインを変更するか、スタンドアロンサーバーのコマンドの /**profile=full-ha** 部分を削除する必要があることがあります。*

***Web** サービスサブシステム*

***Apache CXF** でのロギングを有効にするには、**standalone/domain.xml** ファイルに以下のシステムプロパティーを設定します。*

```
<system-properties>
<property name="org.apache.cxf.logging.enabled" value="true"/>
</system-properties>
```

バグの報告

16.2. ハンドラーおよびハンドラーチェーンの概要

*各エンドポイント設定は **PRE** および **POST** ハンドラーチェーンに関連付けることができます。各ハンドラーチェーンには **JAXWS** 準拠のハンドラーを含めることができます。送信メッセージの場合、**PRE** ハンドラーチェーンハンドラーは、**@HandlerChain** アノテーションなどの標準の **JAXWS** を使用してエンドポイントにアタッチされたハンドラーの前に実行されます。**POST** ハンドラーチェーンハンドラーは、通常のエンドポイントハンドラーの後に実行されます。受信メッセージには、逆が適用されます。**JAX-WS** は **XML** ベースの **Web** サービスの標準 **API** で、で **http://jcp.org/en/jsr/detail?id=224** 説明されています。*

*ハンドラーチェーンには、チェーンが起動するプロトコルを設定する **protocol-bindings** 属性も含まれます。*

***JAXWS** ハンドラーは、ハンドラーチェーン内の子要素 ハンドラー です。ハンドラーは、ハンド*

*ラー クラス の完全修飾クラス名である **class** 属性を取ります。エンドポイントがデプロイされると、参照デプロイメントごとにそのクラスのインスタンスが作成されます。デプロイメントクラスローダーまたはモジュール **org.jboss.as.webservices.server.integration** のクラスローダーのいずれかがハンドラークラスをロードできる必要があります。*

*手順**16.1 CLI** で **handler-chains** およびハンドラーを追加する方法*

1. ***JBoss EAP CLI** の起動*

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

2. ***JBoss CLI** よりハンドラーチェーンおよびハンドラーを追加します。*

*例**16.1** ハンドラーチェーンの追加*

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers:add(protocol-bindings="##SOAP11_HTTP")
```

*例**16.2** ハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=foo-hander:add(class="org.jboss.ws.common.invocation.RecordingServerHandler")
```

*例**16.3** ハンドラーの追加*

```
[standalone@localhost:9999 /] /subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=bar-handler:add(class="com.arjuna.webservices11.wsarj.handler.InstanceIdentifierInHandler")
```

3. *サーバーをリロードします。*

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

4. ***handler-chain** およびハンドラーが正しく追加されたことを確認します。*

*例**16.4 handler-chain** の読み取り*

```
[standalone@localhost:9999 /] /profile=default/subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers:read-resource
```

例**16.5** ハンドラーの読み取り

```
[standalone@localhost:9999 /] /profile=default/subsystem=webservices/endpoint-config=Standard-Endpoint-Config/post-handler-chain=my-handlers/handler=bar-handler:read-resource
```

上記コマンドで使用されるオプションは、ハンドラーの追加または変更の必要に応じて変更できます。

JBoss EAP で利用可能なハンドラーについては、『**API** ドキュメント』 **javadoc** を参照してください。

https://access.redhat.com/documentation/ja-JP/JBoss_Enterprise_Application_Platform/6.4/html/API_Documentation/files/javadoc/javax/xml/ws/handler/Handler.html

ハンドラー、**handler-chains**、エンドポイント、および関連する問題に関する詳細は、『**JBoss EAP**『開発ガイド』』 を参照してください。 **https://access.redhat.com/documentation/ja-jp/red_hat_jboss_enterprise_application_platform/?version=6.4**

バグの報告

第17章 HTTP クラスタリングおよび負荷分散

17.1. HTTP サーバー名の規則

以下の表は、**HTTPD** 関連のトピックを説明する際に使用される命名規則および場所規則の概要を示しています。

表17.1 オペレーティングシステムが提供する Apache HTTP Server

Operating System	HTTPD_CONF.D	HTTPD_CONF	HTTPD_MODULES
Red Hat Enterprise Linux (httpd)	**/etc/httpd/conf.d**	**/etc/httpd/conf**	**/etc/httpd/modules**
HPUX(Web Server Suite)	以下の注記を参照してください。	**/opt/hpws/apache/conf**	**/opt/hpws/apache/modules**

注記

Apache HTTP Server 用の **HP-UX** の **Web** サーバースイートには **conf.d** はありません。ただし、以下のように作成できます。

手順17.1 title

1. **Create /opt/hpws/apache/conf.d.**
2. **Include conf.d/*.conf** を **httpd.conf** ファイルに追加します。

表17.2 JBoss Enterprise Web Server

Operating System	HTTPD_CONF.D	HTTPD_CONF	HTTPD_MODULES	ディストリビューション
Red Hat Enterprise Linux	**/*EWS_HOME*/httpd/conf.d**	**/*EWS_HOME*/httpd/conf**	**/*EWS_HOME*/httpd/modules**	zip
Red Hat Enterprise Linux	**/etc/httpd/conf.d**	**/etc/httpd/conf**	**/usr/lib/httpd/modules**	rpm
Solaris	**/*EWS_HOME*/etc/httpd/conf.d**	**/*EWS_HOME*/etc/httpd/conf**	**/*EWS_HOME*/lib/httpd/modules**	zip

Operating System	*HTTPD_CONF.D*	*HTTPD_CONF*	*HTTPD_MODULES*	ディストリビューション
Windows	***/EWS_HOME*/etc/httpd/conf.d**	***/EWS_HOME*/etc/httpd/conf**	***/EWS_HOME*/lib/httpd/modules**	zip

注記

パスの差異：

- ***64** ビットアーキテクチャーを使用している場合は、上記の表のファイルパスを修正して **lib64/** ディレクトリーを使用します。*
- ***Red Hat Enterprise Linux 7** インストールを使用している場合は、**httpd** ディレクトリーの名前が **httpd22** になります。*

バグの報告

17.2. はじめに

17.2.1. 高可用性および負荷分散クラスター

*クラスタリングとは、サーバーなどの複数のリソースを単一のエンティティーとして使用することを指します。主なクラスタリングには、負荷分散**(LB)**と高可用性**(HA)** の **2** つのタイプがあります。**LB** クラスターでは、すべてのリソースが同時に実行され、管理層はそれらの間で負荷を分散します。*

__HA__ クラスタリングでは、__1__ つのリソースが実行され、もう __1__ つは最初のリソースが利用できなくなった場合にステップで使用できます。__HA__ クラスタリングの目的は、ハードウェア、ソフトウェア、またはネットワークの停止による影響を軽減することです。

__JBoss EAP 6__ は、クラスタリングを複数のレベルでサポートします。高可用性を実現できるランタイムおよびアプリケーションのコンポーネントには、以下が含まれます。

- *アプリケーションサーバーのインスタンス*
- *内部 __JBoss Web__ サーバー、__Apache HTTP__ サーバー、__Microsoft IIS__、または __Oracle__*

iPlanet Web Server と併用される Web アプリケーション

- *ステートフル、ステートレス、およびエンティティー Enterprise JavaBean (EJB)*
- *シングルサインオン (SSO) メカニズム*
- *分散キャッシュ*
- *HTTP セッション*
- *JMS サービスおよびメッセージ駆動型 Bean (MDB)*

JBoss EAP 6 では、jgroups と modcluster の 2 つのサブシステムによってクラスタリングが使用できるようになります。ha プロファイルおよび full-ha プロファイルではこれらのシステムが有効になります。JBoss EAP 6 では、これらのサービスは必要に応じて起動およびシャットダウンしますが、配布可能 として設定されたアプリケーションがサーバーにデプロイされた場合のみ起動します。

JBoss EAP 6 では、Infinispan はキャッシュプロバイダーとして提供されます。Infinispan は JBoss EAP 6 のクラスタリングとレプリケーションキャッシュを管理します。

バグの報告

17.2.2. 高可用性が有益なコンポーネント

高可用性 (HA) は、JBoss EAP 6 の幅広いカテゴリーに分類されます。

コンテナ

JBoss EAP 6 の複数のインスタンス（スタンドアロンサーバーとして実行）またはサーバーグループのメンバー（管理対象ドメインの一部として実行）を高可用性に設定することが可能です。つまり、1 つのインスタンスまたはメンバーがクラスターから停止または非表示になると、そのワークロードがピアに移動します。負荷負荷も管理できるため、負荷分散機能も提供することができます。そのため、サーバーまたはサーバーグループのより優れたリソースを持つリソースを持つサーバーやサーバーグループは、負荷が高いときに追加で追加することも、負荷が高いときに追加で追加したりできます。

Web サーバー

互換性のある負荷分散メカニズムの 1 つを使用して、Web サーバー自体を HA 用にクラスター化することができます。最も柔軟性の高い mod_cluster コネクターは、JBoss EAP 6 コンテナーと密接に統合されています。その他の選択肢は、Apache mod_jk または mod_proxy コネクター、ISAPI コネクター、NSAPI コネクターなどです。

アプリケーション

Java Enterprise Edition 6(Java EE 6)仕様により、デプロイされたアプリケーションを高可用性にすることが可能です。ステートレスまたはステートフルセッション EJB をクラスター化できるため、作業に関与するノードがなくなると、別のノードが引き継ぎ、ステートフルセッション Bean の場合は状態が保持されます。

バグの報告

17.2.3. HTTP コネクターの概要

JBoss EAP 6 には、Apache Web Server、Microsoft IIS、Oracle iPlanet などの外部 Web サーバーに構築された負荷分散および高可用性メカニズムを使用する機能があります。JBoss EAP 6 はコネクターを使用して外部 Web サーバーと通信します。これらのコネクターは JBoss EAP 6 の web サブシステム内に設定されます。

Web サーバーには、HTTP リクエストが JBoss EAP 6 ノードにルーティングされる方法を制御するソフトウェアモジュールが含まれます。これらのモジュールの挙動や設定方法はモジュールごとに異なります。モジュールは、複数の JBoss EAP 6 ノード全体でワークロードを分散したり、障害発生時にワークロードを別のサーバーに移動したりするために設定されます。これらの機能は、ロードバランシング および 高可用性(HA) と呼ばれます。

JBoss EAP 6 は複数のコネクターをサポートします。使用中の Web サーバーや必要な機能に応じてコネクターを選択します。

以下の表は、JBoss EAP 6 と互換性のある HTTP コネクターの違いを示しています。HTTP コネクターのサポートされる構成に関する最新情報は、を参照してください
https://access.redhat.com/articles/111663。

表17.3 HTTP コネクター機能および制約

コネクター	Web サーバー	サポート対象オペレーティングシステム	サポート対象プロトコル	デプロイメント状態への適合	スティッキーセッションのサポート
mod_cluster	JBoss Enterprise Web Server の httpd、オペレーティングシステム (Red Hat Enterprise Linux、Hewlett-Packard HP-UX) によって提供される httpd	Red Hat Enterprise Linux、Microsoft Windows Server、Oracle Solaris、Hewlett-Packard HP-UX	HTTP、HTTPS、AJP	必要。アプリケーションのデプロイメントとアンデプロイメントを検出し、アプリケーションがそのサーバーにデプロイされたかどうかに基づいて、サーバーにクライアント要求を送信するかどうかを動的に決定します。	はい
mod_jk	JBoss Enterprise Web Server の httpd、オペレーティングシステム (Red Hat Enterprise Linux、Hewlett-Packard HP-UX) によって提供される httpd	Red Hat Enterprise Linux、Microsoft Windows Server、Oracle Solaris、Hewlett-Packard HP-UX	AJP	不可。 アプリケーションの状態に関係なく、コンテナーが利用可能な限り、クライアント要求をコンテナーに送信します。	はい
mod_proxy	JBoss Enterprise Web Server の httpd	Red Hat Enterprise Linux、Microsoft Windows Server、Oracle Solaris	HTTP、HTTPS、AJP	不可。 アプリケーションの状態に関係なく、コンテナーが利用可能な限り、クライアント要求をコンテナーに送信します。	はい
ISAPI コネクター	Microsoft IIS	Microsoft Windows Server	AJP	不可。 アプリケーションの状態に関係なく、コンテナーが利用可能な限り、クライアント要求をコンテナーに送信します。	はい

コネクター	Web サーバー	サポート対象オペレーティングシステム	サポート対象プロトコル	デプロイメント状態への適合	スティッキーセッションのサポート
NSAPI コネクター	Oracle iPlanet Web Server	Oracle Solaris	AJP	不可。 アプリケーションの状態に関係なく、コンテナーが利用可能な限り、クライアント要求をコンテナーに送信します。	はい

*各 **HTTP** コネクターの詳細について*

- *「**mod_cluster HTTP** コネクター」*
- *「**Apache mod_jk HTTP** コネクター」*
- *「**Apache mod_proxy HTTP** コネクター」*
- *「インターネットサーバー **API(ISAPI)**について」*
- *「**Netscape Server API(NSAPI)**」*

***JBoss EAP 6** でサポートされる構成は次のとおり **https://access.redhat.com/articles/111663** です。*

バグの報告

17.2.4. ノードのタイプ

***HTTP** コネクターノード*

*ワーカーノードは、ノード と呼ばれることもあります。これは、プロキシーとして機能する **1** つ以上のクライアント向け **HTTP** サーバーからの要求を許可する **JBoss EAP 6** サーバーインスタンスで*

す。**JBoss EAP 6** は、**Apache HTTP Server**、**Microsoft IIS**、または **Oracle iPlanet Web Server**（旧名 **Netscape Web Server**）からの **HTTP**、**HTTPS** および **AJP** リクエストを許可できます。

JBoss EAP 6 でサポートされる **HTTP** コネクターの概要と設定方法は、「**HTTP** コネクターの概要」 を参照してください。

クラスターノード

クラスターノード はワーカーノードの特殊な設定です。このようなクラスターは、負荷分散、高可用性、またはその両方です。負荷分散クラスターでは、状況固有の測定により、中央マネージャーはノードに負荷を均等に分散します。高可用性クラスターでは、一部のノードがアクティブに作業を行い、アクティブなノードの **1** つがクラスターから離れる場合にステップを待機しています。

例**17.1 mod_cluster** が設定された **Apache HTTP Server**

たとえば、**mod_cluster** が設定された **Apache HTTP Server** がクライアントのリクエストへの負荷分散プロキシーとして動作する設定が可能です。これらの要求は、現在の負荷に従って特定のクラスターノードに転送されます。スティッキーセッションが無効の場合、受信リクエストはすべて現在の負荷に従って分散されます。スティッキーセッションが有効な場合（デフォルト）、すでに作成されたセッション内の後続のリクエストは、セッションが作成されたワーカーノードにルーティングされます。これらのクラスターノードのいずれかが停止またはオーバーロードすると、負荷分散プロキシーとして動作する **mod_cluster** がある **Apache HTTP Server** はクライアントのリクエストを別のクラスターノードに転送します。クラスター内のセッションレプリケーションにより、クライアントのデータは失われません。

同様に、クラスター化されていないワーカーノードの設定が簡単に行えます。ワーカーノードが停止している場合、その近傍に **former** セッションデータがない場合には、クライアントは **HTTP 200** 以外のエラーコードを取得しません。

スティッキーセッションの設定は、ワーカーノードがクラスター化されているかどうかに拘らず、負荷分散プロキシーは重要ではありません。

バグの報告

17.3. コネクター設定

17.3.1. JBoss EAP 6 にて HTTP コネクターのスレッドプールを定義

概要

JBoss EAP 6 のスレッドプールは、Executor モデルを使用して異なるコンポーネント間で共有できます。これらのプールは、異なる(HTTP)コネクターによる共有だけでなく、Executor モデルをサポートする JBoss EAP 6 内の他のコンポーネントも共有できます。現在の Web パフォーマンス要件に一致するように HTTP コネクタースレッドプールを取得することは複雑です。現在のスレッドプールと現在の Web 負荷要求と予想される Web 負荷要求を詳細に監視する必要があります。このタスクでは、Executor モデルを使用して HTTP コネクターのスレッドプールを設定する方法を説明します。管理 CLI と XML 設定ファイルの両方を使用してこれを設定する方法を説明します。

注記

JBoss EAP をドメインモードで実行している場合は、この手順で接頭辞 /profile=PROFILE_NAME をすべての管理 CLI コマンドに追加します。

手順17.2 HTTP コネクターのスレッドプールの設定

1. *スレッドファクトリーの定義*

   設定ファイルを開きます（スタンドアロンサーバーに対して変更する場合はstandalone.xml、ドメインベースの設定に対して変更する場合は domain.xml ）。このファイルは EAP_HOME/standalone/configuration または EAP_HOME/domain/configuration フォルダーにあります。

   次のサブシステムエントリーを追加します。値はサーバーの要件に合わせて変更します。

   ```
   <subsystem xmlns="urn:jboss:domain:threads:1.1">
       <thread-factory name="http-connector-factory" thread-name-pattern="HTTP-%t"
   priority="9" group-name="uq-thread-pool"/>
   </subsystem>
   ```

   管理 CLI を使用してこのタスクを実行する場合は、CLI コマンドプロンプトで次のコマンドを実行します。

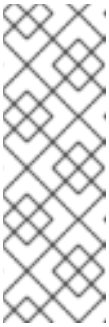
   ```
   [standalone@localhost:9999 /] ./subsystem=threads/thread-factory=http-connector-
   factory:add(thread-name-pattern="HTTP-%t", priority="9", group-name="uq-thread-
   pool")
   ```

2. *エクゼキューターの作成*

   6 つの組み込みエクゼキュータークラスの 1 つを使用して、このファクトリーのエクゼキューターとして動作します。6 つのエクゼキューターは以下のとおりです。

   - *unbounded-queue-thread-pool: このタイプのスレッドプールは常にタスクを許可します。実行されているスレッドの最大数を下回ると、新しいスレッドが起動し、送信さ*

*れたタスクが実行されます。それ以外の場合、タスクはスレッドが利用可能になるとバインドされていない **FIFO** キューに置かれます。*



注記

***Executors.singleThreadExecutor()** が提供する単一スレッドエグゼキュータータイプは、スレッド制限が **1** つに制限されているバインドされていないキューエグゼキューターです。このタイプのエクゼキューターは **unbounded-queue-thread-pool-executor** 要素を使用してデプロイされます。*

- ***bounded-queue-thread-pool:** このタイプのエクゼキューターは固定長のキューと、**core** サイズと **maximum** サイズという **2** つのプールサイズを維持します。タスクが受け入れられると、実行中のプールスレッドの数が **core** サイズ未満の場合は、タスクの実行に新しいスレッドが開始されます。スペースがキューに残ると、タスクはキューに配置されます。実行中のプールスレッドの数が **maximum** サイズ未満の場合は、新しいスレッドが開始されてタスクを実行します。エグゼキューターでブロックが有効になっていると、呼び出しスレッドは、キューで領域が利用可能になるまでブロックします。ハンドオフエクゼキューターが設定されている場合、タスクは **Handoff** エグゼキューターに委任されます。そうでない場合は、タスクは拒否されます。*

- ***blocking-bounded-queue-thread-pool:** スレッドの送信タスクがブロックされるバインドされたキューを持つスレッドプールエクゼキューター。このようなスレッドプールにはコアおよび最大サイズがあり、指定されたキューの長さがあります。タスクが送信されると、実行中のスレッド数がコアサイズ未満の場合、新しいスレッドが作成されます。そうでないと、キューに余裕がある場合はタスクはキューに置かれます。実行中スレッドの数が最大サイズ未満の場合は、新しいスレッドが作成されます。そうでないと、呼び出し元は、その部屋がキューで利用可能になるまでブロックされます。*

- ***queueless-thread-pool:** 場合によっては、別のスレッドでタスクを実行するには単純なスレッドプールが必要であり、キューを干渉しないタスクを完了するため、スレッドを再利用する必要があります。このタイプのプールは、タスクを受け入れ、他の実行中のタスクが完了するまで実行を遅らせるのではなく、タスクは常に承認直後に開始するなど、長時間実行されるタスクを処理するのに適しています。このタイプのエクゼキューターは **queueless-thread-pool-executor** 要素を使用して宣言されます。*

- ***blocking-queueless-thread-pool:** スレッドの送信タスクがブロックされるキューのないスレッドプールエクゼキューター。タスクが送信されると、実行中のスレッド数が最大サイズ未満の場合は、新しいスレッドが作成されます。それ以外の場合は、呼び出し元は、別のスレッドがタスクを完了し、新しいタスクを許可するまでブロックします。*

- ***scheduled-thread-pool:*** *これは、**java.util.concurrent.ScheduledThreadPoolExecutor** クラスに基づいて、特定の時間および時間間隔でタスクを実行する特別なタイプのエクゼキューターです。このタイプのエクゼキューターは **scheduled-thread-pool-executor** 要素で設定されます。*

*この例では、**unbounded-queue-thread-pool** を使用してエクゼキューターとして機能します。サーバーのニーズに合わせて **max-threads** および **keepalive-time** パラメーターの値を変更します。*

```
<unbounded-queue-thread-pool name="uq-thread-pool">
  <thread-factory name="http-connector-factory" />
  <max-threads count="10" />
  <keepalive-time time="30" unit="seconds" />
</unbounded-queue-thread-pool>
```

*管理 **CLI** を使用する場合は、以下を行います。*

```
[standalone@localhost:9999 /] ./subsystem=threads/unbounded-queue-thread-
pool=uq-thread-pool:add(thread-factory="http-connector-factory", keepalive-time=
{time=30, unit="seconds"}, max-threads=30)
```

3. ***HTTP Web** コネクターがこのスレッドプールを使用するようにする*

*同じ設定ファイルで、**Web** サブシステム下にある **HTTP** コネクター要素を見つけ、前の手順で定義したスレッドプールを使用するよう編集します*

```
<connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"
executor="uq-thread-pool" />
```

*管理 **CLI** を使用する場合は、以下を実行します。*

```
[standalone@localhost:9999 /] ./subsystem=web/connector=http:write-
attribute(name=executor, value="uq-thread-pool")
```

4. *サーバーの再起動*

*サーバー（スタンドアロンまたはドメイン）を再起動して、変更を有効にします。以下の管理 **CLI** コマンドを使用して、上記の手順からの変更が実施されたかどうかを確認します。*

```
[standalone@localhost:9999 /] ./subsystem=threads:read-resource(recursive=true)
```

```
{
    "outcome" => "success",
    "result" => {
        "blocking-bounded-queue-thread-pool" => undefined,
        "blocking-queueless-thread-pool" => undefined,
        "bounded-queue-thread-pool" => undefined,
        "queueless-thread-pool" => undefined,
        "scheduled-thread-pool" => undefined,
        "thread-factory" => {"http-connector-factory" => {
            "group-name" => "uq-thread-pool",
            "name" => "http-connector-factory",
            "priority" => 9,
            "thread-name-pattern" => "HTTP-%t"
        }},
        "unbounded-queue-thread-pool" => {"uq-thread-pool" => {
            "keepalive-time" => {
                "time" => 30L,
                "unit" => "SECONDS"
            },
            "max-threads" => 30,
            "name" => "uq-thread-pool",
            "thread-factory" => "http-connector-factory"
        }}
    }
}
[standalone@localhost:9999 /] ./subsystem=web/connector=http:read-
resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "configuration" => undefined,
        "enable-lookups" => false,
        "enabled" => true,
        "executor" => "uq-thread-pool",
        "max-connections" => undefined,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "name" => "http",
        "protocol" => "HTTP/1.1",
        "proxy-name" => undefined,
        "proxy-port" => undefined,
        "redirect-port" => 443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "http",
        "ssl" => undefined,
        "virtual-server" => undefined
    }
}
```

結果

スレッドファクトリーとエクゼキューターが正常に作成され、このスレッドプールを使用するよう ***HTTP*** *コネクターが編集されます。*

バグの報告

17.4. WEB サーバーの設定

17.4.1. スタンドアロン Apache HTTP Server

JBoss EAP 6 は、認定されたバージョンの **Red Hat Enterprise Linux 6** に含まれる **Apache HTTP** サーバーでテストされ、サポートされています。**Apache HTTP** サーバーは、**Microsoft Windows Server** などの他のオペレーティングシステムにも利用可能です。しかし、**Apache HTTP** サーバーは **Apache Foundation** によって作成された個別の製品であるため、お客様が使用した **Apache HTTP** サーバーのバージョンが **JBoss EAP** と互換性があることを確認することは容易ではありませんでした。

スタンドアロン **Apache HTTP** サーバーバンドルが **JBoss EAP 6** の個別ダウンロードとして利用できるようになりました。これにより、**Red Hat Enterprise Linux** 以外の環境や、**Apache HTTP** サーバーが設定されていて、**Web** アプリケーションに別のインスタンスを使用するシステムにおいて、インストールと設定が容易になります。この **Apache HTTP Server** は、インストールプラットフォームで利用可能な **JBoss EAP 6** ダウンロードの配下にある **Customer Service Portal** で個別ダウンロードとしてダウンロードできます。

バグの報告

17.4.2. HTTPD 変数規則

表17.4 ネイティブ

製品	HTTPD_CONF	HTTPD_MODULES
Red Hat Enterprise Linux	/etc/httpd/conf	/etc/httpd/modules
HPUX	/opt/hpws/apache/conf	/opt/hpws/apache/modules

表17.5 EWS

製品	HTTPD_CONF	HTTPD_MODULES
Red Hat Enterprise Linux	/HTTPD_HOME/EWS-ROOT/httpd/conf	/HTTPD_HOME/EWS-ROOT/httpd/modules

製品	HTTPD_CONF	HTTPD_MODULES
Solaris	/HTTPD_HOME/EWS-ROOT/etc/httpd/conf	/HTTPD_HOME/EWS-ROOT/lib/httpd/modules または /HTTPD_HOME/EWS-ROOT/lib64/httpd/modules
Windows	/HTTPD_HOME/EWS-ROOT/etc/httpd/conf	/HTTPD_HOME/EWS-ROOT/lib/httpd/modules または /HTTPD_HOME/EWS-ROOT/lib64/httpd/modules

バグの報告

17.4.3. Red Hat Enterprise Linux 5、6、7 への Apache HTTP Server のインストール(Zip)

前提条件

- ***root** または管理者権限。*
- *サポートされるバージョンの **Java** がインストールされている必要があります。*
- *以下のパッケージがインストールされている必要があります:*
  - ***krb5-workstation***
  - ***mod_auth_kerb** （**Kerberos** 機能に必要）*
  - ***elinks** （**apachectl** 機能に必要）*
  - ***apr-util-devel** （**Apache Portability Runtime(APR)**）*

- ***apr-util-ldap*** *（**Red Hat Enterprise Linux 7** のみ。**LDAP** 認証機能に必要です）*

Apache HTTP Server Zip *アーカイブには、複数の **Kerberos** モジュールへのシンボリックリンクが含まれます。そのため、**mod_auth_kerb** パッケージが前提条件となります。**Kerberos** 機能が必要ない場合は、**mod_auth_kerb** パッケージをインストールする必要はなく、関連するシンボリックリンクを削除できます(**EAP_HOME/httpd/modules/mod_auth_kerb.so**)。*

*手順**17.3 Apache HTTP Server** のインストール*

1. ***Red Hat*** *カスタマーポータル上でご使用のプラットフォームの **JBoss EAP** ダウンロードリストへ移動します。*

   *カスタマーポータル にログインし、**Software Downloads** ページに移動します。適切な **Product** および **Version** を選択します。*

2. *一覧から **Apache HTTP Server** バイナリーを選択します。*

   *オペレーティングシステムおよびアーキテクチャー用の **Apache HTTP Server** オプションを検索します。**Download** リンクをクリックします。**Apache HTTP Server** ディストリビューションを含む **Zip** ファイルがコンピューターにダウンロードされます。*

3. ***Apache HTTP Server*** *バイナリーを実行するシステムに **Zip** を展開します。*

   *任意のサーバーの **Zip** ファイルを一時的な場所に展開します。**Zip** ファイルには、**jboss-ews-version-number** フォルダーの下に **httpd** ディレクトリーが含まれます。**httpd** フォルダーをコピーし、**EAP_HOME** ディレクトリー内に配置します。*

   Apache HTTP Server *が **EAP_HOME/httpd/** ディレクトリーにあります。このディレクトリーは **HTTPD_HOME** と呼ばれます。*

4. *インストール後のスクリプトを実行して、**Apache** ユーザーおよびグループ の アカウントを作成します。*

   *端末エミュレーターで **EAP_HOME/httpd** ディレクトリーに移動し、**root** ユーザー権限で以下のコマンドを実行します。*

   ```
   ./.postinstall
   ```

   *次に、以下のコマンドを実行して、**apache** ユーザーがシステムにインストールされていることを確認します。*

   ```
   id apache
   ```

ユーザーが存在しない場合は、適切なユーザーグループと共に追加する必要があります。これを行うには、**root** ユーザー権限で実行します。

```
getent group apache >/dev/null || groupadd -g 48 -r apache
getent passwd apache >/dev/null || useradd -r -u 48 \
-g apache -s /sbin/nologin  -d HTTPD_HOME/httpd/www -c "Apache" apache
```

この作業が完了したら、**apache** ユーザーが **Apache HTTP Server** サービスを実行している場合、**HTTP** ディレクトリーの所有者を変更する必要があります。

```
chown -R apache:apache httpd
```

上記のコマンドが正常に行われたことを確認するには、**apache** ユーザーに **Apache HTTP Server** のインストールパスの実行権限があることを確認します。

```
ls -l
```

出力は以下のようになります。

```
drwxrwxr-- 11 apache apache 4096 Feb 14 06:52 httpd
```

5. **Apache HTTP Server** を設定します。

   Apache HTTP Server を起動する前に、組織のニーズに合わせて設定します。一般的なガイダンスは、**Apache Foundation** の **http://httpd.apache.org/** のドキュメントを参照してください。

6. **Apache HTTP Server** を起動します。

   以下のコマンドを使用して **Apache HTTP Server** を起動します。

   ```
   HTTPD_HOME/httpd/sbin/apachectl start
   ```

7. **Apache HTTP Server** を停止します。

   Apache HTTP Server を停止するには、以下のコマンドを実行します。

   ```
   HTTPD_HOME/httpd/sbin/apachectl stop
   ```

バグの報告

17.4.4. Red Hat Enterprise Linux (RHEL) 5、6、および 7 への Apache HTTP Server のインストール (RPM)

前提条件

- *ルートレベルのアクセス。*
- *最新バージョンの **elinks** パッケージがインストールされています（**apachectl** 機能に必要）。*
- ***Red Hat Enterprise Linux (RHEL)** チャンネルをサブスクライブする必要があります (**RHEL** チャンネルから **Apache HTTP Server** をインストールするため)。*
- ***jbappplatform-6-ARCH-server-VERS-rpm Red Hat Network(RHN)**チャンネルをサブスクライブする必要があります（**Apache HTTP Server** の **EAP** 固有のディストリビューションをインストールするため）。*

*以下の方法の **1** つを使用して **Apache HTTP Server** をインストールできます。*

- ***Red Hat Enterprise Linux (RHEL)** チャンネルからのインストール。**Apache HTTP Server** のインストールには **Red Hat Enterprise Linux (RHEL)** チャンネルの有効なサブスクリプションが必要です。*
- ***jbappplatform-6-ARCH-server-VERS-rpm** チャンネル（**JBoss EAP** 固有のディストリビューション）から以下を行います。**JBoss EAP** は **Apache HTTP Server** の独自のバージョンを配布します。**Apache HTTP Server** の **JBoss EAP** 固有のディストリビューションをインストールするには、**jbappplatform-6-ARCH-server-VERS-rpm** チャンネルへのアクティブなサブスクリプションが必要です。*

*手順**17.4 Red Hat Enterprise Linux 5 および 6 への Apache HTTP Server のインストールおよび設定 (RPM)***

1. ***httpd**のインストール*

   ***JBoss EAP** 固有のバージョンの **httpd** パッケージをインストールするには、以下のコマンドを実行します。*

   ```
   yum install httpd
   ```

- *Red Hat Enterprise Linux(RHEL)チャンネルから **httpd** を明示的にインストールするには、以下のコマンドを実行します。*

```
yum install httpd --disablerepo=jbappplatform-6-*
```

注記

*上記のコマンドのいずれかを実行して、システムに **httpd** パッケージをインストールする必要があります。*

2. *サービスブートの挙動設定*

*コマンドラインまたはサービス設定のグラフィカルツールから、システムの起動時に **httpd** サービスのサービス動作を定義できます。次のコマンドを実行して動作を定義します。*

```
chkconfig httpd on
```

サービス設定ツールを使用するには、以下のコマンドを実行し、表示されたウインドウでサービス設定を変更します。

```
system-config-services
```

3. ***Start httpd***

*以下のコマンドを使用して **httpd** を起動します。*

```
service httpd start
```

4. ***Stop httpd***

*以下のコマンドを使用して **httpd** を停止します。*

```
service httpd stop
```

手順17.5 Red Hat Enterprise Linux 7 への Apache HTTP Server のインストールおよび設定 (RPM)

1. **Install httpd22**

   JBoss EAP 固有のバージョンの **httpd22** パッケージをインストールするには、以下のコマンドを実行します。

   ```
   yum install httpd22
   ```

2. サービスブートの挙動設定

   以下のコマンドを実行して、システムの起動時に **httpd22** サービスを起動します。

   ```
   systemctl enable httpd22.service
   ```

3. **Start httpd22**

   以下のコマンドを使用して **httpd22** を起動します。

   ```
   systemctl start httpd22.service
   ```

4. **Stop httpd22**

   以下のコマンドを使用して **httpd22** を停止します。

   ```
   systemctl stop httpd22.service
   ```

バグの報告

17.4.5. Microsoft Windows Server 環境向け Apache HTTP Server サービスの管理

手順17.6 Microsoft Windows Server 環境用の Apache HTTP Server サービスのインストール

- このコマンドを使用して **Apache HTTP Server** サービスをインストールします。

```
cd /D "%EWS_HOME%\bin"
httpd -k install
```

このコマンドは、**Apache2.2** という名前の **Apache HTTP Server** サービスをインストールします。

サービスに別の名前（**ApacheBalancer** など）を指定するには、以下のコマンドを使用します。

```
cd /D "%EWS_HOME%\bin"
httpd -k install -n ApacheBalancer
```

手順**17.7 Microsoft Windows Server** 環境用の **Apache HTTP Server** サービスの開始

- サービスを起動するには、**httpd.exe** またはサービスマネージャーを使用します。

  httpd.exe の使用：

```
cd /D "%EWS_HOME%\bin"
httpd -k start -n Apache2.2
```

  サービスマネージャーの使用：

```
net start Apache2.2
```

手順**17.8 Microsoft Windows Server** 環境の **Apache HTTP Server** サービスを停止します。

- サービスを停止するには、**httpd.exe** またはサービスマネージャーを使用します。

  httpd.exe の使用：

```
cd /D "%EWS_HOME%\bin"
httpd -k stop -n Apache2.2
```

  サービスマネージャーの使用：

```
net stop Apache2.2
```

手順17.9 Microsoft Windows Server 環境用の Apache HTTP Server サービスのアンインストール

- サービスをアンインストールするには、名前で参照する必要があります。たとえば、サービス名 **ApacheBalancer** をアンインストールするには、以下のコマンドを使用します。

```
cd /D "%EWS_HOME%\bin"
httpd -k uninstall -n ApacheBalancer
```

バグの報告

17.4.6. Apache HTTP Server での mod_cluster 設定

概要

mod_cluster コネクターは **Apache HTTP Server** ベースのロードバランサーです。通信チャネルを使用して、リクエストを **Apache HTTP Server** からアプリケーションサーバーノードのセットの **1** つに転送します。以下の派生を設定して **mod_cluster** を設定できます。

注記

mod_cluster は **Apache HTTP Server** に転送される必要がある **URL** を自動的に設定するため、**ProxyPass** ディレクティブを使用する必要はありません。

表17.6 mod_cluster の派生

派生	説明	値

派生	説明	値
CreateBalancers	バランサーが Apache HTTP Server の VirtualHosts でどのように作成されるかを定義します。**ProxyPass /balancer://mycluster1/ のようなディレクティブを許可します。**	0: Apache HTTP Server に定義されるすべての VirtualHosts を作成します。 1: バランサーを作成してはいけない (バランサー名の定義に最低でも 1 つの ProxyPass または ProxyMatch が必要) 2: メインサーバーのみを作成する デフォルト: 2 **値 1 を使用する場合は、必ず ProxyPass ディレクティブにバランサーを設定するようにしてください。これは、デフォルト値は空の stickysession および nofailover=Off であり、MCMP CONFIG メッセージを介して受信された値は無視されます。**
UseAlias	エイリアスがサーバー名に対応することを確認します。	0: エイリアスを無視する 1: エイリアスを確認する デフォルト: 0
LBstatusRecalTime	負荷分散論理がノードの状態を再計算する間隔 (秒単位)。	デフォルト: 5 秒
WaitBeforeRemove	削除されたノードを httpd が記憶しなくなるまでの時間 (分単位）。	デフォルト: 10 秒
ProxyPassMatch/ProxyPass	ProxyPassMatch および ProxyPass は（バックエンド URL ではなく）**!** を使用する場合、パスのリバースプロキシーを防ぐ mod_proxy ディレクティブです。これは、Apache HTTP Server が静的なコンテンツに対応できるようにするために使用されます。以下に例を示します。 **ProxyPassMatch ^(/.*\.gif)$!** 上記の例では、Apache HTTP Server は直接 **.gif** ファイルに対応できます。	

mod_cluster *ロジックのホットスタンバイノードは、他のすべてのノードがダウンした場合にすべての要求がルーティングされる最後のノードです。これは* ***mod_proxy*** *のホットスタンバイ論理と似ています。*

ホットスタンバイノードを設定するには、**mod_cluster** サブシステムの **dynamic-load-provider** を、ファクターが **0** に設定されている **simple-load-provider** に置き換えます。

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.2">
   <mod-cluster-config advertise-socket="modcluster" connector="ajp">
-       <dynamic-load-provider>
-          <load-metric type="busyness"/>
-       </dynamic-load-provider>
+       <simple-load-provider factor="0"/>
   </mod-cluster-config>
</subsystem>
```

mod_cluster-manager コンソールでは、ノードが **OK** ステータスと **Load: 0** で表示されます。詳細は、『**JBoss Enterprise 『Application Platform『開発ガイド』』** の「**Apache mod_cluster-manager**』 アプリケーション」を参照してください。

たとえば、以下の **3** つのノードがあるとします。

- ノード **A**、**Load: 10**
- ノード **B**、**Load: 10**
- ノード **C**、**Load: 0**

この場合、負荷はノード **A** と **B** の間で分散されます。両方のノードが使用できない場合は、ノード **C** に負荷が集中します。

mod_manager

mod_manager ディレクティブのコンテキストは、指定がある場合を除きすべて **VirtualHost** になります。サーバー設定 コンテキストは、ディレクティブが **VirtualHost** 設定外になければならないことを示します。そうでない場合、エラーメッセージが表示され、**Apache HTTP Server** が開始しません。

表**17.7 mod_manager** の派生

派生	説明	値
EnableMCPMReceive	VirtualHost がノードから MCPM を受信できるようにします。mod_cluster が動作するようにするため、Apache HTTP Server 設定に EnableMCPMReceive が含まれます。VirtualHost のアドバタイズを設定する場所に保存します。	
MemManagerFile	設定の保存、共有メモリーまたはロックされたファイルのキーの生成に mod_manager が使用する名前のベース名。絶対パス名である必要があります。ディレクトリーは必要な場合に作成されます。 これらのファイルは NFS 共有ではなくローカルドライブに格納することが推奨されます。 コンテキスト: server config	**$server_root/logs/**
Maxcontext	mod_cluster によってサポートされるコンテキストの最大数。 コンテキスト: server config	デフォルト: 100
Maxnode	mod_cluster によってサポートされるノードの最大数。 コンテキスト: server config	デフォルト: 20
Maxhost	mod_cluster によってサポートされるホスト（エイリアス）の最大数。バランサーの最大数も含まれます。 コンテキスト: server config	デフォルト: 20
Maxsessionid	mod_cluster-manager ハンドラーにアクティブなセッションの数を提供するために保存されるアクティブな **sessionid** の数。5 分以内に mod_cluster がセッションから情報を受信しないとセッションは非アクティブになります。 コンテキスト: server config このフィールドはデモおよびデバッグの目的のみで使用されます。	0: 論理はアクティベートされない。
MaxMCMPMaxMessSize	他の Max ディレクティブからの MCMP メッセージの最大サイズ。	他の Max ディレクティブより計算されます。最小： 1024

派生	説明	値
ManagerBalancerName	JBoss EAP インスタンスがバランサー名を提供しない場合に使用されるバランサーの名前。	**mycluster**
PersistSlots	ファイルのノード、エイリアス、およびコンテキストを保持するよう mod_slotmem に伝えます。 コンテキスト: server config	オフ
CheckNonce	mod_cluster-manager ハンドラーを使用する際に **nonce** のチェックを切り替えます。	on/off デフォルト： on - **Nonce** をチェック
AllowDisplay	mod_cluster-manager メインページの追加表示を切り替えます。	on/off デフォルト: off - バージョンのみを表示
AllowCmd	mod_cluster-manager の URL を使用するコマンドを許可します。	on/off デフォルト: on - コマンドを許可
ReduceDisplay	メインの mod_cluster-manager ページに表示される情報を減らし、ページ上により多くのノードを表示できるようにします。	on/off デフォルト: off - 情報をすべて表示

SetHandler mod_cluster-manager	mod_cluster がクラスターから可視できるノードの情報を表示します。情報には一般的な情報が含まれ、追加でアクティブなセッションの数を調べます。
	
	`<Location /mod_cluster-manager>`
	`SetHandler mod_cluster-manager`
	`Order deny,allow`
	`Allow from 127.0.0.1`
	`</Location>`
		on/off
		
		デフォルト: off

注記

httpd.conf に定義された場所にアクセスする場合：

Transferred: バックエンドサーバーに送信された **POST** データに対応。

Connected: mod_cluster の状態ページが要求されたときに処理された要求の数に対応。

Num_sessions: mod_cluster がアクティブと報告するセッションの数に対応 **(**過去 **5** 分以内に要求があった場合**)**。**Maxsessionid** がゼロの場合、このフィールドは存在しません。 このフィールドはデモおよびデバッグの目的でのみ使用されます。

バグの報告

17.4.7. 外部 Web サーバーを JBoss EAP 6 アプリケーションの Web フロントエンドとして使用

概要

外部 **Web** サーバーを **Web** フロントエンドとして使用する理由や、**JBoss EAP 6** でサポートされるさまざまな **HTTP** コネクターの利点と欠点については、「**HTTP** コネクターの概要」 を参照してください。場合によっては、お使いのオペレーティングシステムに同梱される **Apache HTTP Server** を使用できます。それ以外の場合は、**JBoss Enterprise Web Server** の一部として同梱される **Apache HTTP Server** を使用できます。

使用する **Web** サーバーおよび **HTTP** コネクターを決定したら、以下のいずれかの手順を参照してください。

- 「**Red Hat Enterprise Linux 5**、**6**、**7** への **Apache HTTP Server** のインストール**(Zip)**」
- 「**Apache HTTP Server** または **JBoss Enterprise Web Server** への **mod_cluster** モジュールのインストール **(Zip)**」

- 「Apache HTTP Server への mod_jk モジュールのインストール (ZIP)」
- 「Microsoft IIS が ISAPI コネクターを使用するよう設定」
- 「Oracle Solaris での NSAPI コネクターの設定」
- 「外部 Web サーバーからの要求を許可するよう JBoss EAP 6 を設定」

バグの報告

17.4.8. 外部 Web サーバーからの要求を許可するよう JBoss EAP 6 を設定

概要

JBoss EAP 6 は、リクエストを許可するプロキシー、検索するポートおよびプロトコルのみを認識する必要はありません。これは mod_cluster ではなく、JBoss EAP 6 の設定と密接に結合されます。しかし、mod_jk、mod_proxy、ISAPI コネクター、および NSAPI コネクター では、以下のタスクが動作します。この例のプロトコルおよびポートを設定する必要のあるプロトコルおよびポートに置き換えます。

mod_cluster に対して JBoss EAP 6 を設定するには、「mod_cluster ワーカーノードの設定」 を参照してください。

前提条件

- このタスクを実行するには、管理 CLI または管理コンソールにログインする必要があります。タスクの正確な手順では管理 CLI を使用しますが、管理コンソールでは同じ基本的な手順が使用されます。
- 使用するプロトコル (HTTP、HTTPS、または AJP) のリストが必要です。

手順17.10 設定の編集およびソケットバインディングの追加

1. jvmRoute システムプロパティーを設定します。

スタンドアロンモードインスタンスの場合は、**/host=NODE_NAME** のプレフィックスを削除します。**NODE_NAME** は、ホスト名に置き換えます。

```
/host=NODE_NAME/system-property=jvmRoute/:add(value=NODE_NAME)
```

2. **Web** サブシステムで利用可能なコネクターをリストします。

注記

この手順は、スタンドアロンサーバーまたは管理対象ドメインのサーバーグループに **ha** または **full-ha** プロファイルを使用していない場合のみ必要になります。これらの設定には、必要なコネクターがすべて含まれています。

外部 **Web** サーバーが **JBoss EAP 6** の **Web** サーバーに接続できるようにするには、**web** サブシステムにコネクターが必要です。各プロトコルには、ソケットグループに関連付けられる独自のコネクターが必要です。

現在利用可能なコネクターをリストするには、以下のコマンドを実行します。

```
/subsystem=web:read-children-names(child-type=connector)
```

必要なコネクター **(HTTP**、**HTTPS**、**AJP)** を示す行がない場合は、コネクターを追加する必要があります。

3. コネクターの設定を確認します。

コネクターの設定方法の詳細を確認するには、その設定を読み取ります。以下のコマンドは、**AJP** コネクターの設定を読み取ります。他のコネクターの設定出力も同様になります。

```
/subsystem=web/connector=ajp:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "enable-lookups" => false,
        "enabled" => true,
        "max-post-size" => 2097152,
        "max-save-post-size" => 4096,
        "protocol" => "AJP/1.3",
        "redirect-port" => 8443,
        "scheme" => "http",
        "secure" => false,
        "socket-binding" => "ajp",
        "ssl" => undefined,
```

```
        "virtual-server" => undefined
    }
}
```

4. 必要なコネクターを **Web** サブシステムに追加します。

    コネクターを **Web** サブシステムに追加するには、ソケットバインディングが必要です。ソケットバインディングは、サーバーまたはサーバーグループによって使用されるソケットバインディンググループに追加されます。以下の手順は、サーバーグループが **server-group-one** で、ソケットバインディンググループが **standard-sockets** であることを前提としています。

    a. ソケットをソケットバインディンググループに追加します。

        ソケットをソケットバインディンググループに追加するには、以下のコマンドを実行し、プロトコルとポートを必要なものに置き換えます。

        ```
        /socket-binding-group=standard-sockets/socket-binding=ajp:add(port=8009)
        ```

    b. ソケットバインディングを **Web** サブシステムに追加します。

        以下のコマンドを発行してコネクターを **Web** サブシステムに追加し、ソケットバインディング名とプロトコルを必要なものに置き換えます。

        ```
        /subsystem=web/connector=ajp:add(socket-binding=ajp, protocol="AJP/1.3",
        enabled=true, scheme="http")
        ```

バグの報告

17.5. クラスタリング

17.5.1. クラスタリングサブシステムに TCP 通信を使用

デフォルトでは、クラスターノードは **UDP** プロトコルを使用して相互のステータスを監視します。一部のネットワークでは **TCP** のみを使用できます。このような場合には、**TCPPING** プロトコルスタックを設定に追加し、デフォルトのメカニズムとして使用することができます。これらの設定オプションは、コマンドラインベースの管理 **CLI** で利用できます。

mod_cluster サブシステムはデフォルトで **UDP** 通信も使用し、ここで **TCP** を使用することも可能です。

ネットワーク通信に **TCP** を使用するには、次の **2** つの手順を参照して **JGroups** および **mod_cluster** サブシステムを設定します。

- 「TCP を使用するよう JGroups サブシステムを設定」
- 「mod_cluster サブシステムのアドバタイズの無効化」

バグの報告

17.5.2. TCP を使用するよう JGroups サブシステムを設定

デフォルトでは、JGroups サブシステムはマルチキャスト UDP を使用して通信します。以下の手順に従って、代わりにユニキャスト TCP を使用するように JGroups サブシステムを設定します。

TCP も使用するように mod_cluster サブシステムを設定する場合は、「mod_cluster サブシステムのアドバタイズの無効化」 を参照してください。

1. お使いの環境に合わせて以下のスクリプトを変更します。

   以下のスクリプトをテキストエディターにコピーします。管理対象ドメインで別のプロファイルを使用する場合は、プロファイル名を変更します。スタンドアロンサーバーを使用する場合は、コマンドの /profile=full-ha の部分を削除します。以下のように、コマンドの下部に記載されているプロパティーを変更します。これらのプロパティーはそれぞれオプションです。

   initial_hosts

   よく知られており、最初のメンバーシップを検索するのに利用できる HOST[PORT] 構文を使用したホストのカンマ区切りのリスト。

   port_range

   必要に応じて、ポート範囲を割り当てることができます。ポート範囲を 2 に割り当て、ホストの初期ポートが 7600 である場合は、TCPPING はポート 7600-7602 のホストへの接続を試みます。ポート範囲は、initial_hosts で指定された各アドレスに適用されます。このプロパティーはオプションです。

   timeout

   クラスターメンバーの任意のタイムアウト値 (ミリ秒単位)

   num_initial_members

クラスターが完了とみなされるまでのノード数。このプロパティーはオプションです。

```
batch
##  If  tcp  is already added then you can remove it  ##
/profile=full-ha/subsystem=jgroups/stack=tcp:remove
/profile=full-ha/subsystem=jgroups/stack=tcp:add(transport={"type" =>"TCP", "socket-
binding" => "jgroups-tcp"})
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=TCPPING)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=MERGE2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FD_SOCK,socket-
binding=jgroups-tcp-fd)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FD)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=VERIFY_SUSPECT)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=BARRIER)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.NAKACK)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=UNICAST2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.STABLE)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=pbcast.GMS)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=UFC)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=MFC)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=FRAG2)
/profile=full-ha/subsystem=jgroups/stack=tcp/:add-protocol(type=RSVP)
/profile=full-ha/subsystem=jgroups:write-attribute(name=default-stack,value=tcp)
run-batch
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=initial_hosts/:add(value="HostA[
600],HostB[7600]")
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=port_range/:add(value=0)
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=timeout/:add(value=3000)
/profile=full-
ha/subsystem=jgroups/stack=tcp/protocol=TCPPING/property=num_initial_members/:add(value
=3)
```

2. スクリプトをバッチモードで実行します。

警告

バッチファイルを実行する前に、プロファイルを実行しているサーバーをシャットダウンする必要があります。

端末エミュレーターで、**jboss-cli.sh** スクリプトが含まれるディレクトリーに移動し、以下のコマンドを入力します。

```
./jboss-cli.sh -c --file=SCRIPT_NAME
```

SCRIPT_NAME は、スクリプトを含むパスの名前に置き換えます。

結果

TCPPING スタックが *JGroups* サブシステムで利用できるようになりました。これを使用すると、*JGroups* サブシステムはすべてのネットワーク通信に *TCP* を使用します。*TCP* も使用するように *mod_cluster* サブシステムを設定する場合は、「*mod_cluster* サブシステムのアドバタイズの無効化」 を参照してください。

バグの報告

17.5.3. mod_cluster サブシステムのアドバタイズの無効化

デフォルトでは、*mod_cluster* サブシステムのバランサーはマルチキャスト *UDP* を使用して可用性をバックグラウンドワーカーにアドバタイズします。必要に応じて、アドバタイズメントを無効にできます。この動作を設定するには、以下の手順を使用します。

手順17.11

1. *Apache HTTP Server* 設定を変更します。

   Apache HTTP Server 設定を変更し、サーバーのアドバタイズを無効にし、代わりにプロキシーリストを使用します。プロキシー一覧はワーカーで設定され、ワーカーが対話できるすべての *mod_cluster*対応 *Web* サーバーが含まれます。

   Web サーバーの *mod_cluster* 設定は *HTTPD_HOME* にあります。ファイル自体の詳細は、「*Apache HTTP Server* または *JBoss Enterprise Web Server* への *mod_cluster* モジュールのインストール *(Zip)*」 および 「*mod_cluster* が有効な *Web* サーバーに対するサーバーアドバタイズメントプロパティーの設定」 を参照してください。*MCPM* 要求をリッスンする仮想ホスト（ *EnableMCPMReceive* ディレクティブを使用）が含まれるファイルを開き、以下のように *ServerAdvertise* ディレクティブを変更してサーバーのアドバタイズを無効にします。

   ```
   ServerAdvertise Off
   ```

2. *JBoss EAP 6* の *mod_cluster* サブシステム内でアドバタイズを無効にし、プロキシーのリストを提供します。

   Web ベースの管理コンソールまたはコマンドライン管理 *CLI* を使用して、*mod_cluster* サブシステムのアドバタイズを無効にし、プロキシーのリストを提供できます。*mod_cluster* サ

ブシステムはアドバタイズが無効になっているとプロキシーを自動的に検出できないため、プロキシーのリストが必要です。

- *管理コンソール*

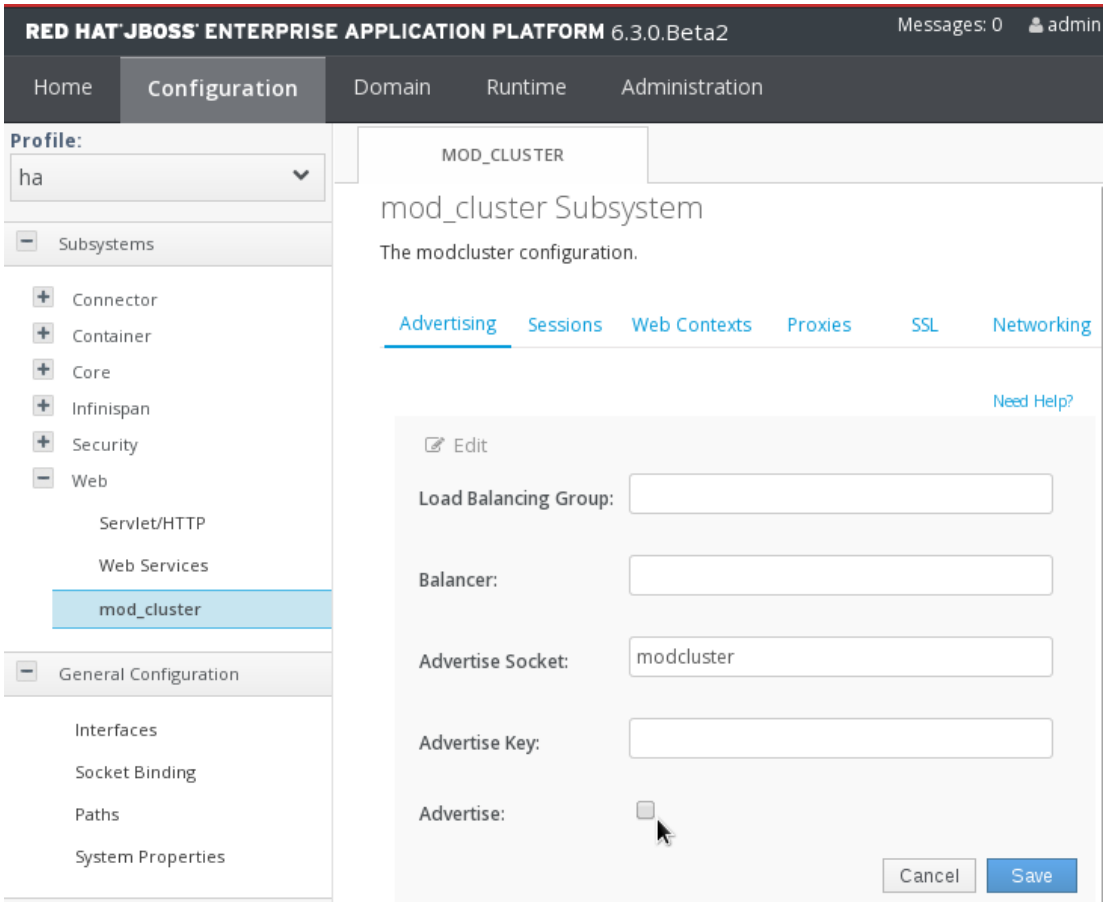
  *管理対象ドメインを使用する場合は、**mod_cluster** プロファイルや **ha** プロファイルなど、有効化されているプロファイルでのみ **full-ha** を設定できます。*

1. *管理コンソールにログインし、画面の上部にある **Configuration** タブを選択します。管理対象ドメインを使用する場合は、左上の **Profile** ドロップダウンメニューから **ha** または **full-ha** プロファイルを選択します。*

2. ***Subsystems** メニューを展開し、**Web** メニューを展開し、**mod_cluster** を選択します。*

3. ***mod_cluster** の **Advertising** タブで **Edit** をクリックします。アドバタイズを無効にするには、**Advertise** の横にあるチェックボックスの選択を解除し、**Save** をクリックします。*

*図17.1 **mod_cluster** アドバタイズ設定画面*



4. プロキシー タブをクリックします。**Edit** をクリックし、**Proxy List** フィールドにプロキシーサーバーの一覧を入力します。正しい構文は、以下のような **HOSTNAME:PORT** 文字列のコンマ区切りリストです。

```
10.33.144.3:6666,10.33.144.1:6666
```

Save ボタンをクリックして終了します。

- 管理 **CLI**

以下の **2** つの管理 **CLI** コマンドは、上記の管理コンソールの手順と同じ設定を作成します。ここでは、管理対象ドメインを実行し、サーバーグループが **full-ha** プロファイルを使用していることを仮定します。別のプロファイルを使用する場合は、コマンドで名前を変更します。**standalone-ha** プロファイルを使用してスタンドアロンサーバーを使用する場合は、コマンドの **/profile=full-ha** の部分を削除します。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=false)

/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="10.33.144.3:6666,10.33.144.1:6666")
```

結果

Apache HTTP Server のバランサーがその存在をワーカーノードにアドバタイズしなくなり、**UDP** マルチキャストが使用されないようになります。

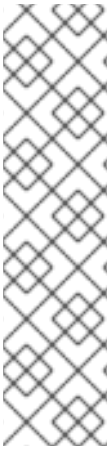
注記

属性 **advertise="false"** を設定するには、**proxy-list="address:port"** 属性も設定する必要があります。**proxy-list** 属性が空の場合、**advertise="false"** 属性は無視されます。**mod_cluster** サブシステムを完全に無効にするには、サーバー設定から削除します。

バグの報告

17.5.4. HornetQ クラスタリングの UDP の TCP への変更

以下の例では、EAP 6 に同梱されるデフォルトの standalone-full-ha.xml ファイルが使用されます。



注記

セキュリティーが有効になっている場合、cluster-password 属性を設定する必要があります。

```
<cluster-password>${jboss.messaging.cluster.password:ChangeMe>}</cluster-password>
```

1. broadcast-groups および discovery-groups を削除します。

```
<broadcast-groups>
   <broadcast-group name="bg-group1">
      <socket-binding>messaging-group</socket-binding>
      <broadcast-period>5000</broadcast-period>
      <connector-ref>netty</connector-ref>
   </broadcast-group>
</broadcast-groups>
<discovery-groups>
   <discovery-group name="dg-group1">
      <socket-binding>messaging-group</socket-binding>
      <refresh-timeout>10000</refresh-timeout>
   </discovery-group>
</discovery-groups
```

2. "messaging-group" ソケットバインディングを削除します (任意設定)。

```
<socket-binding name="messaging-group" port="0" multicast-address="${jboss.messaging.group.address:231.7.7.7}" multicast-port="${jboss.messaging.group.port:9876}"/>
```

3. 適切な Netty コネクターを設定します (クラスターの他のノードごとに 1 つ)。

たとえば、クラスターが **3** ノードの場合は **2** つの **Netty** コネクターを設定し、クラスターが **2** ノードの場合は **1** つの **Netty** コネクターを設定します。**3** ノードクラスターの設定例を以下に示します。

```
<netty-connector name="other-cluster-node1" socket-binding="other-cluster-node1"/>
<netty-connector name="other-cluster-node2" socket-binding="other-cluster-node2"/>
```

4. 関連するソケットバインディングを設定します。

注記

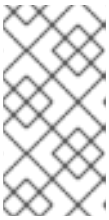
必要な場合は、**host** または **port** に対してシステムプロパティーの置換を使用できます。

```
<outbound-socket-binding name="other-cluster-node1">
    <remote-destination host="otherNodeHostName1" port="5445"/>
</outbound-socket-binding>
<outbound-socket-binding name="other-cluster-node2">
    <remote-destination host="otherNodeHostName2" port="5445"/>
</outbound-socket-binding>
```

5. デフォルトで使用される **discovery-group** の代わりに、これらのコネクターを使用する **cluster-connection** を設定します。

```
<cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty</connector-ref>
    <static-connectors>
        <connector-ref>other-cluster-node1</connector-ref>
        <connector-ref>other-cluster-node2</connector-ref>
    </static-connectors>
</cluster-connection>
```

各ノードが、クラスターの他のノードすべて対してコネクターを持つように、この処理を各クラスターノードで繰り返す必要があります。



注記

ノード自体への接続を設定しないでください。これは、設定が間違っていると見なされます。

バグの報告

17.6. WEB、HTTP コネクター、および HTTP クラスタリング

17.6.1. mod_cluster HTTP コネクター

mod_cluster モジュールは負荷分散を有効にし、コネクター と呼ばれます。他のコネクターについては、以下のいずれかを参照してください。

- 「Apache mod_jk HTTP コネクター」
- 「インターネットサーバー API(ISAPI)について」
- 「Netscape Server API(NSAPI)」

他のコネクターと比べて **mod_cluster** コネクターには複数の利点があります。

- *mod_cluster Management Protocol (MCMP)は、JBoss Enterprise Application Platform 6 サーバーと mod_cluster モジュールが有効になっている Apache HTTP Server との間の追加接続です。HTTP メソッドのカスタムセットを使用してサーバー側の負荷分散係数およびライフサイクルイベントを Apache HTTP Server サーバーに返信するために JBoss Enterprise Application Platform サーバーによって使用されます。*

- *mod_cluster がある Apache HTTP Server を動的に設定すると、手動で設定せずに JBoss EAP 6 サーバーが負荷分散配置に参加できます。*

- *JBoss EAP 6 は、mod_cluster を使用する Apache HTTP Server に依存せずに負荷分散係数の計算を行います。これにより、負荷分散メトリックが他のコネクターよりも正確になります。*

- *mod_cluster コネクターにより、アプリケーションライフサイクルを細かく制御できるようになります。各 JBoss EAP 6 サーバーは Web アプリケーションコンテキストライフサイクルイベントを Apache HTTP Server に転送するため、特定のコンテキストのルーティングリクエストを開始または停止するよう通知します。これにより、リソースを使用できないことが原因の HTTP エラーがエンドユーザーに表示されないようになります。*

- *AJP、HTTP、または HTTPS トランスポートを使用できます。*

バグの報告

17.6.2. mod_cluster サブシステムの設定

mod_cluster サブシステムは、管理コンソールおよび管理 CLI を使用して設定できます。このトピックでは、管理コンソールに表示されるさまざまな設定オプションについて説明します。各オプションの管理 CLI コマンドの例が提供されます。

注記

mod_cluster 設定ページは、ha プロファイルと full- ha プロファイルでのみ表示されます。管理対象ドメインの場合、これらのプロファイルは ha および full-ha で、スタンドアロンサーバーの場合は standalone-ha および standalone-full-ha になります。

管理コンソール

Configuration タブをクリックします。管理対象ドメインを設定する場合は、Profile ドロップダウ

*ンリストから正しいプロファイルを選択します。**Subsystems** メニューを展開し、**Web** メニューを展開し、**mod_cluster** を選択します。*

*表17.8 **mod_cluster** アドバタイズの設定オプション*

オプション	説明	CLI コマンド
負荷分散グループ (Load Balancing Group)	null でない場合、要求はロードバランサーの特定の負荷分散グループに送信されます。ロードバランシンググループを使用しない場合は、空欄のままにします。これは、デフォルトでは未設定です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=myGroup)
バランサー (Balancer)	この属性は、Apache HTTP Server の mod_cluster により自動的に設定される mod_proxy バランサーを指定します。デフォルトは **none** です。この場合、**mycluster** のデフォルトが使用されます（mod_proxy 用語で表される場合は**balancer://mycluster/**）。このデフォルト値は、ManagerBalancerName ディレクティブで Apache HTTP Server 側で設定されます。 JBoss EAP 6 のワーカーインスタンスで 2 つの異なる **balancer** 属性値を使用する場合、Apache HTTP Server で mod_cluster によって自動的に作成される 2 つの異なる mod_proxy バランサーがあります。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=balancer,value=myBalancer)
ソケットのアドバタイズ (Advertise Socket)	クラスターのアドバタイズに使用するソケットバインディングの名前です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-socket,value=modcluster)
セキュリティーキーのアドバタイズ (Advertise Security Key)	アドバタイズのセキュリティーキーが含まれる文字列。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-security-key,value=myKey)

オプション	説明	CLI コマンド
アドバタイズ (Advertise)	アドバタイズを有効にするかどうか。デフォルトは **true** です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=true)

表17.9 mod_cluster セッション設定オプション

オプション	説明	CLI コマンド
スティッキーセッション (Sticky Session)	リクエストにスティッキーセッションを使用するかどうか。つまり、クライアントが特定のノードに接続すると、利用不可能でない限り、追加の通信が同じノードにルーティングされます。デフォルトは **true** で、推奨される設定です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)
スティッキーセッションの強制 (Sticky Session Force)	**true** の場合、最初のノードが利用できなくなりますが、失敗すると、リクエストは新しいノードにリダイレクトされません。デフォルトは **false** です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-force,value=false)
スティッキーセッションの削除 (Sticky Session Remove)	フェイルオーバー時にセッション情報を削除します。デフォルトは **false** です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-remove,value=false)

表17.10 mod_cluster Web コンテキスト設定オプション

オプション	説明	CLI コマンド

オプション	説明	CLI コマンド
コンテキストの自動有効化 (Auto Enable Contexts)	デフォルトで新しいコンテキストを **mod_cluster** に追加するかどうか。デフォルトは **true** です。デフォルトを変更し、コンテキストを手動で有効にする必要がある場合、Web アプリケーションは **enable()** MBean メソッドを使用するか、**mod_cluster** マネージャーを使用してコンテキストを有効にできます。これは、httpd の設定に指定される名前付き仮想ホストまたはポートで httpd プロキシーで実行される Web アプリケーションです。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=auto-enable-contexts,value=true)
除外されたコンテキスト (Excluded Contexts)	**mod_cluster** が無視すべきコンテキストのコンマ区切りリスト。ホストが指定されていない場合、ホストは **localhost** であると想定されます。**ROOT** Web アプリケーションのルートコンテキストを示します。デフォルト値は **ROOT,invoker,jbossws,juddi,console** です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=excluded-contexts,value="ROOT,invoker,jbossws,juddi,console")

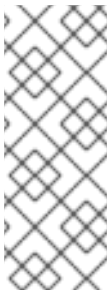
表17.11 mod_cluster プロキシー設定オプション

オプション	説明	CLI コマンド
プロキシー URL	定義された場合、この値は MCMP コマンドの URL の前に付加されます。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-url,value=myhost)
プロキシリスト (Proxy List)	**hostname:port** 形式の httpd プロキシーアドレスのコンマ区切りリスト。これは、**mod_cluster** プロセスが最初の通信を試みるプロキシーのリストを示します。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=proxy-list,value="127.0.0.1,127.0.0.2")

SSL 通信の設定 mod_cluster

*デフォルトでは、**mod_cluster** 通信は暗号化されていない **HTTP** リンク上で行われます。コネクタースキームを **HTTPS** に設定すると（表17.9「mod_cluster セッション設定オプション」を参照）、以下の設定は、接続を暗号化する情報を見つける場所を **mod_cluster** に指示します。*

表17.12 mod_cluster SSL 設定オプション

オプション	説明	CLI コマンド
キーエイリアス (Key Alias)	証明書が作成されたときに選択されたキーエイリアス。	/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-alias,value=jboss)
パスワード	このパスワードは、certificate-key-file(Key File)と ca-certificate-file(Cert File)の両方のキーストアパスワードで、Cert File 内の Key Alias で指定するキー/証明書エントリーの両方です。  **注記** @CA-certificate-password はトラストストアのパスワードであり、指定がない場合は値は undefined のままになります。	/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=password,value=changeit)
CA 証明書ファイル/トラストストア	Web サーバー証明書の検証に使用されるトラストストア。	/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ca-certificate-file,value=${user.home}/jboss.crt)
キーストア (Key Store)	このインスタンスを識別する証明書および秘密鍵を保持するキーストア。	/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=certificate-key-file,value=${user.home}/.keystore)

オプション	説明	CLI コマンド
暗号スイート (Cipher Suite)	許可された暗号スイート。	`/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=cipher-suite,value=ALL)`
失効 URL (Revocation URL)	証明局失効リストの URL。	`/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ca-revocation-url,value=jboss.crl)`
プロトコル	有効 な SSL プロトコル。 また、プロトコルの組み合わせ（コンマ区切り）を指定することもできます。例： TLSv1、TLSv1.1、TLSv1.2 **警告** Red Hat は、影響を受けるすべてのパッケージで TLSv1.1 または TLSv1.2 を利用するために SSL を明示的に無効にすることを推奨します。	`/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=protocol,value="TLSv1, TLSv1.1,TLSv1.2")`

***mod_cluster** ネットワーク設定*

*利用可能な **mod_cluster** ネットワークオプションは、**mod_cluster** サービスが通信するさまざまなタイプのサービスに対して、さまざまなタイムアウト動作を制御します。*

表17.13 mod_cluster ネットワーク設定オプション

オプション	説明	CLI コマンド
ノードタイムアウト (Node Timeout)	ワーカーへのプロキシー接続のタイムアウト (秒単位)。mod_cluster はこの期間バックエンド応答を待ち、その経過後にエラーを返します。**node-timeout** 属性が未定義の場合は、httpd **ProxyTimeout** ディレクティブが使用されます。**ProxyTimeout** が定義されていない場合は、httpd **Timeout** ディレクティブが使用されます。デフォルトは **300** 秒です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=node-timeout,value=-1)
ソケットタイムアウト (Socket Timeout)	タイムアウトの前およびプロキシーをエラーのように警告する前に、httpd プロキシーから MCMP コマンドへの応答を待つ秒数。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=socket-timeout,value=20)
停止コンテキストタイムアウト (Stop Context Timeout)	コンテキストがクリーンにシャットアウトされるまでの、stopContextTimeoutUnit で指定された単位の時間 (配布可能なコンテキストに対する保留中の要求の完了、または配布可能でないコンテキストに対するアクティブなセッションの破棄/期限切れ)。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=stop-context-timeout,value=10)
セッション排出ストラテジ (Session Draining Strategy)	Web アプリケーションをアンデプロイする前にセッションを排出 (drain) するかどうか。 **DEFAULT** web アプリケーションが分散可能でない場合のみ、web アプリケーションのアンデプロイ前にセッションをドレインします。 **ALWAYS** web アプリケーションが分散可能であっても、web アプリケーションのアンデプロイ前に常にセッションをドレインします。 **NEVER** Web アプリケーションをアンデプロイする前にセッションを排出 (drain) しません (配布不可能な Web アプリケーションを含む）。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=session-draining-strategy,value=DEFAULT)

オプション	説明	CLI コマンド
最大試行回数 (Max Attempts)	httpd プロキシーがノードに対して指定のリクエストの送信を試みる回数。この回数試行してから停止します。最小値は **1** で、1回のみ試行します。**mod_proxy** のデフォルト値は **1** でもあるため、再試行は行われません。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=max-attempts,value=1)
パケットのフラッシュ (Flush Packets)	Web サーバーへのパケットのフラッシュを有効にするかどうか。デフォルトは **false** です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-packets,value=false)
フラッシュの待機 (Flush Wait)	Web サーバーにパケットをフラッシュするまでの待機時間（秒単位）。デフォルトは **-1** です。**-**1 を値として指定すると、パケットをフラッシュする前に永久に待機します。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-wait,value=-1)
Ping	ワーカーからの ping への応答を待つ時間（秒単位）。デフォルトは **10** 秒です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ping,value=10)
SMAX	ソフト最大アイドル接続数（mod_proxy ドキュメントでは **smax** と同じ）。最大値は httpd スレッド設定によって異なり、**ThreadsPerChild** または **1** のいずれかになります。	profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=smax,value=ThreadsPerChild)
TTL	smax より上のアイドル接続の残存時間 (秒数単位)。デフォルト値は 60 です。	/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ttl,value=-1)

***mod_cluster Load Provider** 設定オプション*

*以下の **mod_cluster** 設定オプションは管理コンソールでは使用できませんが、管理 **CLI** を使用してのみ設定できます。*

*動的ロードプロバイダーが存在しない場合には、単純なロードプロバイダーが使用されます。各クラスターメンバーに負荷係数 **1** を割り当て、負荷分散アルゴリズムを適用せずに作業を均等に分散します。これを追加するには、以下の管理 **CLI** コマンドを使用します。*

```
[standalone@localhost:9990 /] /subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=simple-load-provider, value=1)
```

*次のリクエストを受信するワーカーを決定するために、動的ロードプロバイダーはさまざまなアルゴリズムの組み合わせを使用するように設定できます。ロードプロバイダーを作成してお使いの環境に合わせて設定できます。また、**CLI** で複数の負荷メトリクスを同時に追加することで、同時に複数の負荷メトリックをアクティブにすることができます。デフォルトの動的ロードプロバイダーは負荷メトリックの決定にビジー状態を使用します。動的ロードプロバイダーオプションと可能な負荷メトリックを以下に示します。*

*表**17.14 mod_cluster** 動的ロードプロバイダーオプション*

オプション	説明	CLI コマンド
衰退 (Decay)	履歴メトリックの重要性が衰退すべき要因。	`/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=decay,value=2)`
履歴	負荷を決定するときに考慮する、負荷メトリックの履歴記録数。	`/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=history,value=9)`

オプション	説明	CLI コマンド
負荷メトリック (Load Metric)	JBoss EAP 6 の動的ロードプロバイダーに含まれるデフォルトの負荷メトリックはビジー状態であり、リクエストに対応するスレッドプールのスレッド数からワーカーの負荷を計算します。**実際の負荷を分割するこのメトリックの容量を設定できます (calculated_load / capacity)。動的ロードプロバイダー内に複数の負荷メトリクスを設定できます。**	/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=capacity,value=1.0) /subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=type,value=busyness) /subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=busyness/:write-attribute(name=weight,value=1)

負荷メトリックアルゴリズム

cpu

__cpu__ 負荷メトリックは、__CPU__ 負荷の平均を使用して次のワークロードを取得するノードを決定します。

mem

__mem__ 負荷メトリックは、空きネイティブメモリーを負荷メトリックとして使用します。このメトリクスの使用は推奨されません。これはバッファーやキャッシュを含む値を提供するため、メモリー管理が良いすべてのシステムにおいて非常に低い値になります。

heap

ヒープ 負荷メトリックは、ヒープ使用率を使用して次のワークロードを取得するワーカーを決定します。

セッション

セッション 負荷メトリックは、アクティブなセッションの数をメトリックとして使用します。

requests

*要求 負荷メトリックは、クライアント要求の数を使用して次のワークロードを受け取るワーカーを決定します。たとえば、容量 **1000** は、**1000** 要求/秒がフルロードであると見なされます。*

send-traffic

***send-traffic** 負荷メトリックは、ワーカーからクライアントに送信されるトラフィックの量を使用します。たとえば、デフォルトの容量が **512** の場合は、平均送信トラフィックが **512 KB**/秒以上である場合に、ノードがフルロードであると見なされます。*

receive-traffic

***receive-traffic** 負荷メトリックは、クライアントからワーカーに送信されるトラフィックの量を使用します。たとえば、デフォルトの容量が **1024** の場合は、平均受信トラフィックが **1024 KB**/秒以上である場合にワーカーがフルロードであると見なされます。*

busyness

このメトリックは、要求の処理で負荷が大きいスレッドプールからのスレッドの量を表します。

*例**17.2** 負荷メトリックの追加*

*負荷メトリックを追加するには、**add-metric** コマンドを使用します。*

```
/subsystem=modcluster/mod-cluster-config=configuration/:add-metric(type=sessions)
```

*例**17.3** 既存メトリックの値設定*

*既存のメトリックの値を設定するには、**write-attribute** コマンドを使用します。*

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-
provider=configuration/load-metric=cpu/:write-attribute(name="weight",value="3")
```

例**17.4** 既存メトリックの値変更

既存のメトリックの値を変更するには、**write-attribute** コマンドを使用します。

```
/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/load-metric=cpu/:write-attribute(name="type",value="busyness")
```

例**17.5** 既存メトリックの削除

既存のメトリックを削除するには、**remove-metric** コマンドを使用します。

```
/subsystem=modcluster/mod-cluster-config=configuration/:remove-metric(type=sessions)
```

バグの報告

17.6.3. Apache HTTP Server または JBoss Enterprise Web Server への mod_cluster モジュールのインストール (Zip)

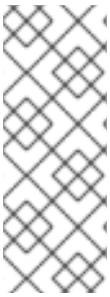
前提条件

- このタスクを実行するには、**Red Hat Enterprise Linux 6** または **JBoss Enterprise Web Server** にインストールされた **Apache HTTP Server** を使用するか、**JBoss EAP 6** のダウンロード可能な個別コンポーネントとして含まれるスタンドアロン **Apache HTTP Server** を使用する必要があります。
- **Red Hat Enterprise Linux 6** に **Apache HTTP Server** をインストールする必要がある場合は、『**Red Hat Enterprise Linux 6 デプロイメントガイド**』に記載された手順を実行します。
- **JBoss EAP 6** の個別のダウンロード可能なコンポーネントとして含まれるスタンドアロンの **Apache HTTP Server** をインストールする必要がある場合は、「**Red Hat Enterprise Linux 5、6、7 への Apache HTTP Server のインストール(Zip)**」を参照してください。
- **JBoss Enterprise Web Server** をインストールする必要がある場合は、『**JBoss Enterprise Web Server インストールガイド**』に記載された手順を実行します。

- *Red Hat カスタマーポータル(https://access.redhat.com)から、お使いのオペレーティングシステムおよびアーキテクチャー用の Webserver Connecter Natives パッケージをダウンロードします。このパッケージには、お使いのオペレーティングシステム用に事前にコンパイルされた mod_cluster バイナリー Web サーバーモジュールが含まれます。アーカイブを抽出した後に、モジュールは EAP_HOME/modules/system/layers/base/native/lib/httpd/modules ディレクトリーにあります。*

  etc/ ディレクトリーには設定ファイルのサンプルが含まれ、share/ ディレクトリーには補足ドキュメントがいくつか含まれています。

- *管理 (root) 権限を使用してログインする必要があります。*



備考

64 ビットシステムを使用する場合、mod_cluster バイナリー web サーバーモジュールは EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules に配置されます。モジュールへのアクセスが必要な場合は、このパスを使用する必要があります。

手順17.12 mod_cluster モジュールのインストール

1. *Apache HTTP Server の設定場所を判断します。*

   Apache HTTP Server の設定場所は、Red Hat Enterprise Linux の Apache HTTP Server を使用しているか、JBoss EAP 6 でダウンロード可能な個別コンポーネントとして含まれるスタンドアロン Apache HTTP Server を使用しているか、JBoss Enterprise Web Server で利用可能な Apache HTTP Server を使用しているかによって異なります。これは、以下の 3 つのオプションの 1 つであり、このタスクの残りの部分では HTTPD_HOME と呼ばれます。

   - *Apache HTTP Server - /etc/httpd/*
   - *JBoss EAP 6 Apache HTTP Server - この場所はインフラストラクチャーの要件に基づいてユーザーによって選択されます。*
   - *JBoss Enterprise Web Server Apache HTTP Server - EWS_HOME/httpd/*

2. *モジュールを Apache HTTP Server モジュールディレクトリーにコピーします。*

   4 つのモジュール（ .so で終わるファイル）を、展開された Webserver Natives アーカイブの EAP_HOME/modules/system/layers/base/native/lib/httpd/modules ディレクトリーか

ら **HTTPD_MODULES/** ディレクトリーにコピーします。

3. **JBoss Enterprise Web Server** の場合は、**mod_proxy_balancer** モジュールを無効にします。

JBoss Enterprise Web Server を使用する場合、**mod_proxy_balancer** モジュールはデフォルトで有効になります。これは **mod_cluster** と互換性がありません。これを無効にするには、**HTTPD_CONF/httpd.conf** を編集し、モジュールを読み込む行の前に **#**（ハッシュ）記号を追加して以下の行をコメントアウトします。この行はコメントなしで表示され、以下のように表示されます。

```
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

```
# LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

ファイルを保存してから閉じます。

4. **mod_cluster** モジュールを設定します。

Webserver Natives アーカイブには、サンプルの **mod_cluster.conf** ファイル **(EAP_HOME/modules/system/layers/base/native/etc/httpd/conf)**が含まれます。このファイルはガイドとして使用するか、または編集して **HTTPD_CONF.D/JBoss_HTTP.conf** ファイルを作成します。

注記

JBoss_HTTP.conf という名前を使用することは、本書の任意の規則です。設定ファイルは、**.conf** 拡張子が **conf.d/** ディレクトリーに保存されている場合、その名前に関係なく読み込まれます。

以下を設定ファイルに追加します。

```
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so
```

これにより、**Apache HTTP Server** は **mod_cluster** が機能するために必要なモジュールを

自動的に読み込みます。

5. プロキシサーバーリスナーを作成します。

HTTPD_CONF.D/JBoss_HTTP.conf の編集を継続し、大文字の値を環境に適した値に置き換えて、以下の最低限の設定を追加します。

```
Listen IP_ADDRESS:PORT
<VirtualHost IP_ADDRESS:PORT>
  <Location />
        Order deny,allow
        Deny from all
        Allow from *.MYDOMAIN.COM
  </Location>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0
  EnableMCPMReceive

  ManagerBalancerName mycluster
  ServerAdvertise On

</VirtualHost>
```

これらのディレクティブは、**IP_ADDRESS:PORT** でリッスンする新しい仮想サーバーを作成し、**MYDOMAIN.COM** からの接続を許可し、**mycluster** という名前のバランサーとしてアドバタイズします。これらのディレクティブは、**Apache Web Server** のドキュメントに記載されています。**ServerAdvertise** ディレクティブおよび **EnableMCPMReceive** ディレクティブの詳細とサーバーアドバタイズの影響については、「**mod_cluster** が有効な **Web** サーバーに対するサーバーアドバタイズメントプロパティーの設定」 を参照してください。

ファイルを保存して終了します。

6. **Apache HTTP** サーバーを再起動します。

Apache HTTP Server の再起動方法は、**Red Hat Enterprise Linux** の **Apache HTTP Server** を使用しているか、**JBoss Enterprise Web Server** に含まれる **Apache HTTP Server** を使用しているかによって異なります。以下の **2** つの方法のいずれかを選択します。

- **Red Hat Enterprise Linux 6** の **Apache HTTP Server**

以下のコマンドを実行します。

```
[root@host]# service httpd restart
```

- ***JBoss Enterprise Web Server* の *Apache HTTP Server***

  ***JBoss Enterprise Web Server* は、*Red Hat Enterprise Linux* と *Microsoft Windows Server* の両方で実行されます。*Apache HTTP Server* の再起動方法はそれぞれ異なります。**

  - ***Red Hat Enterprise Linux***

    ***Red Hat Enterprise Linux* では、*JBoss Enterprise Web Server* は *Apache HTTP Server* をサービスとしてインストールします。*Apache HTTP Server* を再起動するには、以下の *2* つのコマンドを実行します。**

    ```
    [root@host ~]# service httpd stop
    [root@host ~]# service httpd start
    ```

  - ***Microsoft Windows Server***

    コマンドプロンプトで以下のコマンドを管理権限で実行します。

    ```
    C:\> net stop httpd
    C:\> net start httpd
    ```

結果

***Apache HTTP Server* がロードバランサーとして設定され、*JBoss EAP 6* を実行している *mod_cluster* サブシステムを使用できるようになりました。*JBoss EAP 6* が *mod_cluster* を認識するよう設定するには、「*mod_cluster* ワーカーノードの設定」 を参照してください。**

バグの報告

17.6.4. Apache HTTP Server または JBoss Enterprise Web Server への mod_cluster モジュールのインストール (RPM)

前提条件

- このタスクを実行するには、Red Hat Enterprise Linux 6 または JBoss Enterprise Web Server にインストールされた Apache HTTP Server を使用するか、JBoss EAP 6 のダウンロード可能な個別コンポーネントとして含まれるスタンドアロン Apache HTTP Server を使用する必要があります。
- Red Hat Enterprise Linux 6 に Apache HTTP Server をインストールする必要がある場合は、『『Red Hat Enterprise Linux 6 デプロイメントガイド』』 に記載された手順を実行します。
- JBoss EAP 6 の個別のダウンロード可能なコンポーネントとして含まれるスタンドアロンの Apache HTTP Server をインストールする必要がある場合は、「Red Hat Enterprise Linux 5、6、7 への Apache HTTP Server のインストール(Zip)」 を参照してください。
- JBoss Enterprise Web Server をインストールする必要がある場合は、『JBoss Enterprise Web Server インストールガイド』 に記載された手順を実行します。
- 管理 (root) 権限を使用してログインする必要があります。
- jbappplatform-6-ARCH-server-VERS-rpm RHN チャンネルへのアクティブなサブスクリプションが必要です。

Red Hat Enterprise Linux 6 での RPM のインストール方法は、Red Hat Enterprise Linux 5 の場合とほぼ同様で、Red Hat Enterprise Linux 6 に Apache HTTP Server 2.2.15 がインストールされている場合は多少の変更のみが必要となります。

1. YUM を使用して mod_cluster-native パッケージをインストールします。

```
yum install mod_cluster-native
```

2. Apache HTTP Server:

   - Apache HTTP Server 2.2.15 の使用を継続する場合は、httpd.conf ファイルの

LoadModule proxy_balancer_module 行をデフォルトで読み込む **mod_proxy_balancer** モジュールを無効にする必要があります。

ファイルを手作業で編集するか、以下のコマンドを使用します。

```
sed -i 's/^LoadModule proxy_balancer_module/#LoadModule
proxy_balancer_module/;s/$//' /etc/httpd/conf/httpd.conf
```

- **Apache HTTP Server 2.2.26** へアップグレードする場合は、以下のコマンドを使用して最新バージョンをインストールします。

```
yum install httpd
```

3. ブート時に **Apache HTTP Server** サービスが起動するようにするには、以下のコマンドを実行します。

   - **Red Hat Enterprise Linux 5** および **6** の場合:

```
service httpd add
```

   - **Red Hat Enterprise Linux 7** の場合:

```
systemctl enable httpd22.service
```

4. 以下のコマンドを使用して、**mod_cluster** バランサーを起動します。

   - **Red Hat Enterprise Linux 5** および **6** の場合:

```
service httpd start
```

- Red Hat Enterprise Linux 7 の場合:

```
systemctl start httpd22.service
```

バグの報告

17.6.5. mod_cluster が有効な Web サーバーに対するサーバーアドバタイズメントプロパティーの設定

概要

mod_cluster ロードバランサーと対話するように Web サーバーを設定する手順は、「Apache HTTP Server または JBoss Enterprise Web Server への mod_cluster モジュールのインストール (Zip)」 を参照してください。より詳細な説明が必要な設定の 1 つは、サーバーアドバタイズメント です。

サーバーのアドバタイズがアクティブになると、Web サーバーは mod_cluster 仮想ホストに指定された IP アドレスとポート番号を含むメッセージをブロードキャストします。これらの値を設定するには、「Apache HTTP Server または JBoss Enterprise Web Server への mod_cluster モジュールのインストール (Zip)」 を参照してください。UDP マルチキャストがネットワークで利用できない場合や、プロキシーサーバーの静的リストでワーカーを設定する場合は、サーバーのアドバタイズメントを無効にし、ワーカーノードを手動で設定することができます。ワーカーの設定に関する詳細は、「mod_cluster ワーカーノードの設定」 を参照してください。

この手順の変更は、Apache HTTP Server インスタンスに関連付けられた httpd.conf に追加する必要があります。多くの場合、Red Hat Enterprise Linux では /etc/httpd/conf/httpd.conf になります。または、スタンドアロンの Apache HTTP Server インスタンスの etc/ ディレクトリーにある場合があります。

手順17.13 httpd.conf ファイルを編集し、変更を実装する

1. AdvertiseFrequency パラメーターを無効にします（存在する場合）。

   < VirtualHost > ステートメントに以下のような行がある場合は、最初の文字の前に # （ハッシュ）文字を置くことでコメントアウトします。値は 5 とは異なる場合があります。

```
AdvertiseFrequency 5
```

2. サーバーアドバタイズメントを無効にするディレクティブを追加します。

   < VirtualHost> ステートメント内に以下のディレクティブを追加し て、サーバーのアドバタイズを無効にします。

```
ServerAdvertise Off
```

3. **MCPM** メッセージの受信機能を有効にします。

   次のディレクティブを追加して、**web** サーバーがワーカーノードから **MCPM** メッセージを取得できるようにします。

```
EnableMCPMReceive
```

4. **Web** サーバーを再起動します。

   以下のいずれかを実行して **Web** サーバーを再起動します。実行するコマンドは、**Red Hat Enterprise Linux** または **Microsoft Windows Server** を使用しているかによって異なります。

   - **Red Hat Enterprise Linux**

```
[root@host ]# service httpd restart
```

   - **Microsoft Windows Server**

```
net stop Apache2.2
net start Apache2.2
```

結果

Web サーバーは **mod_cluster** プロキシーの **IP** アドレスとポートをアドバタイズしなくなりました。繰り返すには、ワーカーノードが静的アドレスおよびポートを使用してプロキシーと通信するように設定する必要があります。詳細は、「**mod_cluster** ワーカーノードの設定」 を参照してください。

バグの報告

17.6.6. mod_cluster ワーカーノードの設定

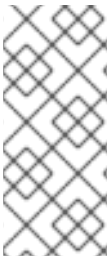
概要

mod_cluster ワーカーノードは、**JBoss EAP 6** サーバーで構成されます。このサーバーは、管理対象ドメインまたはスタンドアロンサーバーのサーバーグループの一部にすることができます。個別のプロセスは **JBoss EAP 6** 内で実行され、クラスターのワーカーノードをすべて管理します。これはマスターと呼ばれます。ノードの概念の詳細については、「ノードのタイプ」 を参照してください。**Web** サーバーの負荷分散の概要は、「**HTTP** コネクターの概要」 を参照してください。

管理対象ドメインのワーカーノードは、サーバーグループ全体で同じ設定を共有します。スタンドアロンサーバーとして実行しているワーカーノードは個別に設定されます。設定手順は同じです。

ワーカーノード設定

- スタンドアロンサーバーは **standalone-ha** または **standalone-full-ha** プロファイルで起動する必要があります。
- 管理対象ドメインのサーバーグループは、**ha** または **full-ha** プロファイルを使用し、**ha-sockets** または **full-ha-sockets** ソケットバインディンググループを使用する必要があります。**JBoss EAP 6** には、これらの要件を満たす **other-server-group** という名前のクラスター対応サーバーグループが同梱されます。



注記

管理 **CLI** コマンドを指定すると、管理対象ドメインが使用されることを前提としています。スタンドアロンサーバーを使用する場合は、コマンドの **/profile=full-ha** の部分を削除します。

手順**17.14** ワーカーノードの設定

1. ネットワークインターフェースの設定

   デフォルトでは、ネットワークインターフェースがすべて **127.0.0.1** に設定されます。スタンドアロンサーバーまたはサーバーグループ内の **1** つまたは複数のサーバーをホストする各物理ホストでは、インターフェースが他のサーバーが見つけることができるパブリック **IP** アドレスを使用するよう設定する必要があります。

   JBoss EAP 6 ホストの **IP** アドレスを変更するには、ホストをシャットダウンし、設定ファイルを直接編集する必要があります。これは、管理コンソールと管理 **CLI** を駆動する管理 **API** が安定した管理アドレスに依存するためです。

*クラスター内の各サーバーの **IP** アドレスをマスターのパブリック **IP** アドレスに変更するには、次の手順を実行します。*

a.

*本トピックでこれまでに説明したプロファイルを使用して **JBoss EAP** サーバーを起動します。*

b.

***Linux** の **EAP_HOME/bin/jboss-cli.sh** コマンド、**Microsoft Windows Server** では **EAP_HOME\bin\jboss-cli.bat** コマンドを使用して、管理 **CLI** を起動します。**localhost** 上のドメインコントローラーに 接続 するか、 **IP_ADDRESS**に接続 してリモートサーバーのドメインコントローラーに接続する場合は **connect** と入力します。*

c.

*以下のコマンドを入力して、管理、パブリック、および セキュアでない インターフェースの外部 **IP** アドレスを変更します。コマンドの **EXTERNAL_IP_ADDRESS** はホストの実際の外部 **IP** アドレスに置き換えてください。*

```
/interface=management:write-attribute(name=inet-
address,value="${jboss.bind.address.management:EXTERNAL_IP_ADDRESS}"
/interface=public:write-attribute(name=inet-
address,value="${jboss.bind.address.public:EXTERNAL_IP_ADDRESS}"
/interface=unsecure:write-attribute(name=inet-
address,value="${jboss.bind.address.unsecure:EXTERNAL_IP_ADDRESS}"
reload
```

各コマンドに対して以下のような結果が表示されるはずです。

```
"outcome" => "success"
```

d.

*管理対象ドメインに参加し、マスターではないホストの場合、ホスト名を **master** から一意の名前に変更する必要があります。この名前はスレーブ全体で一意になる必要があり、スレーブがクラスターを識別するために使用されるため、使用する名前を覚えておくようにしてください。*

i.

*以下の構文を使用して、**JBoss EAP** スレーブホストを起動します。*

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

以下に例を示します。

```
bin/domain.sh --host-config=host-slave01.xml
```

ii. *管理 **CLI** を起動します。*

iii. *以下の構文を使用してホスト名を置き換えます。*

```
/host=master:write-attribute(name="name",value=UNIQUE_HOST_SLAVE_NAME)
```

以下に例を示します。

```
/host=master:write-attribute(name="name",value="host-slave01")
```

以下の結果が表示されるはずです。

```
 "outcome" => "success"
```

*これにより、**host-slave01.xml** ファイルの **XML** が以下のように変更されます。*

```
<host name="host-slave01" xmlns="urn:jboss:domain:1.6">
```

e. *新たに設定されたホストが管理対象ドメインに参加する必要がある場合は、**local** 要素を削除し、ドメインコントローラーを示す **remote** 要素 **host** 属性を追加する必要があります。このステップはスタンドアロンサーバーには適用されません。*

i. *以下の構文を使用して、**JBoss EAP** スレーブホストを起動します。*

```
bin/domain.sh --host-config=HOST_SLAVE_XML_FILE_NAME
```

以下に例を示します。

```
bin/domain.sh --host-config=host-slave01.xml
```

ii.

*管理 **CLI** を起動します。*

iii.

次の構文を使用してドメインコントローラーを指定します。

```
/host=UNIQUE_HOST_SLAVE_NAME/:write-remote-domain-
controller(host=DOMAIN_CONTROLLER_IP_ADDRESS,port=${jboss.domain.master
.port:9999},security-realm="ManagementRealm")
```

以下に例を示します。

```
/host=host-slave01/:write-remote-domain-
controller(host="192.168.1.200",port=${jboss.domain.master.port:9999},security-
realm="ManagementRealm")
```

以下の結果が表示されるはずです。

```
"outcome" => "success"
```

*これにより、**host-slave01.xml** ファイルの **XML** が以下のように変更されます。*

```
<domain-controller>
   <remote host="192.168.1.200" port="${jboss.domain.master.port:9999}"
security-realm="ManagementRealm"/>
</domain-controller>
```

2. *各スレーブサーバーの認証を設定します。*

*各スレーブサーバーには、ドメインコントローラーまたはスタンドアロンマスターの **ManagementRealm** で作成されたユーザー名とパスワードが必要です。ドメインコントローラーまたはスタンドアロンマスターで、**EAP_HOME/bin/add-user.sh** コマンドを実行します。スレーブと同じユーザー名を持つユーザーを **ManagementRealm** に追加します。このユーザーが外部 **JBoss EAP 6** インスタンスに対して認証する必要があるかどうかを尋ねられたら、**yes** と回答します。以下は、**slave1** と呼ばれるスレーブの **password changeme** と出力の例です。*

```
user:bin user$ ./add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
```

```
(a): a

Enter the details of the new user to add.
Realm (ManagementRealm) :
Username : slave1
Password : changeme
Re-enter Password : changeme
About to add user 'slave1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave1' to file '/home/user/jboss-eap-6.0/standalone/configuration/mgmt-users.properties'
Added user 'slave1' to file '/home/user/jboss-eap-6.0/domain/configuration/mgmt-users.properties'
Is this new user going to be used for one AS process to connect to another AS process e.g. slave domain controller?
yes/no? yes
To represent the user add the following to the server-identities definition <secret value="Y2hhbmdlbWU=" />
```

3. ***<secret>*** *出力から* ***Base64*** *でエンコードされた* ***add-user.sh*** *要素をコピーします。*

    認証に ***Base64*** *でエンコードされたパスワード値を指定する予定がある場合は、以下の手順で必要なように、* ***<secret>*** *出力の最後の行から* ***add-user.sh*** *要素の値をコピーします。*

4. *新しい認証を使用するようスレーブホストのセキュリティーレルムを変更します。*

    以下の方法の ***1*** *つを用いて秘密の値を指定できます。*

    - *管理* ***CLI*** *を使用して、サーバー設定ファイルに* ***Base64*** *でエンコードされたパスワード値を指定します。*

        a.

        Linux *の* ***EAP_HOME/bin/jboss-cli.sh*** *コマンド、* ***Microsoft Windows Server*** *では* ***EAP_HOME\bin\jboss-cli.bat*** *コマンドを使用して、管理* ***CLI*** *を起動します。* ***localhost*** *上のドメインコントローラーに 接続 するか、* ***IP_ADDRESS****に接続 してリモートサーバーのドメインコントローラーに接続する場合は* ***connect*** *と入力します。*

        b.

        以下のコマンドを入力して秘密の値を指定します。 ***SECRET_VALUE*** *は、直前の手順の* ***add-user*** *出力から返された秘密の値に置き換えてください。*

        ```
        /host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="SECRET_VALUE")
        reload --host=master
        ```

        各コマンドに対して以下のような結果が表示されるはずです。

        ```
        "outcome" => "success"
        ```

- ホストを設定し、**vault** よりパスワードを取得します。

a.

vault.sh スクリプトを使用してマスクされたパスワードを生成します。**VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNlLWE4ODMtZjQ1NWNmNDU4ZDc1TElORV9CUkVBS3ZhdWx0** のような文字列が生成されます。

パスワード **vault** の詳細は、セキュリティー『**アーキテクチャー**』およびその他の **JBoss EAP** セキュリティードキュメントを参照してください。

b.

Linux の **EAP_HOME/bin/jboss-cli.sh** コマンド、**Microsoft Windows Server** では **EAP_HOME\bin\jboss-cli.bat** コマンドを使用して、管理 **CLI** を起動します。**localhost** 上のドメインコントローラーに 接続 するか、 **IP_ADDRESS**に接続 してリモートサーバーのドメインコントローラーに接続する場合は **connect** と入力します。

c.

以下のコマンドを入力して秘密の値を指定します。**SECRET_VALUE** は、必ず前のステップで生成したマスクされたパスワードに置き換えてください。

```
/host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="${VAULT::secret::password::SECRET_VALUE}")
reload --host=master
```

各コマンドに対して以下のような結果が表示されるはずです。

```
"outcome" => "success"
```

注記

vault でパスワードを作成する場合、**Base64** エンコードではなくプレーンテキストで指定する必要があります。

- システムプロパティーとしてパスワードを指定します。

  次の例は、**server.identity.password** をパスワードのシステムプロパティー名として使用します。

  a.

  管理 **CLI** を使用してサーバー設定ファイルにパスワードのシステムプロパティーを指定します。

  i.

  Linux の **EAP_HOME/bin/jboss-cli.sh** コマンド、**Microsoft Windows Server** では **EAP_HOME\bin\jboss-cli.bat** コマンドを使用して、管理 **CLI** を起動します。**localhost** 上のドメインコントローラーに 接続 するか、 **IP_ADDRESS**に 接続 してリモートサーバーのドメインコントローラーに接続する場合は **connect** と入力します。

  ii.

  以下のコマンドを入力し、システムプロパティーを使用する秘密のアイデンティティーを設定します。

  ```
  /host=master/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="${server.identity.password}")
  reload --host=master
  ```

  各コマンドに対して以下のような結果が表示されます。

  ```
  "outcome" => "success"
  ```

  b.

  システムプロパティーとしてパスワードを指定する場合、次の方法のいずれかを用いてホストを設定できます。

  ■

  次の例のように、プレーンテキストのパスワードをコマンドライン引数として入力し、サーバーを起動します。

  ```
  -Dserver.identity.password=changeme
  ```

注記

*パスワードはプレーンテキストで入力する必要があります。**ps -ef** コマンドを実行すると、パスワードを確認できます。*

- *パスワードをプロパティーファイルに格納し、プロパティーファイルの **URL** をコマンドライン引数として渡します。*

  i. *キーと値のペアをプロパティーファイルに追加します。以下に例を示します。*

  ```
  server.identity.password=changeme
  ```

  ii. *コマンドライン引数を用いてサーバーを起動します。*

  ```
  --properties=URL_TO_PROPERTIES_FILE
  ```

  .

5. *サーバーを再起動します。*

ホスト名をユーザー名として使用し、暗号化された文字列をパスワードとして使用してスレーブがマスターに対して認証されます。

結果

スタンドアロンサーバーまたは管理対象ドメインのサーバーグループ内のサーバーが **mod_cluster** ワーカーノードとして設定されます。クラスター化されたアプリケーションをデプロイする場合、セッションはフェイルオーバーのためにすべてのクラスターノードに複製され、外部 **Web** サーバーまたはロードバランサーからのリクエストを許可できます。クラスターの各ノードは、デフォルトで自動検出を使用して他のノードを検出します。自動検出や **mod_cluster** サブシステムのその他の固有の設定を設定するには、「**mod_cluster** サブシステムの設定」 を参照してください。**Apache HTTP Server** を設定するには、「外部 **Web** サーバーを **JBoss EAP 6** アプリケーションの **Web** フロントエンドとして使用」 を参照してください。

バグの報告

17.6.7. クラスター間のトラフィックの移行

概要

JBoss EAP 6 を使用して新しいクラスターを作成した後、アップグレードプロセスの一部として以前のクラスターから新しいクラスターへトラフィックを移行できます。ここでは、停止時間やダウンライムを最小限に抑えてトラフィックを移行する方法について説明します。

前提条件

- 新しいクラスターの設定：「**mod_cluster** サブシステムの設定」（このクラスター **ClusterNEW**）を呼び出します。
- 不要となった古いクラスターの設定（ここでは **ClusterOLD** と呼びます）。

手順**17.15** クラスターのアップグレードプロセス **-** ロードバランシンググループ

1. 前提条件に従って、新しいクラスターを設定します。
2. **ClusterNEW** および **ClusterOLD** の両方で、設定オプション **sticky-session** を **true** に設定するようにしてください（このオプションはデフォルトで **true** に設定されます）。このオプションを有効にすると、クラスターのクラスターノードへの新しいリクエストはすべてそのクラスターノードに送信されます。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)
```

3.
ClusterOLD のすべてのクラスターノードが **ClusterOLD** ロードバランシンググループのメンバーであることを仮定します。この設定は、**CLI** または **xml** 設定（ドメインモードでは **ha** または **full-ha** プロファイル、スタンドアロンモードの場合は **standalone-ha** または **standalone-full-ha** のいずれか）で設定できます。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=ClusterOLD)
```

```
<subsystem xmlns="urn:jboss:domain:modcluster:1.2">
 <mod-cluster-config load-balancing-group="ClusterOLD" advertise-socket="modcluster" connector="ajp">
  <dynamic-load-provider>
   <load-metric type="busyness"/>
  </dynamic-load-provider>
    </mod-cluster-config>
  </subsystem>
```

4.
「**mod_cluster** ワーカーノードの設定」のプロセスを使用して、**ClusterNEW** のノードを個別に **mod_cluster** 設定に追加します。さらに、この手順を使用してロードバランシンググループを **ClusterNEW** に設定します。

この時点で、**mod_cluster-manager** コンソールに、以下に短い例のような出力が表示されます。

```
mod_cluster/<version>

LBGroup ClusterOLD: [Enable Nodes]  [Disable Nodes]  [Stop Nodes]
   Node node-1-jvmroute (ajp://node1.oldcluster.example:8009):
      [Enable Contexts]  [Disable Contexts]  [Stop Contexts]
      Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
      Virtual Host 1:
         Contexts:
            /my-deployed-application-context, Status: ENABLED Request: 0 [Disable] [Stop]
```

```
        Node node-2-jvmroute (ajp://node2.oldcluster.example:8009):
           [Enable Contexts]   [Disable Contexts]   [Stop Contexts]
           Balancer: qacluster, LBGroup: ClusterOLD, Flushpackets: Off, ..., Load: 100
           Virtual Host 1:
              Contexts:
                 /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]


     LBGroup ClusterNEW: [Enable Nodes]   [Disable Nodes]   [Stop Nodes]
        Node node-3-jvmroute (ajp://node3.newcluster.example:8009):
           [Enable Contexts]   [Disable Contexts]   [Stop Contexts]
           Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
           Virtual Host 1:
              Contexts:
                 /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]

        Node node-4-jvmroute (ajp://node4.newcluster.example:8009):
           [Enable Contexts]   [Disable Contexts]   [Stop Contexts]
           Balancer: qacluster, LBGroup: ClusterNEW, Flushpackets: Off, ..., Load: 100
           Virtual Host 1:
              Contexts:
                 /my-deployed-application-context, Status: ENABLED Request: 0 [Disable]
[Stop]
```

5.

ClusterOLD グループ内に古いアクティブなセッションがあり、新しいセッションは ClusterOLD または CLusterNEW グループ内に作成されます。次に、現在アクティブなクライアントのセッションにエラーが発生しないようにクラスターノードの電源を切ることで、ClusterOLD グループ全体を無効にします。

mod_cluster-manager Web コンソールの LBGroup ClusterOLD の [Disable Nodes] リンクをクリックします。

この時点で、すでに確立されたセッションに属するリクエストのみが ClusterOLD ロードバランシンググループのメンバーにルーティングされます。新しいクライアントのセッションは ClusterNEW グループのみに作成されます。ClusterOLD グループ内にアクティブなセッションがなくなったら、そのメンバーを安全に削除できます。

注記

***[Stop Nodes]** を使用すると、即座にこのドメインへの要求のルーティングを停止するようロードバランサーコマンドが実行されます。これにより、別の負荷分散グループにフェイルオーバーが強制され、**ClusterNEW** と **ClusterOLD** の間にセッションレプリケーションがない場合は、クライアントへのセッションデータの損失が発生します。*

デフォルトの負荷分散グループ

*現在の **ClusterOLD** 設定に負荷分散グループ設定が含まれていない場合には（**mod_cluster-manager** コンソールで **LBGroup:** を参照できる）、**ClusterOLD** ノードの無効化を利用できます。この場合、各 **Cluster OLD** ノードの **[Disable Contexts]** をクリックします。これらのノードのコンテンツは無効になり、アクティブなセッションがないと、削除の準備が整います。新しいクライアントのセッションは、有効なコンテキストを持つノードにのみ作成されます（この例では **Cluster NEW** メンバー）。*

***JBoss EAP CLI** の使用*

***mod_cluster-manager Web** コンソールを使用できる可能性に加え、特定のコンテキストを無効にするには **CLI** を使用できます。前述の操作は **stop-context** と呼ばれますが、クラスターノードは **DISABLE-APP** コマンドをロードバランサーに送信し、**mod_cluster-manager** コンソールの特定のコンテキストの横にある **[Disable]** リンクをクリックするのと全く同じ効果を持つことができます（例：**default-host** が前述の **mod_cluster-manager** コンソール出力例から削除されていることに注意してください）。*

```
/profile=full-ha/subsystem=modcluster/:stop-context(context=/my-deployed-application-context, virtualhost=default-host, waittime=50)
```

まとめ

特定のコンテキストを停止する場合、クラスターノードまたは負荷分散グループ全体を停止するには、バランサーがすぐにリクエストのルーティングを停止するため、別の利用可能なコンテキストにフェイルオーバーを強制的に実行します。特定のコンテキストを無効にするには、クラスターノードまたは負荷分散グループ全体に対して、この特定のコンテキスト/ノード/負荷分散グループで新規セッションを遮断しないように指示します。

結果

***JBoss EAP 6** のクラスターが正常にアップグレードされます。*

バグの報告

17.6.8. mod_cluster の fail_on_status パラメーターの設定

fail_on_status パラメーターは、クラスターのワーカーノードによって返されるとそのノードが失敗したことを示す **HTTP** ステータスコードを一覧表示します。ロードバランサーはその後のリクエストをクラスターの別のワーカーノードに送信します。失敗したワーカーノードは、ロードバランサーを **NOTOK** メッセージを送信するまで **STATUS** 状態のままになります。

注記

fail_on_status パラメーターは、この機能をサポートしないため、**Hewlett-Packard** の **HP-UX v11.3 hpws httpd B.2.2.15.15** と使用できません。

fail_on_status パラメーターは、ロードバランサーの **httpd** 設定ファイルで設定する必要があります。**fail_on_status** の複数の **HTTP** ステータスコードは、コンマ区切りのリストとして指定できます。以下の例では、**203** に **HTTP** ステータスコード **204** および **fail_on_status** を指定します。

例**17.6 fail_on_status Configuration**（設定）

```
ProxyPass / balancer://MyBalancer stickysession=JSESSIONID|jsessionid nofailover=on
failonstatus=203,204
ProxyPassReverse / balancer://MyBalancer
ProxyPreserveHost on
```

バグの報告

17.7. APACHE MOD_JK

17.7.1. Apache mod_jk HTTP コネクター

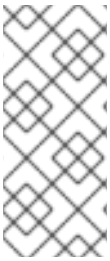
Apache mod_jk は、互換性を確保するために必要なお客様に提供される **HTTP** コネクターです。これは負荷分散機能を提供し、**Red Hat JBoss Enterprise Application Platform 6** のネイティブパッケージの一部です。**X.0 Webserver Connector Natives**（**zip** のインストール）は、**Red Hat** カスタマーポータル(**https://access.redhat.com**)から入手できます。**mod_jk RPM** からインストールできます。**RPM** からのインストールの詳細は、「**Apache HTTP Server** への **mod_jk** モジュールのインストール**(RPM)**」を参照してください。サポートされるプラットフォームについては、を参照してください **https://access.redhat.com/articles/111663**。**mod_jk** コネクターは **Apache** によって維持され、そのドキュメントは にあり **http://tomcat.apache.org/connectors-doc/** ます。

JBoss EAP 6 は **Apache HTTP** プロキシーサーバーからのワークロードを許可します。プロキシーサーバーは **Web** フロントエンドからのクライアントリクエストを受け入れ、ワークを参加する **JBoss**

EAP 6 サーバーに渡します。スティッキーセッションが有効な場合、同じクライアントリクエストは常に同じ JBoss EAP 6 サーバーに送信されます（サーバーが使用できない場合を除く）。

JBoss mod_cluster HTTP コネクターとは異なり、Apache mod_jk HTTP コネクターはサーバーまたはサーバーグループのデプロイメントの状態を認識せず、ワークの送信先に順応できません。

mod_jk AJP 1.3 プロトコルを介して通信します。mod_cluster 他のプロトコルをサポートします。詳細は、「HTTP コネクターの概要」 の 表 HTTP コネクター機能および制約 を参照してください。



注記

mod_cluster mod_jk よりも高度なロードバランサーです。mod_cluster mod_jk および追加機能のすべて機能を提供します。mod_cluster の詳細は、「mod_cluster HTTP コネクター」 を参照してください。

次の手順: JBoss EAP 6 インスタンスを設定し mod_jk ロードバランシンググループに参加します。

- *「外部 Web サーバーからの要求を許可するよう JBoss EAP 6 を設定」*
- *「Apache HTTP Server への mod_jk モジュールのインストール (ZIP)」*

バグの報告

17.7.2. JBoss EAP 6 が Apache mod_jk と通信するよう設定

概要

mod_jk HTTP コネクターには、Web サーバーによってロードされる mod_jk.so モジュールの単一のコンポーネントがあります。このモジュールはクライアント要求を受け取り、コンテナー（この場合は JBoss EAP 6）に転送します。これらのリクエストを許可し、応答を Web サーバーに送信するよう JBoss EAP 6 を設定する必要があります。

Apache HTTP サーバーの設定については、「Apache HTTP Server への mod_jk モジュールのインストール (ZIP)」 を参照してください。

JBoss EAP 6 が Apache HTTP サーバーと通信できるようにするには、AJP/1.3 コネクターを有効にする必要があります。このコネクターはデフォルトで以下の設定に存在します。

- 管理対象ドメインでは、**ha** および **full-ha** プロファイルを使用するサーバーグループおよび **ha** または **full-ha** ソケットバインディンググループを使用するサーバーグループ。**other-server-group** サーバーグループがデフォルトのインストールで正しく設定されている。

- スタンドアロンサーバーでは、**standalone-ha** プロファイルおよび **standalone-full-ha** プロファイルは、クラスター化の設定に対して設定されます。これらのプロファイルの **1** つでスタンドアロンサーバーを起動するには、**EAP_HOME**/ ディレクトリーから以下のコマンドを実行します。適切なプロファイル名を置き換えます。

```
EAP_HOME/bin/standalone.sh --server-config=standalone-ha.xml
```

Windows の場合は、以下のコマンドを入力します。

```
EAP_HOME\bin\standalone.bat --server-config=standalone-ha.xml
```

バグの報告

17.7.3. Apache HTTP Server への mod_jk モジュールのインストール (ZIP)

前提条件

- このタスクを実行するには、サポートされる環境にインストールされた **Apache HTTP Server** を使用するか、**JBoss Enterprise Web Server** からインストールした **Apache HTTP Server** を使用する必要があります。**JBoss Enterprise Web Server** は **JBoss EAP 6** ディストリビューションの一部であることに注意してください。

- **Red Hat Enterprise Linux** ネイティブ **Apache HTTP Server** をインストールする必要がある場合は、『**Red Hat Enterprise Linux デプロイメントガイド**』の説明を使用します。

- **HP-UX** ネイティブ **Apache HTTP Server** をインストールする必要がある **https://h20392.www2.hp.com/portal/swdepot/displayInstallInfo.do?productNumber=HPUXWSATW232** 場合は、にある『**HP-UX Web Server Suite Installation Guide**』の手順を使用します。

- **JBoss Enterprise Web Server** をインストールする必要がある場合は、『**JBoss Enterprise Web Server『インストールガイド』を参照してください**』。

- **Apache HTTP Server** を使用している場合は、ご使用のプラットフォーム用の **JBoss**

EAP 6 ネイティブコンポーネントパッケージを Red Hat カスタマーポータルから https://access.redhat.com からダウンロードします。このパッケージには、mod_jk および mod_cluster 事前にコンパイルされたバイナリーの両方が含まれます。JBoss Enterprise Web Server を使用している場合は、すでに mod_jk のバイナリーが含まれています。

- *Red Hat Enterprise Linux (RHEL) 5 と Apache HTTP サーバー (httpd 2.2.3) を使用している場合は、mod_jk モジュールをロードする前に mod_perl モジュールをロードしてください。*

- *管理 (root) 権限を使用してログインする必要があります。*

- *HTTPD 変数規則を表示するには、を参照してください。 「HTTPD 変数規則」*

手順17.16 mod_jk モジュールのインストール

1. *mod_jk モジュールを設定します。*

   a. *HTTPD_HOME/conf.d/mod-jk.conf という新しいファイルを作成し、以下を追加します。*

JKMOUNT ディレクティブ

JkMount ディレクティブは、Apache HTTP Server が mod_jk モジュールに転送する必要がある URL を指定します。ディレクティブの設定に基づいて、mod_jk は受信した URL を正しいワーカーに送信します。

静的コンテンツを直接提供し、Java アプリケーションのロードバランサーのみを使用するには、URL パスは /application/ である必要があります。mod_jk をロードバランサーとして使用するには、値 /* を使用してすべての URL を mod_jk に転送します。*

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
```

```
JkLogLevel info

# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
# The default setting only sends Java application data to mod_jk.
# Use the commented-out line to send all URLs through mod_jk.
# JkMount /* loadbalancer
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
JkMount status
Order deny,allow
Deny from all
Allow from 127.0.0.1
</Location>
```

値を確認し、セットアップに適切であることを確認します。問題がなければ、ファイルを保存します。

b. ***JKMountFile** ディレクティブの指定*

***mod-jk.conf** の **JKMount** ディレクティブの他に、**mod_jk** に転送される複数の **URL** パターンを含むファイルを指定できます。*

i. *以下を **HTTPD_HOME/conf/mod-jk.conf** ファイルに追加します。*

```
# You can use external file for mount points.
# It will be checked for updates each 60 seconds.
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties
```

ii. *各 **URL** パターンが一致する行を指定して **HTTPD_CONF/uriworkermap.properties** という名前の新しいファイルを作成します。*

以下の例は、ファイルの構文例を示しています。

```
# Simple worker configuration file
/*=loadbalancer
```

c. **httpd** のモジュールディレクトリーへの **mod_jk.so** ファイルのコピー

注記

これは、**Apache HTTP** サーバーに **modules/** ディレクトリーに **mod_jk.so** がない場合にのみ必要です。**JBoss EAP 6** のダウンロードとして含まれている **Apache HTTP** サーバーを使用している場合は、この手順を省略できます。

Native Web Server Connectors Zip パッケージを展開します。オペレーティングシステムが **32** ビットまたは **64** ビットであるかによって、**EAP_HOME/modules/system/layers/base/native/lib/httpd/modules/** ディレクトリーまたは**EAP_HOME/modules/layers/base/native/lib64/httpd/modules/** ディレクトリーのいずれかで **mod_jk.so** ファイルを見つけます。

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

2. **mod_jk** ワーカーノードを設定します。

a.

HTTPD_CONF/workers.properties という新しいファイルを作成します。以下の例を開始点として使用し、必要に応じてファイルを変更します。

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1
```

```
# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

workers.properties ファイルの構文および高度な設定オプションの詳細は、「**Apache mod_jk** ワーカーの設定リファレンス」 を参照してください。

3. **Web** サーバーを再起動します。

Web サーバーの再起動方法は、**Red Hat Enterprise Linux** の **Apache HTTP** サーバーを使用するか、**JBoss Enterprise Web Server** に含まれる **Apache HTTP Server** を使用しているかによって異なります。以下の方法のいずれかを選択します。

- **Red Hat Enterprise Linux** の **Apache HTTP Server**

以下のコマンドを実行します。

```
[root@host]# service httpd restart
```

- **JBoss Enterprise Web Server** の **Apache HTTP Server**

JBoss Enterprise Web Server は、**Red Hat Enterprise Linux** と **Microsoft Windows Server** の両方で実行されます。**Web** サーバーの再起動方法はそれぞれ異なります。

  - **RPM** からインストールされた **Red Hat Enterprise Linux**

Red Hat Enterprise Linux では、**JBoss Enterprise Web Server** は **Web** サーバーをサービスとしてインストールします。**Web** サーバーを再起動するには、以下の **2** つのコマンドを実行します。

```
[root@host ~]# service httpd stop
[root@host ~]# service httpd start
```

- **Zip からインストールされた Red Hat Enterprise Linux**

  Zip アーカイブから **JBoss Enterprise Web Server Apache HTTP Server** をインストールしている場合は、**apachectl** コマンドを使用して **Web** サーバーを再起動します。**mingw _HOME** を、**JBoss Enterprise Web Server Apache HTTP Server** を展開したディレクトリーに置き換えます。

```
[root@host ~]# EWS_HOME/httpd/sbin/apachectl restart
```

- **Microsoft Windows Server**

  コマンドプロンプトで以下のコマンドを管理権限で実行します。

```
net stop Apache2.2
net start Apache2.2
```

- **Solaris**

  コマンドプロンプトで以下のコマンドを管理権限で実行します。**mingw _HOME** を、**JBoss Enterprise Web Server Apache HTTP Server** を展開したディレクトリーに置き換えます。

```
[root@host ~] EWS_HOME/httpd/sbin/apachectl restart
```

結果

Apache HTTP サーバーが **mod_jk** ロードバランサーを使用するよう設定されました。**JBoss EAP 6** が **mod_jk** を認識するよう設定するには、*「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」* を参照してください。

バグの報告

17.7.4. Apache HTTP Server への mod_jk モジュールのインストール(RPM)

前提条件

- このタスクを実行するには、**Red Hat Enterprise Linux 6** または **JBoss Enterprise Web Server** にインストールされた **Apache HTTP Server** を使用するか、**JBoss EAP 6** のダウンロード可能な個別コンポーネントとして含まれるスタンドアロン **Apache HTTP Server** を使用する必要があります。
- **jbappplatform-6-ARCHITECTURE-server-RHEL_VERSION-rpm** チャンネルへのアクティブなサブスクリプションが必要です。
- **Red Hat Enterprise Linux 6** に **Apache HTTP Server** をインストールする必要がある場合は、『**Red Hat Enterprise Linux 6 デプロイメントガイド**』に記載された手順を実行します。
- **JBoss EAP 6** の個別のダウンロード可能なコンポーネントとして含まれるスタンドアロンの **Apache HTTP Server** をインストールする必要がある場合は、「**Red Hat Enterprise Linux 5、6、7 への Apache HTTP Server のインストール(Zip)**」を参照してください。
- **JBoss Enterprise Web Server** をインストールする必要がある場合は、『**JBoss Enterprise Web Server インストールガイド**』に記載された手順を実行します。
- 管理 **(root)** 権限を使用してログインする必要があります。

手順**17.17 Red Hat Enterprise Linux 5: mod_jk と Apache HTTP Server 2.2.3**

1. **mod_jk-ap22 1.2.37** と、**jbappplatform-6-ARCHITECTURE-server-5-rpm** チャンネルから依存関係 **mod_perl** をインストールします。

```
yum install mod_jk
```

2. 必要に応じて、使用する設定ファイルのサンプルをコピーします。

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

これらのファイルは、必要に応じて編集する必要があります。

3. サーバーを起動します。

```
service httpd start
```

注記

以下のエラーメッセージは、**mod_perl** が存在する前に **mod_jk** モジュールがロードされたことを意味しています。

```
Cannot load /etc/httpd/modules/mod_jk.so into server:
/etc/httpd/modules/mod_jk.so: undefined symbol: ap_get_server_description
```

mod_perl モジュールが **mod_jk** モジュールよりも先に読み込まれるようにするには、以下を **/etc/httpd/conf.d/mod_jk.conf** に追加します。

```
<IfModule !perl_module>
    LoadModule perl_module modules/mod_perl.so
</IfModule>
LoadModule jk_module modules/mod_jk.so
```

手順17.18 Red Hat Enterprise Linux 5: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. 以下のコマンドを使用して、**jbappplatform-6-ARCHITECTURE-server-5-rpm** チャンネルで提供される **mod_jk** と最新の **Apache HTTP Server 2.2.26** をインストールします。

```
yum install mod_jk httpd
```

2. 必要に 応じ て、使用する設定ファイルのサンプルをコピーします。

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

これらのファイルは、必要に応じて編集する必要があります。

3. サーバーを起動します。

```
service httpd start
```

手順17.19 Red Hat Enterprise Linux 6: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. ***jbappplatform-6-ARCHITECTURE-server-6-rpm*** チャンネルから ***mod_jk-ap22 1.2.37*** および ***Apache HTTP Server 2.2.26 httpd*** パッケージをインストールします（既存のバージョンはすべて更新されます）。

```
yum install mod_jk httpd
```

2. 必要に応じて、使用する設定ファイルのサンプルをコピーします。

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

これらのファイルは、必要に応じて編集する必要があります。

3. サーバーを起動します。

```
service httpd start
```

手順17.20 Red Hat Enterprise Linux 6: mod_jk と Apache HTTP Server 2.2.15

1. 以下のコマンドを実行して、***mod_jk*** と ***Apache HTTP Server 2.2.15*** をインストールします。

```
yum install mod_jk
```

2. 必要に応じて、使用する設定ファイルのサンプルをコピーします。

-

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd/conf/workers.properties
```

これらのファイルは、必要に応じて編集する必要があります。

3. サーバーを起動します。

```
service httpd start
```

手順17.21 Red Hat Enterprise Linux 7: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. **jbappplatform-6-ARCHITECTURE-server-6-rpm** チャンネルから **mod_jk-ap22 1.2.37** および **Apache HTTP Server 2.2.26 httpd22** パッケージをインストールします（既存のバージョンはすべて更新されます）。

```
yum install mod_jk
```

2. 必要に応じて、使用する設定ファイルのサンプルをコピーします。

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/mod_jk.conf.sample
/etc/httpd22/conf.d/mod_jk.conf
```

```
cp /usr/share/doc/mod_jk-ap22-1.2.37/workers.properties.sample
/etc/httpd22/conf/workers.properties
```

これらのファイルは、必要に応じて編集する必要があります。

3. サーバーを起動します。

```
systemctl start httpd22.service
```

バグの報告

17.7.5. Apache mod_jk ワーカーの設定リファレンス

workers.properties ファイルは mod_jk がクライアント要求を渡すワーカーの動作を定義します。Red Hat Enterprise Linux では、ファイルは /etc/httpd/conf/workers.properties にあります。workers.properties ファイルは、異なるアプリケーションサーバーの場所と、負荷をまたがって分散する方法を定義します。

この設定は、3 つのセクションに分かれます。最初のセクションでは、すべてのワーカーに適用されるグローバルプロパティーを処理します。2 番目のセクションには、特定のロードバランサーに適用される設定が含まれます。3 つ目のセクションには、ロードバランサーによって分散される特定のワーカーノードに適用される設定が含まれます。

プロパティーの一般的な構造は worker です。WORKER_NAME はDIRECTIVE です。WORKER_NAME はワーカーの一意の名前で、DIRECTIVE はワーカーに適用される設定です。

Apache mod_jk ロードバランサーの設定リファレンス

テンプレートは、ロードバランサーごとにデフォルトの設定を指定します。ロードバランサーの設定内でテンプレートを上書きできます。ロードバランサーテンプレートの例は、例 17.7「workers.properties ファイルの例」 で確認できます。

表17.15 グローバルプロパティー

プロパティー	説明
worker.list	mod_jk によって使用されるロードバランサー名の一覧。これらのロードバランサーは要求を受信するために使用できます。

表17.16 必須ディレクティブ

プロパティー	説明
type	ロードバランサーのタイプ。デフォルトのタイプは **ajp13** です。他の可能な値は **ajp14**、**lb**、**status** です。 これらのディレクティブの詳細については、の『Apache Tomcat Connector AJP Protocol Reference』を参照 http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html してください。

表17.17 負荷分散ディレクティブ

プロパティー	説明
balance_workers	ロードバランサーが管理する必要があるワーカーノードを指定します。同じロードバランサーにディレクティブを複数回使用できます。カンマ区切りのワーカーノード名のリストで構成されます。これはワーカーノードごとにではなく、ロードバランサーごとに設定されます。

プロパティー	説明
sticky_session	同じセッションからのリクエストを常に同じワーカーにルーティングするかどうかを指定します。デフォルトは **1** で、スティッキーセッションが有効になります。スティッキーセッションを無効にするには **0**. を設定します。すべてのリクエストが実際にステートレスである場合を除き、スティッキーセッションは通常有効にする必要があります。これはワーカーノードごとにではなく、ロードバランサーごとに設定されます。

*表**17.18** 接続ディレクティブ*

プロパティー	説明
host	ロードバランサーのホスト名または IP アドレス。ロードバランサーは **ajp** プロトコルスタックをサポートする必要があります。デフォルト値は **localhost** です。
port	定義されたプロトコル要求をリッスンしているリモートサーバーインスタンスのポート番号。デフォルト値は **8009** です。これは AJP13 ロードバランサーのデフォルトリッスンポートです。AJP14 ロードバランサーのデフォルト値は **8011** です。
ping_mode	ネットワークの状態に対して接続がプローブされる条件。プローブは CPing に空の AJP13 パケットを使用し、応答で CPong を想定します。ディレクティブフラグの組み合わせを使用して条件を指定します。フラグはコンマまたはスペースで区切られません。ping_mode には、**C**、**P**、**I**、および **A** の任意の組み合わせを使用できます。 ● **C** - Connect（接続）。サーバーへの接続後に 1 回接続をプローブします。**connect_timeout** の値を使用してタイムアウトを指定します。それ以外の場合は、**ping_timeout** の値が使用されます。 ● **P** - Prepost（プレポスト）。各リクエストをサーバーに送信する前に接続をプローブします。**prepost_timeout** ディレクティブを使用してタイムアウトを指定します。それ以外の場合は、**ping_timeout** の値が使用されます。 ● **I** - Interval（間隔）。**connection_ping_interval** で指定された間隔で接続をプローブします（ある場合）。それ以外の場合は、**ping_timeout** の値が使用されます。 ● **A** - All（すべて）。すべての接続プローブを使用することを指定する **CPI** のショートカットです。
ping_timeout、connect_timeout、prepost_timeout、connection_ping_interval	上記の接続プローブ設定のタイムアウト値。値はミリ秒単位で指定され、**ping_timeout** のデフォルト値は 10000 です。
lbfactor	個々のロードバランサーの負荷分散係数を指定し、ロードバランサーのメンバーワーカーノードにのみ適用されます。より強力なサーバーにより多くの作業負荷を割り当てる場合に便利です。ワーカーにデフォルトの 3 倍の負荷を割り当てるには、**worker.my_worker.lbfactor=3** のように **3** を設定します。

例17.7 workers.properties ファイルの例

```
worker.balancer1.type=lb
worker.balancer2.type=lb

worker.balancer1.sticky_sessions=1
worker.balancer1.balance_workers=node1
worker.balancer2.sticky_session=1
worker.balancer2.balance_workers=node2,node3

worker.nodetemplate.type=ajp13
worker.nodetemplate.port=8009

worker.node1.template=nodetemplate
worker.node1.host=localhost
worker.node1.ping_mode=CI
worker.node1.connection_ping_interval=9000
worker.node1.lbfactor=1

worker.node2.template=nodetemplate
worker.node2.host=192.168.1.1
worker.node2.ping_mode=A

worker.node3.template=nodetemplate
worker.node3.host=192.168.1.2
```

上記の例では、複数のロードバランサーを使用して Web サーバーの代わりにコンテンツを提供できます。このような設定の理由は以下のとおりです。

- 異なるロードバランサーで異なるコンテキストを提供するには、すべての開発者が同じ Web サーバーを共有し、独自のロードバランサーを所有する開発環境を提供します。
- 異なるプロセスが提供する仮想ホストが異なる場合に、異なる企業に属するサイト間で明確に分離できます。
- 負荷分散を提供するには、独自のマシンで複数のロードバランサーを実行し、それらの間で要求を分割します。

Apache mod_jk の設定の詳細は、本書の範囲外です。詳細は Apache ドキュメントを参照 http://tomcat.apache.org/connectors-doc/ してください。

バグの報告

17.8. APACHE MOD_PROXY

17.8.1. Apache mod_proxy HTTP コネクター

Apache は、httpd に 2 つの異なるプロキシーおよび負荷分散モジュール（mod_proxy と mod_jk ）を提供します。mod_jk の詳細は、「Apache mod_jk HTTP コネクター」 を参照してください。JBoss EAP 6 はこれらのいずれかの HTTP コネクターをサポートしますが、JBoss HTTP コネクターである mod_cluster は JBoss EAP 6 と外部 httpd と密接に連携し、推奨される HTTP コネクターです。サポートされているすべての HTTP コネクターの概要は、「HTTP コネクターの概要」 を参照してください。これには長所や短所が含まれます。

mod_jk とは異なり、mod_proxy は HTTP および HTTPS プロトコルを介した接続をサポートします。それぞれが AJP プロトコルもサポートします。

mod_proxy は、スタンドアロンまたは負荷分散設定で設定でき、スティッキーセッションをサポートします。

mod_proxy モジュールでは、JBoss EAP 6 に HTTP、HTTPS、または AJP Web コネクターが設定されている必要があります。これは Web サブシステムの一部です。Web サブシステムの設定に関する詳細は、「Web サブシステムの設定」 を参照してください。

注記

mod_cluster mod_proxy よりも高度なロードバランサーです。mod_cluster mod_proxy および追加機能のすべて機能を提供します。mod_cluster の詳細は、「mod_cluster HTTP コネクター」 を参照してください。

バグの報告

17.8.2. Apache HTTP Server への mod_proxy HTTP コネクターのインストール

概要

mod_proxy は、Apache が提供する負荷分散モジュールです。このタスクは基本的な設定を示します。詳細設定、または詳細については、Apache の mod_proxy ドキュメント https://httpd.apache.org/docs/2.2/mod/mod_proxy.html を参照してください。JBoss EAP 6 の観点からした mod_proxy の詳細は、「Apache mod_proxy HTTP コネクター」 および 「HTTP コネクターの概要」 を参照してください。

前提条件

- *JBoss Enterprise Web Server から、またはオペレーティングシステムが提供する Apache HTTP サーバーをインストールする必要があります。スタンドアロン Apache HTTP Server は、Red Hat カスタマーポータル（JBoss EAP 6 のダウンロードエリア）の個別ダウンロードとして提供されます。この Apache HTTP サーバーを使用する場合は、「Red Hat Enterprise Linux 5、6、7 への Apache HTTP Server のインストール(Zip)」 を参照してください。*

- *mod_proxy モジュールをインストールする必要があります。Apache HTTP サーバーには通常、すでに含まれている mod_proxy モジュールが同梱されています。これは、Red Hat Enterprise Linux と、JBoss Enterprise Web Server に同梱される Apache HTTP Server の場合です。*

- *Apache HTTP Server 設定を変更するには、root または管理者権限が必要です。*

- *この例では、JBoss EAP 6 が HTTP または HTTPS Web コネクターで設定されていることを前提としています。これは Web サブシステム設定の一部です。Web サブシステムの設定に関する詳細は、「Web サブシステムの設定」 を参照してください。*

1. *httpd で mod_proxy モジュールを有効にします。*

   HTTPD_CONF/httpd.conf ファイルで以下の行を見つけます。存在しない場合は、下部に追加します。行が存在していても、コメント(#)文字で始まる場合は、文字を削除します。その後ファイルを保存します。通常、モジュールはすでに存在し、有効になっています。

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
LoadModule proxy_http_module modules/mod_proxy_http.so
# Uncomment these to proxy FTP or HTTPS
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
```

2. *非負荷分散プロキシーを追加します。*

   以下の設定を、その他の < VirtualHost > ディレクティブのすぐ下にある HTTPD_CONF/httpd.conf ファイルに追加します。値を設定に適切な値に置き換えます。

   この例では、仮想ホストを使用します。デフォルトの httpd 設定を使用するには、次の手順を参照してください。

```
<VirtualHost *:80>
# Your domain name
ServerName Domain_NAME_HERE
```

```
ProxyPreserveHost On

# The IP and port of JBoss EAP 6
# These represent the default values, if your httpd is on the same host
# as your JBoss EAP 6 managed domain or server

ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/

# The location of the HTML files, and access control information
DocumentRoot /var/www
<Directory /var/www>
Options -Indexes
Order allow,deny
Allow from all
</Directory>
</VirtualHost>
```

変更後に、ファイルを保存します。

3. 負荷分散プロキシーを追加します。

mod_proxy をロードバランサーとして使用し、ワークを複数の **JBoss EAP 6** インスタンスに送信するには、以下の設定を **HTTPD_CONF/httpd.conf** ファイルに追加します。**IP** アドレスの例は以下のようになります。ご使用の環境に適切な値に置き換えてください。

```
<Proxy balancer://mycluster>

Order deny,allow
Allow from all

# Add each JBoss Enterprise Application Server by IP address and port.
# If the route values are unique like this, one node will not fail over to the other.
BalancerMember http://192.168.1.1:8080 route=node1
BalancerMember http://192.168.1.2:8180 route=node2
</Proxy>

<VirtualHost *:80>
 # Your domain name
 ServerName YOUR_DOMAIN_NAME

 ProxyPreserveHost On
 ProxyPass / balancer://mycluster/

 # The location of the HTML files, and access control information DocumentRoot /var/www
 <Directory /var/www>
  Options -Indexes
  Order allow,deny
  Allow from all
 </Directory>

</VirtualHost>
```

上記の例はすべて **HTTP** プロトコルを使用して通信します。適切な **mod_proxy** モジュールを読み込む場合は、**AJP** または **HTTPS** プロトコルを使用できます。詳細は **Apache** の **mod_proxy** ドキュメント **http://httpd.apache.org/docs/2.2/mod/mod_proxy.html** を参照してください。

4. スティッキーセッションを有効にします。

スティッキーセッション は、クライアントリクエストが特定の **JBoss EAP 6** ワーカーに送信されると、利用できない場合を除き、今後のすべてのリクエストが同じワーカーに送信されます。これはほとんどの場合正しい動作です。

mod_proxy のスティッキーセッションを有効にするには、**stickysession** パラメーターを **ProxyPass** ステートメントに追加します。この例では、使用できるその他のパラメーターも示します。詳細は、**Apache** の **mod_proxy** ドキュメント(**http://httpd.apache.org/docs/2.2/mod/mod_proxy.html**)を参照してください。

```
ProxyPass /MyApp balancer://mycluster stickysession=JSESSIONID lbmethod=bytraffic
nofailover=Off
```

5. **Web** サーバーを再起動します。

Web サーバーを再起動して変更を有効にします。

結果

標準または負荷分散設定のいずれかで、**mod_proxy** を使用してクライアントリクエストを **JBoss EAP 6** インスタンスに送信するよう **Apache HTTP** サーバーが設定されます。**JBoss EAP 6** がこれらのリクエストに応答するように設定するには、「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」 を参照してください。

バグの報告

17.9. MICROSOFT ISAPI コネクター

17.9.1. インターネットサーバー API(ISAPI)について

Internet Server API(ISAPI) は、**Microsoft** のインターネット情報サービス(IIS)などの **Web** サーバー用の **OLE** サーバー拡張機能やフィルターを書き込むために使用される **API** のセットです。**ISAPI _redirect.dll** は **IIS** 向けに調整された **mod_jk** の拡張機能です。**isapi_redirect.dll** を使用すると、**JBoss EAP 6** インスタンスをワーカーノードとしてロードバランサーとして設定できます。

バグの報告

17.9.2. Microsoft IIS の Web サーバーコネクターネイティブのダウンロードおよび展開

1. Web ブラウザーで、Red Hat カスタマーポータル https://access.redhat.com （）に移動します。

2. Downloads に移動し、Red Hat JBoss Middleware Download Software を選択してから、Product ドロップダウンリストから Enterprise Application Platform を選択します。

3. Version ドロップダウンリストから適切なバージョンを選択します。

4. サーバーのアーキテクチャーに応じて、Red Hat JBoss Enterprise Application Platform < VERSION > Webserver Connector Natives for Windows Server 2008 x86_64 または Red Hat JBoss Enterprise Application Platform < VERSION > Webserver Connector Natives for Windows Server 2008 i686 のいずれかの Download オプションを選択します。

5. Zip ファイルを開き、jboss-eap- <VERSION> /modules/system/layers/base/native/sbin ディレクトリーの内容をサーバーの場所にコピーします。ここでは、内容が C:\connectors\ にコピーされたことを前提とします。

バグの報告

17.9.3. Microsoft IIS が ISAPI コネクターを使用するよう設定

要件:

- 「Microsoft IIS の Web サーバーコネクターネイティブのダウンロードおよび展開」

注記

Microsoft Windows Server および IIS のサポートされる設定の一覧は、を参照 https://access.redhat.com/articles/111663 してください。

手順17.22 IIS マネージャー (IIS 7) を使用した IIS リダイレクターの設定

1. *Start → Run クリックして IIS マネージャーを開き、inetmgr と入力します。*

2. *左側のツリービューペインで IIS 7 を展開します。*

3. *ISAPI and CGI Registrations をダブルクリックして新しいウィンドウで開きます。*

4. *Actions ペインで Add をクリックします。Add ISAPI or CGI Restriction ウインドウが開きます。*

5. *以下の値を指定します。*

   - *ISAPI or CGI Path:c:\connectors\isapi_redirect.dll*
   - *Description:jboss*
   - *Allow extension path to execute: チェックボックスを選択します。*

6. *OK をクリックして Add ISAPI or CGI Restriction ウィンドウを閉じます。*

7. *JBoss ネイティブ仮想ディレクトリーの定義*

   a. *Default Web Site を右クリックし、Add Virtual Directory をクリックします。Add Virtual Directory ウィンドウが開きます。*

   b. *以下の値を指定して仮想ディレクトリーを追加します。*

      - *エイリアス:jboss*
      - *Physical Path:C:\connectors*

c.
OK をクリックして値を保存し、*Add Virtual Directory* ウィンドウを閉じます。

8. *JBoss* ネイティブ *ISAPI* リダイレクトフィルターの定義

a.
ツリービューペインで *Sites* → *Default Web Site* を展開します。

b.
ISAPI Filters をダブルクリックします。*ISAPI Filters Features* ビューが表示されます。

c.
Actions ペインで *Add* をクリックします。*Add ISAPI Filter* ウインドウが表示されます。

d.
Add ISAPI Filter ウィンドウで、以下の値を指定します。

- *Filter name:jboss*
- *Executable:C:\connectors\isapi_redirect.dll*

e.
OK をクリックして値を保存し、*Add ISAPI Filters* ウィンドウを閉じます。

9. *ISAPI-dll* ハンドラーの有効化

a.
ツリービューペインの *IIS 7* をダブルクリックします。*IIS 7 Home Features View* が開きます。

b.
Handler Mappings をダブルクリックします。*Handler Mappings Features View* が表示されます。

c.
Group by コンボボックスで *State* を選択します。*Handler Mappings* が *Enabled and Disabled Groups* に表示されます。

d.
ISAPI-dll を検索します。*Disabled* グループにある場合は右クリックし、*Edit Feature Permissions* を選択します。

e. 以下のパーミッションを有効にします。

- **Read**
- **Script**
- **Execute**

f. **OK** をクリックして値を保存し、**Edit Feature Permissions** ウインドウを閉じます。

結果

Microsoft IIS が **ISAPI** コネクターを使用するよう設定されました。次に、「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」、次に 「**ISAPI** コネクターがクライアントリクエストを **JBoss EAP 6** に送信するよう設定」 または 「**ISAPI** コネクターがクライアントリクエストを複数の **JBoss EAP 6** サーバーで分散するよう設定」 です。

バグの報告

17.9.4. ISAPI コネクターがクライアントリクエストを JBoss EAP 6 に送信するよう設定

概要

このタスクでは、**JBoss EAP 6** サーバーのグループが **ISAPI** コネクターからのリクエストを受け入れるように設定します。ロードバランシングまたは高可用性フェイルオーバーの設定は含まれません。これらの機能が必要な場合は、「**ISAPI** コネクターがクライアントリクエストを複数の **JBoss EAP 6** サーバーで分散するよう設定」 を参照してください。

この設定は **IIS** サーバーで行われ、**JBoss EAP 6** はすでに 「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」 に従って設定されていることを前提とします。

前提条件

- **IIS** サーバーへの完全な管理者アクセスが必要です。

- 「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」

- 「**Microsoft IIS** が **ISAPI** コネクターを使用するよう設定」

手順**17.23** プロパティーファイルの編集およびリダイレクトの設定

1. ログ、プロパティーファイル、およびロックファイルを格納するディレクトリーを作成します。

   この手順では、ディレクトリー **C:\connectors** の使用を前提としています。異なるディレクトリーを使用する場合は、適切に手順を変更してください。

2. ***isapi_redirect.properties*** ファイルを作成します。

   C:\connectors\isapi_redirect.properties という新しいファイルを作成します。このファイルに次の内容をコピーします。

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

   rewrite.properties ファイルを使用しない場合は、行の先頭に **#** 文字を記入して最後の行をコメントアウトします。詳細は、ステップ **5** を参照してください。

3. ***uriworkermap.properties*** ファイルを作成します。

   uriworkermap.properties ファイルには、デプロイされたアプリケーション **URL** と、それらへの要求を処理するワーカー間のマッピングが含まれます。以下のサンプルファイルはファイルの構文を示しています。**uriworkermap.properties** ファイルを **C:\connectors** に配置します。

```
# images and css files for path /status are provided by worker01
```

```
/status=worker01
/images/*=worker01
/css/*=worker01

# Path /web-console is provided by worker02
# IIS (customized) error page is used for http errors with number greater or equal to 400
# css files are provided by worker01
/web-console/*=worker02;use_server_errors=400
/web-console/css/*=worker01

# Example of exclusion from mapping, logo.gif won't be displayed
# /web-console/images/logo.gif=*

# Requests to /app-01 or /app-01/something will be routed to worker01
/app-01|/*=worker01

# Requests to /app-02 or /app-02/something will be routed to worker02
/app-02|/*=worker02
```

4. ***workers.properties*** *ファイルを作成します。*

    workers.properties *ファイルには、ワーカーラベルとサーバーインスタンス間のマッピング定義が含まれます。以下のサンプルファイルはファイルの構文を示しています。このファイルを* ***C:\connectors*** *ディレクトリーに置きます。*

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers

# First JBoss EAP 6 server definition, port 8009 is standard port for AJP in EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second JBoss EAP 6 server definition
worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. ***rewrite.properties*** *ファイルを作成します。*

    rewrite.properties *ファイルには、特定のアプリケーションの単純な* ***URL*** *書き換えルールが含まれます。以下の例で示されているように、書き換えられたパスは名前と値のペアを使用して指定されます。このファイルを* ***C:\connectors*** *ディレクトリーに置きます。*

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
```

6. ***IIS*** *サーバーを再起動します。*

net stop コマンドおよび **net start** コマンドを使用して **IIS** サーバーを再起動します。

```
C:\> net stop was /Y
C:\> net start w3svc
```

結果

アプリケーションごとに、設定した特定の **JBoss EAP 6** サーバーにクライアント要求を送信するよう **IIS** サーバーが設定されます。

バグの報告

17.9.5. ISAPI コネクターがクライアントリクエストを複数の JBoss EAP 6 サーバーで分散するよう設定

概要

この設定は、指定する **JBoss EAP 6** サーバー全体でクライアントリクエストを分散します。デプロイメントごとに特定の **JBoss EAP 6** サーバーにクライアントリクエストを送信する場合は、代わりに **「ISAPI コネクターがクライアントリクエストを JBoss EAP 6 に送信するよう設定」** を参照してください。

この設定は **IIS** サーバーで行われ、**JBoss EAP 6** はすでに **「外部 Web サーバーからの要求を許可するよう JBoss EAP 6 を設定」** に従って設定されていることを前提とします。

前提条件

- **IIS** サーバーへの完全な管理者アクセス。
- **「外部 Web サーバーからの要求を許可するよう JBoss EAP 6 を設定」**
- **「Microsoft IIS が ISAPI コネクターを使用するよう設定」**

手順**17.24** 複数のサーバー間でのクライアント要求の分散

1. ログ、プロパティーファイル、およびロックファイルを格納するディレクトリーを作成します。

   この手順では、ディレクトリー **C:\connectors** の使用を前提としています。異なるディレ

クトリーを使用する場合は、適切に手順を変更してください。

2. **isapi_redirect.properties** ファイルを作成します。

C:\connectors\isapi_redirect.properties という新しいファイルを作成します。このファイルに次の内容をコピーします。

```
# Configuration file for the ISAPI Connector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Connector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
log_level=info

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

rewrite.properties ファイルを使用しない場合は、行の先頭に **#** 文字を記入して最後の行をコメントアウトします。詳細は、ステップ **5** を参照してください。

3. **uriworkermap.properties** ファイルを作成します。

uriworkermap.properties ファイルには、デプロイされたアプリケーション **URL** と、それらへの要求を処理するワーカー間のマッピングが含まれます。以下のサンプルファイルは負荷分散が設定されたファイルの構文を示しています。ワイルドカード**(*)**文字は、さまざまな **URL** サブディレクトリーのすべてのリクエストを **router** というロードバランサーに送信します。ロードバランサーの設定は、ステップ **4** で説明されています。

uriworkermap.properties ファイルを **C:\connectors** に配置します。

```
# images, css files, path /status and /web-console will be
# provided by nodes defined in the load-balancer called "router"
/css/*=router
/images/*=router
/status=router
/web-console|/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
# /web-console/images/logo.gif=*
```

```
# Requests to /app-01 and /app-02 will be routed to nodes defined
# in the load-balancer called "router"
/app-01|/*=router
/app-02|/*=router

# mapping for management console, nodes in cluster can be enabled or disabled here
/jkmanager|/*=status
```

4. ***workers.properties*** *ファイルを作成します。*

    workers.properties *ファイルには、ワーカーラベルとサーバーインスタンス間のマッピング定義が含まれます。以下のサンプルファイルはファイルの構文を示しています。ロードバランサーはファイルの末尾付近に設定され、ワーカー* ***worker01*** *および* ***worker02*** *で構成されます。* ***workers.properties*** *ファイルは、* ***Apache mod_jk*** *設定に使用される同じファイルの構文に従います。* ***workers.properties*** *ファイルの構文に関する詳細は、* *「**Apache mod_jk** ワーカーの設定リファレンス」 を参照してください。*

    このファイルを ***C:\connectors*** *ディレクトリーに置きます。*

```
# The advanced router LB worker
worker.list=router,status

# First EAP server definition, port 8009 is standard port for AJP in EAP
#
# lbfactor defines how much the worker will be used.
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A – all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker for jkmanager
worker.status.type=status
```

5. ***rewrite.properties*** *ファイルを作成します。*

    rewrite.properties *ファイルには、特定のアプリケーションの単純な* ***URL*** *書き換えルールが含まれます。以下の例で示されているように、書き換えられたパスは名前と値のペアを使用して指定されます。このファイルを* ***C:\connectors*** *ディレクトリーに置きます。*

```
#Simple example
# Images are accessible under abc path
/app-01/abc/=/app-01/images/
```

6. ***IIS*** *サーバーを再起動します。*

    net stop *コマンドおよび* ***net start*** *コマンドを使用して* ***IIS*** *サーバーを再起動します。*

```
C:\> net stop was /Y
C:\> net start w3svc
```

結果

IIS *サーバーは、* ***workers.properties*** *ファイルで参照されている* ***JBoss EAP 6*** *サーバーにクライアントリクエストを送信し、サーバー間で負荷を* ***1:3*** *の比率で分散するよう設定されます。この比率は、各サーバーに割り当てられた負荷分散係数(****lbfactor****)から派生します。*

バグの報告

17.10. ORACLE NSAPI コネクター

17.10.1. Netscape Server API(NSAPI)

Netscape Server API(NSAPI) *は、拡張機能をサーバーに実装するために* ***Oracle iPlanet Web Server***（*旧名* ***Netscape Web Server***）*によって提供される* ***API*** *です。これらの拡張機能はサーバープラグインと呼ばれます。この* ***API*** *は、ネイティブユーティリティーパッケージで* ***JBoss EAP 6*** *が提供する* ***nsapi_redirector.so*** *で使用さ れます。このコネクターを設定するには、「****NSAPI*** *コネクターがクライアントリクエストを複数の* ***JBoss EAP 6*** *サーバーで分散するよう設定」* *を参照してください。*

バグの報告

17.10.2. Oracle Solaris での ***NSAPI*** *コネクターの設定*

概要

NSAPI コネクターは、**Oracle iPlanet Web Server** 内で実行されるモジュールです。

前提条件

- サーバーが、**Intel 32** ビット、**Intel 64** ビット、または **SPARC64** アーキテクチャーで **Oracle Solaris 10** 以上を実行している必要があります。
- **Intel** アーキテクチャー用の **Oracle iPlanet Web Server 7.0.15** 以降、**SPARC** アーキテクチャーでは **7.0.14** 以上が、**NSAPI** コネクターとは別にインストールされ、設定されます。
- ワーカーとして動作する各サーバーに **JBoss EAP 6** がインストールされ、設定されます。「外部 **Web** サーバーからの要求を許可するよう **JBoss EAP 6** を設定」 を参照してください。
- **JBoss** ネイティブコンポーネントの **ZIP** パッケージは、**Customer Service Portal(Customer Service Portal)**からダウンロードされ **https://access.redhat.com** ます。

手順**17.25 NSAPI** コネクターの抽出および設定

1. **JBoss** ネイティブコンポーネントパッケージを抽出します。

   この手順では、ネイティブコンポーネントパッケージが **EAP_HOME** ディレクトリーに展開されることを前提としています。この手順の残りの部分では、**/opt/oracle/webserver7/config/** ディレクトリーは **IPLANET_CONFIG** と呼ばれます。**Oracle iPlanet** 設定ディレクトリーが異なる場合は、適切に手順を変更してください。

2. サーブレットマッピングを無効にします。

   IPLANET_CONFIG/default.web.xml ファイルを開き、**Built In Server Mappings** という見出しのセクションを見つけます。次の **3** つのサーブレットを **XML** コメント文字 **(<!--** および **-->)** で囲み、これらのサーブレットへのマッピングを無効にします。

   - **default**
   - **invoker**
   - **jsp**

以下の設定例は、無効にされたマッピングを示しています。

```
<!-- ============== Built In Servlet Mappings =============== -->
<!-- The servlet mappings for the built in servlets defined above. -->
<!-- The mapping for the default servlet -->
<!--servlet-mapping>
 <servlet-name>default</servlet-name>
 <url-pattern>/</url-pattern>
</servlet-mapping-->
<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
 <servlet-name>invoker</servlet-name>
 <url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->
<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
 <servlet-name>jsp</servlet-name>
 <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

ファイルを保存し、終了します。

3. **iPlanet Web Server** が **NSAPI** コネクターモジュールをロードするよう設定します。

IPLANET_CONFIG/magnus.conf ファイルの最後に次の行を追加し、設定に合わせてファイルパスを変更します。これらの行は、**nsapi_redirector.so** モジュールと **workers.properties** ファイルの場所を定義し、ワーカーとそのプロパティーを一覧表示します。

```
Init fn="load-modules" funcs="jk_init,jk_service"
shlib="EAP_HOME/modules/system/layers/base/native/lib/nsapi_redirector.so"
shlib_flags="(global|now)"

Init fn="jk_init" worker_file="IPLANET_CONFIG/connectors/workers.properties"
log_level="info" log_file="IPLANET_CONFIG/connectors/nsapi.log"
shm_file="IPLANET_CONFIG/connectors/tmp/jk_shm"
```

上記の設定は **32** ビットアーキテクチャー向けです。**64** ビット **Solaris** を使用している場合は、文字列 **lib/nsapi_redirector.so** を **lib64/nsapi_redirector.so** に変更します。

ファイルを保存し、終了します。

4. **NSAPI** コネクターを設定します。

負荷分散のない基本設定または負荷分散設定向けに **NSAPI** コネクターを設定できます。以下のいずれかのオプションを選択します。 その後、設定が完了します。

- 「NSAPI コネクターがクライアントリクエストを JBoss EAP 6 に送信するよう設定」

- 「NSAPI コネクターがクライアントリクエストを複数の JBoss EAP 6 サーバーで分散するよう設定」

バグの報告

17.10.3. NSAPI コネクターがクライアントリクエストを JBoss EAP 6 に送信するよう設定

概要

このタスクでは、NSAPI コネクターが負荷分散やフェイルオーバーなしでクライアント要求を JBoss EAP 6 サーバーにリダイレクトするよう設定します。リダイレクトは、デプロイメントごとに（つまり URL ごとに）行われます。負荷分散の設定については、代わりに 「NSAPI コネクターがクライアントリクエストを複数の JBoss EAP 6 サーバーで分散するよう設定」 を参照してください。

前提条件

- 現在のタスクに進む前に、「Oracle Solaris での NSAPI コネクターの設定」 を完了する必要があります。

手順17.26 基本的な HTTP コネクターの設定

1. JBoss EAP 6 サーバーにリダイレクトする URL パスを定義します。

注記

IPLANET_CONFIG/obj.conf では、前の行から継続する行以外は、行の最初にスペースを挿入しないでください。

IPLANET_CONFIG/obj.conf ファイルを編集します。<Object name="default"> で始まるセクションを見つけ、一致する各 URL パターンを次のサンプルファイルで示された形式で追加

*します。文字列 **jknsapi** は、次の手順で定義される **HTTP** コネクターを示します。例は、ワイルドカードを使用したパターン一致を示しています。*

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
</Object>
```

2. *各パスを提供するワーカーを定義します。*

   ***IPLANET_CONFIG/obj.conf** ファイルの編集を続行します。編集したセクションの終了タグのすぐ後に、**</ Object>** を追加します。*

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="worker01" path="/status"
Service fn="jk_service" worker="worker02" path="/nc(/*)"
Service fn="jk_service" worker="worker01"
</Object>
```

   *上記の例では、**URL** パス **/status** への要求を **worker01** という名前のワーカーにリダイレクトし、**/nc/** の下にあるすべての **URL** パスを **worker02** という名前のワーカーにリダイレクトします。**3** 行目は、前の行で一致しない **jknsapi** オブジェクトに割り当てられたすべての **URL** が **worker01** に提供されることを示しています。*

   ファイルを保存し、終了します。

3. *ワーカーとその属性を定義します。*

   ***IPLANET_CONFIG/connectors/** ディレクトリーに **workers.properties** というファイルを作成します。以下の内容をそのファイルに貼り付けし、お使いの環境に合わせて変更します。*

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

workers.properties ファイルは **Apache mod_jk** と同じ構文を使用します。利用可能なオプションの詳細は、「**Apache mod_jk** ワーカーの設定リファレンス」 を参照してください。

ファイルを保存し、終了します。

4. **iPlanet Web Server** を再起動します。

以下のコマンドを実行し、**iPlanet Web Server** を再起動します。

```
IPLANET_CONFIG/../bin/stopserv
IPLANET_CONFIG/../bin/startserv
```

結果

iPlanet Web Server が、設定した **URL** へのクライアント要求を **JBoss EAP 6** のデプロイメントに送信します。

バグの報告

17.10.4. NSAPI コネクターがクライアントリクエストを複数の JBoss EAP 6 サーバーで分散するよう設定

概要

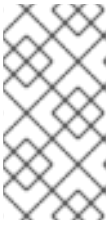
このタスクは、負荷分散設定でクライアントリクエストを **JBoss EAP 6** サーバーに送信するように **NSAPI** コネクターを設定します。**NSAPI** コネクターを負荷分散のない単純な **HTTP** コネクターとして使用する場合は、「**NSAPI** コネクターがクライアントリクエストを **JBoss EAP 6** に送信するよう設定」 を参照してください。

前提条件

- 「**Oracle Solaris** での **NSAPI** コネクターの設定」

手順**17.27** 負荷分散のためコネクターを設定する

1. **JBoss EAP 6** サーバーにリダイレクトする **URL** パスを定義します。



注記

***IPLANET_CONFIG/obj.conf** では、前の行から継続する行以外は、行の最初にスペースを挿入しないでください。*

***IPLANET_CONFIG/obj.conf** ファイルを編集します。**<Object name="default">** で始まるセクションを見つけ、一致する各 **URL** パターンを次のサンプルファイルで示された形式で追加します。文字列 **jknsapi** は、次の手順で定義される **HTTP** コネクターを示します。例は、ワイルドカードを使用したパターン一致を示しています。*

```
<Object name="default">
[...]
NameTrans fn="assign-name" from="/status" name="jknsapi"
NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jmx-console(|/*)" name="jknsapi"
NameTrans fn="assign-name" from="/jkmanager/*" name="jknsapi"
</Object>
```

2. *各パスを提供するワーカーを定義します。*

***IPLANET_CONFIG/obj.conf** ファイルの編集を続行します。前の手順で変更したセクションの終了タグ**(</Object>)**に直接、以下の新しいセクションを追加し、必要に応じて変更します。*

```
<Object name="jknsapi">
ObjectType fn=force-type type=text/plain
Service fn="jk_service" worker="status" path="/jkmanager(/*)"
Service fn="jk_service" worker="router"
</Object>
```

*この **jksnapi** オブジェクトは、**default** オブジェクトの **name="jksnapi"** マッピングにマップされた各パスを提供するために使用されるワーカーノードを定義します。**/jkmanager/*** に一致する **URL** 以外のすべてが、**router** という名前のワーカーにリダイレクトされます。*

3. *ワーカーとその属性を定義します。*

***workers.properties** という名前のファイルを **IPLANET_CONFIG/connector/** で作成します。以下の内容をそのファイルに貼り付けし、お使いの環境に合わせて変更します。*

```
# The advanced router LB worker
# A list of each worker
worker.list=router,status

# First JBoss EAP server
```

```
# (worker node) definition.
# Port 8009 is the standard port for AJP
#

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second JBoss EAP server
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the load-balancer called "router"
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status
```

workers.properties ファイルは **Apache mod_jk** と同じ構文を使用します。利用可能なオプションの詳細は、「**Apache mod_jk** ワーカーの設定リファレンス」 を参照してください。

ファイルを保存し、終了します。

4. **iPlanet Web Server 7.0** を再起動します。

```
IPLANET_CONFIG/../bin/stopserv
IPLANET_CONFIG/../bin/startserv
```

結果

iPlanet Web Server は、設定した **URL** パターンを負荷分散設定の **JBoss EAP 6** サーバーにリダイレクトします。

バグの報告

第18章 MESSAGING

18.1. はじめに

18.1.1. HornetQ

HornetQ は、**Red Hat** により開発されたマルチプロトコル非同期メッセージングシステムです。**HornetQ** は、サーバーに障害が発生した場合のメッセージの信頼性を保証するために、自動クライアントフェイルオーバーで高可用性**(HA)**を提供します。**HornetQ** は、負荷分散されたメッセージで柔軟なクラスタリングソリューションもサポートします。

HornetQ は **JBoss EAP 6** の **Java Message Service(JMS)**プロバイダーで、メッセージングサブシステムとして設定されます。

バグの報告

18.1.2. 低速な HornetQ コンシューマーの処理

サーバー側のキュー（**JMS** トピックサブスクライバーなど）を持つ低速なコンシューマーは、ブローカーのパフォーマンスに大きな問題を引き起こす可能性があります。メッセージがコンシューマーのサーバー側のキューで構築されると、メモリーは一杯になり、ブローカーはページングモードになり、パフォーマンスに悪影響を及ぼす可能性があります。ただし、メッセージをすぐに確認しないコンシューマーをブローカーから切断できるように基準を設定できます。この場合、非永続的な **JMS** サブスクライバーの場合、ブローカーがサブスクリプションを削除し、そのメッセージすべてで有用なサーバーリソースを解放できます。

バグの報告

18.1.3. フェイルオーバー中のブロック呼び出しの処理

クライアントコードがサーバーへのブロック呼び出しにあり、実行を継続する応答を待つと、フェイルオーバーが発生すると、新しいセッションには、進行中の呼び出しに関する知識はありません。そうでないと、この呼び出しはハングし、到達しない応答を待つ可能性があります。

これを防ぐため、**HornetQ** は、**JMS** を使用する場合は **javax.jms.JMSException** をスローし、エラーコード **HornetQException.UNBLOCKED** のある **HornetQException** をスローすることにより、フェイルオーバー時に進行中のブロッキング呼び出しのブロックを解除します。この例外をキャッチし、必要に応じて操作を再試行するかどうかはクライアントコード次第です。

ブロック解除されるメソッドが **commit**（）または **prepare**（）の呼び出しである場合、トランザクションは自動的にロールバックされ、**HornetQ** は（**JMS** を使用する場合）**javax.jms.TransactionRolledBackException** をスローし、コア **API** を使用する場合はエラーコード **HornetQException.TRANSACTION_ROLLED_BACK** を設定して **HornetQException** をスローします。

バグの報告

18.1.4. トランザクションによるフェイルオーバーの処理

セッションがトランザクションであり、現在のトランザクションでメッセージがすでに送受信された場合、サーバーはフェイルオーバー中に送受信されたメッセージが失われていないことを確認できません。

そのため、トランザクションはロールバック専用としてマークされ、その後コミットしようとすると **javax.jms.TransactionRolledBackException**（**JMS** を使用している場合は）をスローし、コア **API** を使用する場合はエラーコード **HornetQException.TRANSACTION_ROLLED_BACK** を設定して **HornetQException** をスローします。

例外をキャッチし、必要に応じてクライアント側のローカルロールバックコードを実行するのはユーザー次第です。セッションを手動でロールバックする必要はありません。すでにロールバックされています。その後、ユーザーは同じセッションでトランザクション操作を再試行できます。

コミット呼び出しの実行中にフェイルオーバーが発生すると、前述したようにサーバーが応答を返さないため、ハングを防ぐために呼び出しのブロックを解除します。この場合、障害が発生する前にトランザクションのコミットがライブサーバーで実際に処理されたかどうかをクライアントが判断するのは容易ではありません。

これを修正するには、クライアントはトランザクションで重複検出を有効にし、呼び出しがブロック解除された後にトランザクション操作を再試行します。フェイルオーバー前にトランザクションが実際にライブサーバーで正常にコミットされた場合、トランザクションが再試行されると、重複検出により、トランザクションで再送信された永続メッセージがサーバーで無視され、それらが複数回送信されないようにします。

注記

ロールバック例外をキャッチして再試行すると、ブロックされていない呼び出しをキャッチし、重複検出を有効にすることで、**1** 回限りの配信保証は、メッセージに障害が発生した場合に提供でき、**100%** の損失や重複の保証が保証されます。

バグの報告

18.1.5. 非トランザクションセッションを使用したフェイルオーバーの処理

セッションがトランザクションではない場合、フェイルオーバー時にメッセージまたは確認応答が失われる可能性があります。

トランザクション以外のセッションに、**1** 回限りの配信保証を提供し、重複検出を有効にし、ブロック解除例外をキャッチします。

バグの報告

18.1.6. 接続失敗の通知

JMS は、接続障害の非同期に通知する標準メカニズムを提供します**(java.jms.ExceptionListener)**。**ExceptionListener** の詳細は、**Oracle javax.jms Javadoc** を参照してください。

HornetQ コア **API** は、**org.hornet.core.client.SessionFailureListener** クラスの形式で同様の機能も提供します。

JMS ExceptionListener または **Core SessionFailureListener** インスタンスは、接続が正常に失敗したか、再接続または再アタッチされたかどうかに関係なく、接続障害が発生した場合に常に **HornetQ** によって呼び出されます。

バグの報告

18.1.7. Java Messaging Service (JMS)

メッセージングシステムにより、異種システムを疎結合して、信頼性も高めることができます。**Java Messaging Service(JMS)** プロバイダーはトランザクションのシステムを使用して、変更をアトミックにコミットまたはロールバックします。**Remote Procedure Call (RPC)** パターンに基づいたシステムとは異なり、メッセージングシステムは、リクエストと応答の間に密接な関係がないパターンを渡す非同期メッセージを主に使用します。ほとんどのメッセージングシステムは **request-response** モードもサポートしますが、メッセージングシステムの主な機能ではありません。

メッセージングシステムはメッセージのコンシューマーからメッセージの送信者を切り離します。

メッセージの送信者とコンシューマーは完全に独立しており、相互に何も知りません。これにより、柔軟に結合されたシステムを作成できます。多くの場合、大規模な企業ではメッセージングシステムを使用して、異種システムを結合するメッセージバスを実装します。メッセージバスは、**Enterprise Service Bus(ESB)**の中核を形成することがよくあります。メッセージバスを使用して異種システムを切り離すことで、システムを拡大し、より容易に適応させることができます。また、不安定な依存関係がないため、より柔軟に新しいシステムを追加したり、古いシステムを廃止したりできます。

バグの報告

18.1.8. サポートされているメッセージ形式

HornetQ は、以下のメッセージ形式に対応しています。

Message Queue パターン

Message Queue パターンでは、メッセージをキューに送信する必要があります。キューに入ると、通常、メッセージは永続化され、配信が保証されます。メッセージがキューを通過すると、メッセージングシステムはメッセージコンシューマーに配信します。メッセージコンシューマーは、処理後にメッセージの配信を確認します。

Message Queue パターンでは、ポイントツーポイントメッセージングと併用すると、複数のコンシューマーをキューに入れることが可能ですが、各メッセージは単一のコンシューマーのみが受信可能となります。

Publish-Subscribe パターン

Publish-Subscribe パターンにより、複数の送信者がサーバー上の単一のエンティティーにメッセージを送信できます。このエンティティーは、「トピック」と呼ばれることがよくあります。各トピックは、「サブスクリプション」と呼ばれる複数のコンシューマーによって参加できます。

各サブスクリプションは、トピックに送信されたすべてのメッセージのコピーを受け取ります。これは、各メッセージが単一のコンシューマーによってのみ消費される **Message Queue** パターンとは異なります。

永続的なサブスクリプションは、サブスクライバーが消費するまで、トピックに送信された各メッセージのコピーを保持します。これらのコピーは、サーバーの再起動時にも保持されます。非永続的なサブスクリプションは、それらを作成した接続のみを保持します。

バグの報告

18.2. トランスポートの設定

18.2.1. アクセプターおよびコネクター

HornetQ は、メッセージングシステムの主要な部分としてコネクターおよびアクセプターを使用します。

アクセプターおよびコネクター

acceptor

アクセプターは、**HornetQ** サーバーが受け入れる接続タイプを定義します。

connector

コネクターは、**HornetQ** サーバーに接続する方法を定義し、**HornetQ** クライアントによって使用されます。

一致するコネクターとアクセプターのペアが同じ **JVM** 内に存在するかどうかに応じて、**2** タイプのコネクターとアクセプターが用意されています。

Invm および **Netty**

invm

invm は **Intra Virtual Machine** の省略形です。クライアントとサーバーの両方が同じ **JVM** で実行されている場合に使用することができます。

Netty

JBoss プロジェクトの名前。クライアントとサーバーが異なる **JVM** で実行されている場合に使用する必要があります。

HornetQ クライアントは、いずれかのサーバーのアクセプターと互換性のあるコネクターを使用する必要があります。**Invm** コネクターのみが **Invm** アクセプターに接続でき、**netty** コネクターのみが **netty** アクセプターに接続できます。コネクターとアクセプターはいずれも **standalone.xml** および **domain.xml** のサーバー上で設定されます。管理コンソールまたは管理 **CLI** を使用して定義できます。

バグの報告

18.2.2. Netty TCP の設定

Netty TCP は暗号化されていない単純な **TCP** ソケットベースのトランスポートです。**Netty TCP** は、古いブロッキング **Java IO** または非ブロッキング **Java NIO** を使用するよう設定できます。多くの同時接続でスケーラビリティーを向上するには、サーバー側で **Java NIO** を使用することが推奨されます。同時接続の数が **Java** 古い **IO** が少ないと、**NIO** よりもレイテンシーが良くなります。

Netty TCP は暗号化されていないため、信頼できないネットワーク上で接続を実行する場合には推奨されません。**Netty TCP** トランスポートでは、すべての接続がクライアント側から開始されます。

例**18.1** デフォルトの **EAP** 設定からの **Netty TCP** の設定例

```
<connectors>
  <netty-connector name="netty" socket-binding="messaging"/>
  <netty-connector name="netty-throughput" socket-binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
  </netty-connector>
  <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging"/>
  <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
    <param key="batch-delay" value="50"/>
    <param key="direct-deliver" value="false"/>
  </netty-acceptor>
  <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
```

設定例は、**HornetQ** の **JBoss EAP 6** 実装がアクセプターおよびコネクター設定でソケットバインディングを使用する方法を示しています。これは、**HornetQ** のスタンドアロンバージョンとは異なり、特定のホストおよびポートを宣言する必要があります。

下表は **Netty TCP** 設定プロパティーの説明になります。

表**18.1 Netty TCP** 設定プロパティー

プロパティー	デフォルト	説明

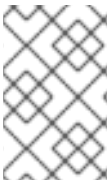
プロパティー	デフォルト	説明
batch-delay	**0** ミリ秒	パケットをトランスポートに書き込む前に、**HornetQ** は **batch-delay** ミリ秒の最大書き込みを一括処理するよう設定できます。これにより、メッセージ転送の平均待ち時間が長くなり、非常に小さいメッセージのスループットの合計が増加します。
direct-deliver	**true**	メッセージがサーバーに到達し、待機しているコンシューマーに配信されると、デフォルトでは、メッセージが到達した同じスレッドで配信が実行されます。これにより、メッセージが比較的小さく、コンシューマーの数が少ない環境では適切な待ち時間になりますが、スループットと待ち時間は低減されます。スループットを最大限にする場合は、このプロパティーを「**false**」に設定します。
local-address	**[使用可能なローカルアドレス]**	**Netty** コネクターでは、リモートアドレスへの接続時にクライアントが使用するローカルアドレスを指定するために使用されます。ローカルアドレスが指定されていない場合、コネクターは利用可能なローカルアドレスを使用します。
local-port	**0**	**Netty** コネクターでは、リモートアドレスへの接続時にクライアントが使用するローカルポートを指定するために使用されます。**local-port** のデフォルト**(0)**が使用される場合、コネクターはシステムが一時ポートを取得できるようにします。有効なポートは **0** から **65535** です。

プロパティー	デフォルト	説明
nio-remoting-threads	**-1**	**NIO** を使用するように設定されている場合、デフォルトでは受信パケットを処理するために **Runtime.getRuntime（）.availableProcessors**（）によって報告されるコア（またはハイパースレッド）の数の **3** 倍のスレッドを使用します。この値をオーバーライドするには、スレッド数にカスタム値を設定します。
tcp-no-delay	**true**	**true** の場合、**Nagle** アルゴリズムが有効になります。このアルゴリズムによりネットワーク上で送信されるパケット数を減らすことで、**TCP/IP** ネットワークの効率を向上します。
tcp-send-buffer-size	**32768** バイト	このパラメーターは **TCP** が送信するバッファーのサイズ **(**バイト単位**)** を決定します。
tcp-receive-buffer-size	**32768** バイト	このパラメーターは **TCP** が受信するバッファーのサイズ **(**バイト単位**)** を決定します。
use-nio	**false**	**true** の場合、ノンブロッキング **NIO** が使用されます。**false** に設定された場合、古いブロッキング **Java IO** が使用されます。多くの同時接続を処理するには、非ブロッキング **Java NIO** を使用する必要があります。それ以外の場合は、古い（ブロッキング）**IO** になります。
use-nio-global-worker-pool	**false**	このパラメーターにより、すべての **JMS** 接続で **Java** スレッドの **1** つのプールが共有されます（各コネクションに独自のプールがあるわけではありません）。これは、オペレーティングシステムのプロセスの最大数を使い切らないようにするためのものです。

重要

use-nio を **true** に設定した場合には、**use-nio-global-worker-pool** パラメーターを使用して、マシンが多数の接続を作成するリスクを最小限に抑え、**OutOfMemory** エラーが発生する可能性があります。

```
<netty-connector name="netty" socket-binding="messaging">
  <param key="use-nio" value="true"/>
  <param key="use-nio-global-worker-pool" value="true"/>
</netty-connector>
```



注記

Netty TCP プロパティーはすべてのタイプのトランスポートに対して有効です。

バグの報告

18.2.3. Netty セキュアソケットレイヤー (SSL) の設定

Netty TCP は暗号化されていない単純な **TCP** ソケットベースのトランスポートです。**Netty SSL** は **Netty TCP** と似ていますが、**SSL(Secure Sockets Layer)**を使用して **TCP** 接続を暗号化することでセキュリティーを強化します。

警告

SSL コネクター/アクセプターでの **SSLv3** プロトコルの使用は、「**Poodle**」の脆弱性により許可されませんでした。このプロトコルに接続する **JMS** クライアントは拒否されます。

次の例は、一方向 **SSL** の **Netty** 設定を示しています。

注記

以下のパラメーターのほとんどは、アクセプターおよびコネクターで使用できます。ただし、一部のパラメーターはアクセプターでのみ機能します。パラメーターの説明では、これらのパラメーターをコネクターとアクセプターで使用する際の違いを説明します。

```
<acceptors>
 <netty-acceptor name="netty" socket-binding="messaging"/>
   <param key="ssl-enabled" value="true"/>
   <param key="key-store-password" value="[keystore password]"/>
   <param key="key-store-path" value="[path to keystore file]"/>
 </netty-acceptor>
</acceptors>
```

表**18.2 Netty SSL** 設定プロパティー

プロパティー名	デフォルト	説明
ssl-enabled	**true**	**SSL** を有効にします。

プロパティー名	デフォルト	説明
key-store-password	*[キーストアのパスワード]*	*アクセプターで使用されると、サーバー側のキーストアのパスワードになります。* *コネクターで使用されると、クライアント側のキーストアのパスワードになります。これは、**2** 通りの **SSL**（相互認証）を使用している場合にコネクターにのみ関係します。この値はサーバー上で設定できますが、ダウンロードしてクライアントで使用します。*

プロパティー名	デフォルト	説明
key-store-path	*[キーストアファイルへのパス]*	*アクセプターで使用される場合、これはサーバーの証明書を保持するサーバー（自己署名または認証局によって署名されるかどうか）の **SSL** キーストアへのパスになります。* *コネクターで使用される場合、これはクライアント証明書を保持するクライアント側の **SSL** キーストアへのパスとなります。これは、双方向 **SSL**（相互認証など）を使用している場合にコネクターにのみ関係します。この値はサーバー上で設定されますが、ダウンロードしてクライアントで使用します。*

*双方向 **SSL** (サーバーとクライアント間の相互認証) に **Netty** を設定する場合は、前述の一方向 **SSL** の例で説明したパラメーターの他に **3** つのパラメーターが使用されます。*

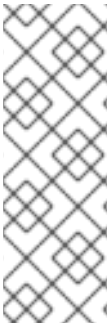
- ***need-client-auth:** クライアント接続の双方向（相互認証）の必要性を指定します。*
- ***trust-store-password:** アクセプターで使用されると、サーバー側のトラストストアのパスワードになります。コネクターで使用されると、クライアント側のトラストストアのパスワードになります。これは、一方向および双方向 **SSL** のコネクターに関連します。この値はサーバー上で設定できますが、ダウンロードしてクライアントで使用します。*
- ***trust-store-path:** アクセプターで使用されると、サーバーが信頼するすべてのクライアントのキーを保持するサーバー側の **SSL** トラストストアへのパスになります。コネクターで使用される場合、これはクライアント側の **SSL** キーストアへのパスになります。これは、クライアントが信頼するすべてのサーバーの公開鍵を保持します。これは、一方向および双方向 **SSL** の*

コネクターに関連します。このパスはサーバー上で設定できますが、ダウンロードしてクライアントで使用します。

バグの報告

18.2.4. Netty HTTP の設定

Netty HTTP は **HTTP** プロトコル上でパケットをトンネリングします。これは、ファイアウォールが **HTTP** トラフィックのみを通過できるシナリオで役に立ちます。**Netty HTTP** は **Netty TCP** と同じプロパティーを使用し、以下の追加プロパティーを使用します。



注記

以下のパラメーターはアクセプターおよびコネクターで使用できます。**Netty HTTP** トランスポートは標準の **HTTP** ポートを再利用することを許可しません（デフォルトでは **8080**）。標準の **HTTP** ポートを使用すると例外が発生します。標準の **HTTP** ポートを介して **HornetQ** 接続をトンネリングするのに 「**Netty** サーブレットの設定」 **(Netty Servlet Transport)**を使用できます。

```
<socket-binding name="messaging-http" port="7080" />
```

```
<acceptors>
  <netty-acceptor name="netty" socket-binding="messaging-http">
    <param key="http-enabled" value="false"/>
    <param key="http-client-idle-time" value="500"/>
    <param key="http-client-idle-scan-period" value="500"/>
    <param key="http-response-time" value="10000"/>
    <param key="http-server-scan-period" value="5000"/>
    <param key="http-requires-session-id" value="false"/>
  </netty-acceptor>
</acceptors>
```

下表は、**Netty HTTP** の設定に使用する追加プロパティーの説明になります。

表**18.3 Netty HTTP** 設定プロパティー

プロパティー名	デフォルト	説明
http-enabled	**false**	**true** の場合、**HTTP** が有効になります。
http-client-idle-time	**500** ミリ秒	接続を維持するために空の **HTTP** 要求を送信する前にクライアントがアイドル状態でいられる期間。
http-client-idle-scan-period	**500** ミリ秒	アイドル状態のクライアントに対してスキャンを行う頻度 **(**ミリ秒単位**)**。
http-response-time	**10000** ミリ秒	接続を維持するために空の **HTTP** 応答を送信する前に、サーバーが待機できる期間。
http-server-scan-period	**5000** ミリ秒	応答が必要なクライアントに対してスキャンを行う頻度 **(**ミリ秒単位**)**。
http-requires-session-id	**false**	**true** の場合、クライアントは最初の呼び出しの後にセッション **ID** を取得するため待機します。

警告

自動クライアントフェールオーバーは、**Netty HTTP** トランスポートを通じて接続しているクライアントに対してはサポートされません。

バグの報告

18.2.5. Netty サーブレットの設定

サーブレットトランスポートを使用すると、HTTP 経由で HornetQ トラフィックをサーブレットエンジンで稼働しているサーブレットにトンネリングし、インVM HornetQ サーバーへリダイレクトできます。Netty HTTP トランスポートは、特定ポートで HTTP トラフィックをリッスンする Web サーバーとして機能します。サーブレットトランスポートでは HornetQ トラフィックは、すでに Web サイトや他のアプリケーションを提供している可能性のあるサーブレットエンジンを介してプロキシーされます。

Netty サーブレットトランスポートが動作するようにサーブレットエンジンを設定するには、以下の手順に従います。

- サーブレットをデプロイします。以下の例はサーブレットを使用する Web アプリケーションを表しています。

```
<web-app>
  <servlet>
    <servlet-name>HornetQServlet</servlet-name>
    <servlet-class>org.jboss.netty.channel.socket.http.HttpTunnelingServlet</servlet-
class>
    <init-param>
      <param-name>endpoint</param-name>
      <param-value>local:org.hornetq</param-value>
    </init-param>
      <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>HornetQServlet</servlet-name>
    <url-pattern>/HornetQServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

init パラメーター エンドポイント は、サーブレットがパケットを転送する Netty アクセプターのホスト属性を指定します。

- Netty サーブレットアクセプターをサーバー側の設定に挿入します。以下の例は、サーバー設定ファイル（standalone.xml および domain.xml）でのアクセプターの定義を示しています。

```
<acceptors>
  <acceptor name="netty-servlet">
    <factory-class>
      org.hornetq.core.remoting.impl.netty.NettyAcceptorFactory
    </factory-class>
    <param key="use-servlet" value="true"/>
```

```
    <param key="host" value="org.hornetq"/>
  </acceptor>
</acceptors>
```

- 最後に、サーバー設定ファイル（**standalone.xml** および **domain.xml**）でクライアントのコネクターを定義します。

```
<netty-connector name="netty-servlet" socket-binding="http">
  <param key="use-servlet" value="true"/>
  <param key="servlet-path" value="/messaging/HornetQServlet"/>
</netty-connector>
```

- また、以下の設定をコネクターに追加すると、サーブレットトランスポートを **SSL** 上でも使用できます。

```
<netty-connector name="netty-servlet" socket-binding="https">
  <param key="use-servlet" value="true"/>
  <param key="servlet-path" value="/messaging/HornetQServlet"/>
  <param key="ssl-enabled" value="true"/>
  <param key="key-store-path" value="path to a key-store"/>
  <param key="key-store-password" value="key-store password"/>
</netty-connector>
```

警告

自動クライアントフェールオーバーは、**HTTP** トンネリングサーブレットを介して接続しているクライアントに対してはサポートされません。

注記

HornetQ クラスターの設定するために Netty サーブレットを使用して EAP 6 サーバーを設定することはできません。

バグの報告

18.3. デッド接続の検出

18.3.1. サーバーでデッド接続リソースを閉じる

HornetQ コアまたは JMS クライアントアプリケーションは、終了する前にそのリソースを閉じる必要があります。アプリケーションのコードで finally ブロックを使用すると、アプリケーションがリソースを自動的に閉じるように設定できます。

以下の例は、finally ブロックでセッションとセッションファクトリーを閉じるコアクライアントアプリケーションを示しています。

```
ServerLocator locator = null;
ClientSessionFactory sf = null;
ClientSession session = null;

try
{
   locator = HornetQClient.createServerLocatorWithoutHA(..);

   sf = locator.createClientSessionFactory();;

   session = sf.createSession(...);

   ... do some operations with the session...
}

finally
{
   if (session != null)
   {
      session.close();
   }

   if (sf != null)
   {
      sf.close();
```

```
    }

    if(locator != null)
    {
      locator.close();
    }
  }
```

以下の例は、**finally** ブロックでセッションとセッションファクトリーを閉じる **JMS** クライアントアプリケーションを示しています。

```
  Connection jmsConnection = null;

  try
  {
    ConnectionFactory jmsConnectionFactory =
  HornetQJMSClient.createConnectionFactoryWithoutHA(...);

    jmsConnection = jmsConnectionFactory.createConnection();

    ... do some operations with the connection...
  }
  finally
  {
    if (connection != null)
    {
      connection.close();
    }
  }
```

接続 Time to Live (TTL) パラメーターの使用

connection-ttl パラメーターは、クライアントからデータまたは **ping** パケットを受信しない場合に、サーバーが接続を存続する期間を決定します。このパラメーターにより、古いセッションなどのデッドサーバーリソースが長く維持され、障害が発生したネットワーク接続が復旧したときにクライアントが再接続できるようになります。

HornetQConnectionFactory インスタンスに **connection-ttl** パラメーターを指定すると、**JMS** クライアントの接続 **TTL** を定義できます。**JMS** 接続ファクトリーインスタンスを **JNDI** に直接デプロイする場合は、**standalone.xml** および **domain.xml** サーバー設定ファイルで **connection-ttl** パラメーターを定義できます。

connection-ttl パラメーターのデフォルト値は **60000** ミリ秒です。クライアントが独自の接続 **TTL** を指定する必要がない場合は、サーバー設定ファイルに **connection-ttl-override** パラメーターを定義

して、すべての値を上書きできます。**connection-ttl-override** パラメーターはデフォルトで無効にされ、値が **-1** になります。

ガベージコレクション

HornetQ はガベージコレクションを使用して、**finally** ブロックで明示的に閉じられていないセッションを検出し、閉じます。**HornetQ** サーバーは、セッションを閉じる前に以下のような警告をログに記録します。

```
[Finalizer] 20:14:43,244 WARNING [org.hornetq.core.client.impl.DelegatingSession]  I'm
closing a ClientSession you left open. Please make sure you close all ClientSessions
explicitly before let
ting them go out of scope!
[Finalizer] 20:14:43,244 WARNING [org.hornetq.core.client.impl.DelegatingSession]  The
session you didn't close was created here:
java.lang.Exception
   at org.hornetq.core.client.impl.DelegatingSession.<init>(DelegatingSession.java:83)
   at org.acme.yourproject.YourClass (YourClass.java:666)
```

このログメッセージには、**JMS** 接続またはユーザーセッションが作成され、閉じられなかったコード部分に関する情報が含まれます。

バグの報告

18.3.2. クライアントサイド障害の検出

クライアントアプリケーションは、クライアントがシャットダウンしないように、**ping** パケットをサーバーに自動的に送信します。同様に、クライアントアプリケーションは、サーバーからデータを受信する限り、接続がアライブ状態であると見なします。

クライアントが **client-failure-check-period** パラメーターで指定された期間サーバーからデータパケットを受信しないと、クライアントは接続が失敗したと見なします。その後、クライアントはフェイルオーバーを開始するか、**FailureListener** インスタンスを起動します。

JMS クライアントでは、**HornetQConnectionFactory** インスタンスの **ClientFailureCheckPeriod** 属性を使用してクライアント障害チェック期間が設定されます。**JMS** 接続ファクトリーインスタンスをサーバー側の **JNDI** に直接デプロイする場合は、**standalone.xml** および **domain.xml** サーバー設定ファイルで **client-failure-check-period** パラメーターを指定できます。

クライアント障害チェック期間のデフォルト値は **30000** ミリ秒です。**-1** の値は、サーバーからデータが受信されない場合、クライアントが接続を閉じないことを意味します。

非同期接続実行の設定

デフォルトでは、サーバー側で受信されるパケットはリモーティングスレッドで実行されます。スレッドプールからの任意のスレッドで操作を非同期に処理して、リモーティングスレッドを解放できます。**standalone.xml** および **domain.xml** サーバー設定ファイルの **async-connection-execution-enabled** パラメーターを使用して、非同期接続実行を設定できます。このパラメーターのデフォルト値は「**true**」です。

注記

スレッドプールから任意のスレッドで操作を非同期的に処理すると、待ち時間が少し追加されます。パフォーマンス上の理由から、短い実行操作は常にリモーティングスレッドで処理されます。

バグの報告

18.4. サイズの大きなメッセージの処理

18.4.1. サイズの大きなメッセージの処理

HornetQ は、クライアントまたはサーバーでメモリーのサイズが制限されていても、大きなメッセージの使用をサポートします。大きなメッセージは、そのままストリーミングしたり、より効率的な転送のためにさらに圧縮したりできます。ユーザーはメッセージのボディーに **InputStream** を設定すると、大きなメッセージを送信できます。メッセージが送信されると、**HornetQ** はこの **InputStream** を読み取り、データを断片的にサーバーに送信します。

クライアントまたはサーバーは、大きなメッセージのボディーをメモリーに保存しません。コンシューマーは、最初にボディーが空の大きなメッセージを受け取った後、メッセージに **OutputStream** を設定して、断片的にディスクファイルへストリーミングします。

バグの報告

18.4.2. HornetQ の大きなメッセージの設定

サーバーの設定

スタンドアロンモードでは、大きなメッセージは **EAP_HOME/standalone/data/largemessages** ディレクトリーに保存されます。ドメインモードでは、大きなメッセージは **EAP_HOME/domain/servers/SERVERNAME/data/largemessages** ディレクトリーに保存されます。設定プロパティー **large-messages-directory** は、大きなメッセージが保存される場所を示します。

重要

最良のパフォーマンスを得るため、大きなメッセージは、メッセージジャーナルやページングディレクトリーとは別の物理ボリュームに保存することが推奨されます。

バグの報告

18.4.3. パラメーターの設定

さまざまなパラメーターを設定して **HornetQ** の大きなメッセージを設定できます。

クライアント側での **HornetQ Core API** の使用

クライアント側で **HornetQ Core API** を使用している場合は、**ServerLocator.setMinLargeMessageSize** パラメーターを設定して大きなメッセージの最小サイズを指定する必要があります。大きなメッセージの最小サイズ**(min-large-message-size)**はデフォルトで **100KiB** に設定されています。

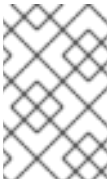
```
ServerLocator locator = HornetQClient.createServerLocatorWithoutHA(new
TransportConfiguration(NettyConnectorFactory.class.getName()))

locator.setMinLargeMessageSize(25 * 1024);

ClientSessionFactory factory = HornetQClient.createClientSessionFactory();
```

Java Messaging Service (JMS) クライアントに対するサーバーの設定

Java Messaging Service(JMS)を使用している場合は、サーバー設定ファイル（**standalone.xml** および **domain.xml**）の **min-large-message-size** 属性に大きなメッセージの最小サイズを指定する必要があります。大きなメッセージの最小サイズ**(min-large-message-size)**はデフォルトで **100KiB** に設定されています。



注記

min-large-message-size 属性の値はバイト単位である必要があります。

高速かつ効率的な転送のために大きなメッセージを圧縮することを選択できます。圧縮/圧縮解除操作はすべてクライアント側で処理されます。圧縮されたメッセージが **min-large-message-size** よりも小さい場合、通常のメッセージとしてサーバーに送信されます。**Java Messaging Service(JMS)**を使用すると、サーバーロケーターまたは **ConnectionFactory** にブール値プロパティー **compress-large-messages "true"** を設定して大きなメッセージを圧縮できます。

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty"/>
  </connectors>
...
  <min-large-message-size>204800</min-large-message-size>
  <compress-large-messages>true</compress-large-messages>
</connection-factory>
```

バグの報告

18.5. ページング

18.5.1. ページングについて

HornetQ は、各キューに数百万個のメッセージが含まれる多くのメッセージキューをサポートします。**HornetQ** サーバーは制限されたメモリーで実行されるため、すべてのメッセージキューを一度にメモリーに保存することは困難です。

ページングは **HornetQ** サーバーによって使用されるメカニズムで、限られたメモリーで大きなメッセージキューに対応するために、必要に応じてメッセージを透過的に出し入れします。

メモリーにある特定アドレスのメッセージのサイズが最大設定メッセージサイズを越えたとき、**HornetQ** はメッセージをディスクにページングします。

注記

HornetQ のページングはデフォルトで有効になっています。

バグの報告

18.5.2. ページファイル

複数のファイルにメッセージを格納するファイルシステムの各アドレスには、個別のフォルダーがあります。メッセージを格納するこれらのファイルは、ページファイルと呼ばれます。各ファイルには、最大設定メッセージサイズ**(page-size-bytes)**までのメッセージが含まれています。

システムは必要に応じてページファイルを移動して、ページのメッセージがすべてクライアントに受信されるとすぐにページファイルを削除します。

注記

コンシューマーがキューからメッセージを読み取るメッセージセレクターを持っている場合、セレクターに一致するメモリーのメッセージのみがコンシューマーに送信されます。コンシューマーがこれらのメッセージの配信を確認すると、新しいメッセージはページングされ、メモリーに読み込まれます。パフォーマンス上の理由から、**HornetQ** は、ページングされたメッセージをコンシューマーのメッセージセレクターと一致するかどうかを確認せず、新しいページ化されたメッセージがコンシューマーのメッセージセレクターと一致することを確認することはありません。ページファイルのディスクのコンシューマーのセレクターに一致するメッセージがありますが、別のコンシューマーがメモリーでメッセージを読み取り、空き領域を提供するまで、**HornetQ** はそれらをメモリーに読み込みません。空き領域が利用できない場合、セレクターを持つコンシューマーは新しいメッセージを受信できない可能性があります。

バグの報告

18.5.3. ページングフォルダーの設定

グローバルページングパラメーターはサーバー設定ファイル（**standalone.xml** および **domain.xml**）に指定されます。**paging-directory** パラメーターを使用して、ページングディレクトリー/フォルダーの場所を設定できます。

```
<hornetq-server>
 ...
 <paging-directory>/location/paging-directory</paging-directory>
```

```
...
</hornetq-server>
```

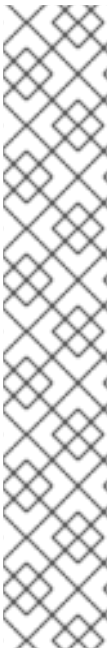
paging-directory パラメーターは、ページファイルを保存する場所/フォルダーを指定するために使用されます。**HornetQ** は、このページングディレクトリーの各ページングアドレスにフォルダーを作成します。ページファイルはこれらのフォルダーに保存されます。

デフォルトのページングディレクトリーは **EAP_HOME/standalone/data/messagingpaging**（スタンドアロンモード）と **EAP_HOME/domain/servers/SERVERNAME/data/messagingpaging**（ドメインモード）です。

バグの報告

18.5.4. ページングモード

アドレスに送信されたメッセージが設定サイズを越える場合は、そのアドレスが「ページ/ページングモード」になります。



備考

ページングはアドレスごとに個別に行われます。アドレスに **max-size-bytes** を設定する場合は、一致する各アドレスに指定した最大サイズがあることを意味します。ただし、一致するすべてのアドレスの合計サイズが **max-size-bytes** に制限されるわけではありません。

page モードであっても、メモリー不足によるエラーが原因でサーバーがクラッシュする可能性があります。**HornetQ** は、ディスク上の各ページファイルへの参照を保持します。ページファイルが非常に多い状況では、**HornetQ** はメモリーの消費に直面する可能性があります。このリスクを最小限に抑えるには、**page-size-bytes** 属性を適切な値に設定することが重要です。**JBoss EAP 6** サーバーのメモリーを（宛先の数）***(max-size-bytes)** より高く設定する必要があります。そうでないと、メモリー不足のエラーが発生する可能性があります。

サーバー設定ファイル（**standalone.xml** および **domain.xml**）のアドレスの最大サイズをバイト単位で設定できます**(max-size-bytes)**。

```
<address-settings>
  <address-setting match="jms.someaddress">
    <max-size-bytes>104857600</max-size-bytes>
    <page-size-bytes>10485760</page-size-bytes>
```

```
    <address-full-policy>PAGE</address-full-policy>
  </address-setting>
</address-settings>
```

以下の表には、アドレス設定のパラメーターをまとめています。

表18.4 ページングアドレス設定

要素	デフォルト値	説明
max-size-bytes	10485760	ページングモードになる前にアドレスが取得できる最大メモリーサイズを指定します。
page-size-bytes	2097152	これは、ページングシステムで使用される各ページファイルのサイズを指定するために使用されます。
address-full-policy	PAGE	この属性の値は、ページングの決定に使用されます。この属性の値のいずれかを設定できます： **PAGE**: ページングを有効にし、設定された制限を超えたメッセージをディスクに指定するには、**DROP**: 設定された制限を超えるメッセージを警告なしで破棄するには、**FAIL**: FAIL: メッセージをドロップし、クライアントメッセージプロデューサーに例外を送信するには、**BLOCK**: クライアントメッセージプロデューサーがセット制限を超えるメッセージを送信するときにブロックします。
page-max-cache-size	5	システムは、ページングナビゲーション中に入出力を最適化するために、メモリーに最大 ***page-max-cache-size*** のページファイルを維持します。

重要

最大サイズに達したときにメッセージをページングしたくない場合は、***address-full-policy*** をそれぞれ ***DROP*** に設定することで、メッセージをドロップしたり、クライアント側で例外のあるメッセージをドロップしたり、プロデューサーが追加のメッセージを送信しないようにブロックしたりすることを選択できます。 デフォルト設定では、アドレスが ***max-size-bytes*** に達すると、すべてのアドレスがメッセージのページングに設定されます。

複数のキューを持つアドレス

複数のキューがバインドされているアドレスにメッセージがルーティングされると、メッセージのコピーはメモリー内に **1** つだけあります。各キューは、このメッセージの元のコピーへの参照のみを処理します。したがって、メモリーは、元のメッセージを参照するすべてのキューがメッセージを配信した場合にのみ解放されます。

備考

単一のレイジーキュー/サブスクリプションによって、アドレス全体の入出力のパフォーマンスが軽減されることがあります。これは、ページングシステム上の追加ストレージから送信されたメッセージがすべてのキューにあるためです。

バグの報告

18.6. DIVERTS

*迂回は **HornetQ** で設定されるオブジェクトです。メッセージをあるアドレス（メッセージがルーティングされる）から他のアドレスに迂回するのに役立ちます。迂回はサーバー設定ファイル（**standalone.xml** および **domain.xml**）で設定できます。*

迂回は、以下のタイプに分類できます。

- *特別な迂回**:** メッセージは新しいアドレスにのみ迂回され、古いアドレスには送信されません*
- *特別でない迂回： メッセージは古いアドレスに引き続き送信され、そのコピーは新しいアドレスに送信されます。特別でない迂回はメッセージのフローを分割するために使用できます。*

迂回は、トランスフォーマーとオプションのメッセージフィルターを適用するように設定できます。オプションのメッセージフィルターは、指定されたフィルターに一致するメッセージのみを迂回するのに役立ちます。トランスフォーマーは、メッセージを別のフォームに変換するために使用されます。トランスフォーマーが指定されている場合、迂回されたメッセージはすべて トランスフォーマーによって変換されます。

迂回により、メッセージは同じサーバー内のアドレスにのみ迂回されます。別のサーバーのアドレスにメッセージを迂回する必要がある場合は、以下で説明されているパターンに従います。

- *メッセージをローカルストアと転送キューに迂回します。そのキューから消費して別のサーバーのアドレスにメッセージを送信するブリッジを設定*

迂回とブリッジを組み合わせてさまざまなルーティングを作成できます。

バグの報告

18.6.1. 特別な迂回

特別な迂回により、すべてのメッセージを古いアドレスから新しいアドレスに迂回します。一致するメッセージは古いアドレスにルーティングされません。**standalone.xml** および **domain.xml** サーバー設定ファイルに **exclusive** 属性を **true** に設定すると、特別な迂回を有効にすることができます。

以下の例は、サーバー設定ファイルで設定された特別な迂回を示しています。

```
<divert name="prices-divert">
  <address>jms.topic.priceUpdates</address>
  <forwarding-address>jms.queue.priceForwarding</forwarding-address>
  <filter string="office='New York'"/>
  <transformer-class-name>
    org.hornetq.jms.example.AddForwardingTimeTransformer
  </transformer-class-name>
  <exclusive>true</exclusive>
</divert>
```

以下のリストは、上記の例で使用された属性を示しています。

- アドレス: このアドレスに送信されたメッセージは別のアドレスに迂回されます。
- **forwarding-address:** メッセージは古いアドレスからこのアドレスに迂回されます。
- **filter-string: filter-string** 値に一致するメッセージは迂回されます。その他のメッセージはすべて通常のアドレスにルーティングされます。
- **transformer-class-name:** このパラメーターを指定すると、一致する各メッセージの変換が実行されます。これにより、迂回する前にメッセージのボディーまたはプロパティーを変更できます。
- 排他的： 特別な 迂回を有効または無効にするために使用されます。

バグの報告

18.6.2. 特別でない迂回

特別でない迂回は、元のメッセージのコピーを新しいアドレスに転送します。元のメッセージは古いアドレスに到達し続けます。**standalone.xml** および **domain.xml** サーバー設定ファイルで **exclusive** プロパティーを **false** に設定すると、特別でない迂回を設定できます。

以下の例は、特別でない迂回を示しています。

```
<divert name="order-divert">
  <address>jms.queue.orders</address>
  <forwarding-address>jms.topic.spyTopic</forwarding-address>
  <exclusive>false</exclusive>
</divert>
```

上記の例では、**jms.queue.orders** アドレスに送信されたすべてのメッセージのコピーを作成し、**jms.topic.spyTopic** アドレスに送信します。

バグの報告

18.7. クライアントクラスパス

HornetQ では、クライアントが **HornetQ Core API**、**JMS**、または **JNDI** を使用するかどうかに応じて、クライアントクラスパスにいくつかの **jar** が必要です。

警告

ここに記載されている **jar** はすべて **HornetQ** ディストリビューションの **EAP_HOME/bin/client** ディレクトリーにあります。リリースの正しいバージョンの **jar** のみを使用するようにしてください。異なる **HornetQ** バージョンからバージョンの **jar** を混在させたり、一致させたりしないでください。異なる **jar** バージョンの混在と一致により、少しエラーが発生してエラーが発生する可能性があります。

クライアントクラスパスを設定するには、**EAP_HOME/bin/client/jboss-client.jar** を追加します。**EAP_HOME/bin/client/README-EJB-JMS.txt** の説明どおりに、**Maven** の依存関係設定を使用することもできます。

バグの報告

18.8. 設定

18.8.1. JMS サーバーの設定

HornetQ に **JMS** サーバーを設定するには、サーバー設定ファイルを編集します。サーバー設定は、ドメインサーバーの **EAP_HOME/domain/configuration/domain.xml** ファイル、スタンドアロンサーバーの **EAP_HOME/standalone/configuration/standalone-full.xml** ファイルに含まれます。

サーバー設定ファイルの **< subsystem xmlns="urn:jboss:domain:messaging:1.4 ">** 要素にはすべての **JMS** 設定が含まれます。**JNDI** に必要な **JMS ConnectionFactory**、**Queue**、または **Topic** インスタンスを追加します。

1. **JBoss EAP 6** で **JMS** サブシステムを有効にします。

   要素内に **<extensions>** 以下の行が存在し、コメントアウトされていないことを確認します。

   ```
   <extension module="org.jboss.as.messaging"/>
   ```

2. 基本の **JMS** サブシステムを追加します。

   メッセージングサブシステムが設定ファイルに存在しない場合は、追加します。

   a. 使用する プロファイルに対応する **<profile>** を検索し、その **< subsystems>** タグを 見つけます。

   b. **< profile>** タグの直後に以下の **XML** を 貼り付けます。

   ```
   <subsystem xmlns="urn:jboss:domain:messaging:1.4">
      <hornetq-server>
      <!-- ALL XML CONFIGURATION IS ADDED HERE -->
      </hornetq-server>
   </subsystem>
   ```

その他の設定はすべて、その上の空いている行に追加します。

3. **JMS** の基本設定を追加します。

< subsystem xmlns="urn:jboss:domain:messaging:1.4"> <hornetq-server > タグの後に空の行に以下の **XML** を追加します。

```
<journal-min-files>2</journal-min-files>
<journal-type>NIO</journal-type>
<persistence-enabled>true</persistence-enabled>
```

ニーズに合わせて上記の値を変更します。

警告

journal-file-size の値は **min-large-message-size**（デフォルトでは **100KiB**）以上の値である必要があります。そうでないと、サーバーはメッセージを保存できません。

4. **HornetQ** に接続ファクトリーインスタンスを追加します。

クライアントは **JMS ConnectionFactory** オブジェクトを使用してサーバーに接続します。**JMS** 接続ファクトリーオブジェクトを **HornetQ** に追加するには、以下のように各接続ファクトリーに単一の **< jms-connection-factories >** タグと **< connection-factory >** 要素が含まれます。

```
<jms-connection-factories>
  <connection-factory name="InVmConnectionFactory">
      <connectors>
          <connector-ref connector-name="in-vm"/>
      </connectors>
      <entries>
          <entry name="java:/ConnectionFactory"/>
      </entries>
  </connection-factory>
  <connection-factory name="RemoteConnectionFactory">
      <connectors>
          <connector-ref connector-name="netty"/>
      </connectors>
      <entries>
          <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
      </entries>
```

```
    </connection-factory>
    <pooled-connection-factory name="hornetq-ra">
        <transaction mode="xa"/>
        <connectors>
            <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
            <entry name="java:/JmsXA"/>
        </entries>
    </pooled-connection-factory>
</jms-connection-factories>
```

5. ***netty*** *コネクターおよびアクセプターの設定*

*この **JMS** 接続ファクトリーは **netty** アクセプターおよびコネクターを使用します。これらは、サーバー設定ファイルにデプロイされたコネクターおよびアクセプターオブジェクトへの参照です。コネクターオブジェクトは、**HornetQ** サーバーへの接続に使用されるトランスポートとパラメーターを定義します。アクセプターオブジェクトは、**HornetQ** サーバーが受け入れる接続タイプを識別します。*

***Netty** コネクターを設定する に は、以下の設定を追加します。*

```
<connectors>
    <netty-connector name="netty" socket-binding="messaging"/>
    <netty-connector name="netty-throughput" socket-binding="messaging-throughput">
        <param key="batch-delay" value="50"/>
    </netty-connector>
    <in-vm-connector name="in-vm" server-id="0"/>
</connectors>
```

***netty** アクセプターを設定するには、以下の設定を追加します。*

```
<acceptors>
    <netty-acceptor name="netty" socket-binding="messaging"/>
    <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
        <param key="batch-delay" value="50"/>
        <param key="direct-deliver" value="false"/>
    </netty-acceptor>
    <in-vm-acceptor name="in-vm" server-id="0"/>
</acceptors>
```

6. *設定を確認します。*

これまでの手順に従った場合、メッセージサブシステムは以下のようになるはずです。

```
<subsystem xmlns="urn:jboss:domain:messaging:1.4">
    <hornetq-server>
        <journal-min-files>2</journal-min-files>
        <journal-type>NIO</journal-type>
        <persistence-enabled>true</persistence-enabled>
```

```
        <jms-connection-factories>
            <connection-factory name="InVmConnectionFactory">
                <connectors>
                    <connector-ref connector-name="in-vm"/>
                </connectors>
                <entries>
                    <entry name="java:/ConnectionFactory"/>
                </entries>
            </connection-factory>
            <connection-factory name="RemoteConnectionFactory">
                <connectors>
                    <connector-ref connector-name="netty"/>
                </connectors>
                <entries>
                    <entry name="java:jboss/exported/jms/RemoteConnectionFactory"/>
                </entries>
            </connection-factory>
            <pooled-connection-factory name="hornetq-ra">
                <transaction mode="xa"/>
                <connectors>
                    <connector-ref connector-name="in-vm"/>
                </connectors>
                <entries>
                    <entry name="java:/JmsXA"/>
                </entries>
            </pooled-connection-factory>
        </jms-connection-factories>
        <connectors>
            <netty-connector name="netty" socket-binding="messaging"/>
            <netty-connector name="netty-throughput" socket-binding="messaging-throughput">
                <param key="batch-delay" value="50"/>
            </netty-connector>
            <in-vm-connector name="in-vm" server-id="0"/>
        </connectors>
        <acceptors>
            <netty-acceptor name="netty" socket-binding="messaging"/>
            <netty-acceptor name="netty-throughput" socket-binding="messaging-throughput">
                <param key="batch-delay" value="50"/>
                <param key="direct-deliver" value="false"/>
            </netty-acceptor>
            <in-vm-acceptor name="in-vm" server-id="0"/>
        </acceptors>
    </hornetq-server>
</subsystem>
```

7. ソケットバインディンググループを設定します。

    Netty コネクターはメッセージングおよび **messaging -throughput** ソケットバインディングを参照します。メッセージング ソケットバインディングはポート **5445** を使用し、**messaging-throughput** ソケットバインディングはポート **5455** を使用します。 **&**

lt;socket-binding-group> タグはサーバー設定ファイルの個別のセクションにあります。**<socket -binding-groups>** 要素に以下のソケットバインディングがあることを確認 します。

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
  ...
  <socket-binding name="messaging" port="5445"/>
  <socket-binding name="messaging-throughput" port="5455"/>
  ...
 </socket-binding-group>
```

8. キューインスタンスを **HornetQ** への追加します。

   HornetQ 向けにキューインスタンス **(**または **JMS** 宛先**)** を設定する方法は **4** つあります。

   - 管理コンソールの使用

     管理コンソールを使用するには、**Message-Enabled** モードでサーバーを起動している必要があります。これは、**-c** オプションを使用して、**standalone-full.xml** （スタンドアロンサーバーの場合）設定ファイルの使用を強制することで実行できます。たとえば、スタンドアロンモードでは、以下はメッセージ **enabled** モードでサーバーを起動します。

     ```
     ./standalone.sh -c standalone-full.xml
     ```

     サーバーが起動したら、管理コンソールにログインして **Configuration** タブを選択します。**Subsystems** メニューを展開し、**Messaging** メニューを展開し、**Destinations** をクリックします。**JMS Messaging Provider** テーブルの **Default** の横にある 表示 をクリックし、**Add** をクリックして **JMS** 宛先の詳細を入力します。

   - 管理 **CLI** の使用**:**

     最初に、管理 **CLI** へ接続します。

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

     次に、メッセージングサブシステムに移動します。

     ```
     cd /subsystem=messaging/hornetq-server=default
     ```

最後に、**add** 操作を実行します。以下の例の値は独自の値に置き換えてください。

```
./jms-queue=testQueue:add(durable=false,entries=
["java:jboss/exported/jms/queue/test"])
```

- **JMS** 設定ファイルの作成および **deployments** フォルダーへの追加

JMS 設定ファイル **example-jms.xml** を作成して開始します。以下のエントリーを追加し、値を独自の値に置き換えます。

```
<?xml version="1.0" encoding="UTF-8"?>       <messaging-deployment
xmlns="urn:jboss:messaging-deployment:1.0">
   <hornetq-server>
      <jms-destinations>
         <jms-queue name="testQueue">
            <entry name="queue/test"/>
            <entry name="java:jboss/exported/jms/queue/test"/>
         </jms-queue>
         <jms-topic name="testTopic">
            <entry name="topic/test"/>
            <entry name="java:jboss/exported/jms/topic/test"/>
         </jms-topic>
      </jms-destinations>
   </hornetq-server>
</messaging-deployment>
```

このファイルを **deployments** フォルダーに保存し、デプロイメントを実行します。

- **JBoss EAP 6** の設定ファイルにエントリーを追加します。

キュー属性は **2** つの方法で設定できます。

  - **JMS** レベルの設定

以下は、**standalone.xml** または **domain.xml** 設定ファイルで事前定義されたキューを示しています。

```
<jms-queue name="selectorQueue">
 <entry name="/queue/selectorQueue"/>
```

```
 <selector string="color='red'"/>
 <durable>true</durable>
</jms-queue>
```

queue のこの **name** 属性はキューの名前を定義します。**jms** レベルでこれを行う場合、コアキューの実際の名前が **jms.queue.selectorQueue** になるように命名規則に従います。

entry 要素は、キューを **JNDI** にバインドするために使用される名前を設定します。これは必須要素であり、キューには複数のものを含めることができ、同じキューを異なる名前にバインドできます。

selector 要素は、事前定義されたキューが持つ **JMS** メッセージセレクターを定義します。セレクターに一致するメッセージのみがキューに追加されます。これは、省略した場合のデフォルト **null** を持つオプションの要素です。

durable 要素はキューを永続化するかどうかを指定します。これは再度オプションであり、省略されている場合はデフォルトで **true** に設定されます。

- コアレベルでの設定

キューは、**standalone.xml** または **domain.xml** ファイルのコアレベルで事前定義できます。以下に例を示します。

```
<core-queues>
 <queue name="jms.queue.selectorQueue">
  <address>jms.queue.selectorQueue</address>
  <filter string="color='red'"/>
  <durable>true</durable>
 </queue>
</core-queues>
```

9. 追加の設定を実行します。

追加の設定が必要な場合は、**EAP_HOME/docs/schema/jboss-as-messaging_1_4.xsd** の **DTD** を確認します。

バグの報告

18.8.2. JMS アドレスの設定

JMS サブシステムには、メッセージの配信方法およびタイミング、試行回数、メッセージの有効期限などの側面を制御するいくつかの設定可能なオプションがあります。これらの設定オプションはすべて、< address-settings > 設定要素内に存在します。

アドレス設定の一般的な機能は、ワイルドカードとも呼ばれる複数のアドレスに一致する構文です。

ワイルドカード構文

*アドレスワイルドカードを使用すると、1 つのステートメントで複数の同様のアドレスを照合できます。これは、システムがアスタリスク(*)文字を使用して 1 つの検索で複数のファイルまたは文字列を照合するのと似ています。以下の文字はワイルドカードステートメントに特別な意味を持ちます。*

表18.5 JMS ワイルドカード構文

文字	説明
. (シングルピリオド)	ワイルドカード式で単語と単語の間を示す。
# (ポンドまたはハッシュ記号)	0 個以上の連続する単語に相当する。
* (アスタリスク)	1 つの単語と一致します。

表18.6 JMS ワイルドカードの例

例	説明
news.europe.#	**news.europe**、**news.europe.sport**、**news.europe.politic** と一致しますが、**news.usa** や **europe** は一致しない。
news.*	**news.europe** と一致しますが、**news.europe.sport** は一致しない。
news.*.sport	**news.europe.sport** と **news.usa.sport** と一致しますが、**news.europe.politics** は一致しない。

例18.2 デフォルトアドレス設定

この例の値は、このトピックの残りを説明するために使用されます。

```
<address-settings>
   <!--default for catch all-->
   <address-setting match="#">
      <dead-letter-address>jms.queue.DLQ</dead-letter-address>
```

```
        <expiry-address>jms.queue.ExpiryQueue</expiry-address>
        <redelivery-delay>0</redelivery-delay>
        <max-size-bytes>10485760</max-size-bytes>
        <address-full-policy>BLOCK</address-full-policy>
        <message-counter-history-day-limit>10</message-counter-history-day-limit>
    </address-setting>
</address-settings>
```

表18.7 JMS アドレス設定の説明

要素	説明	デフォルト値	Type
address-full-policy	max-size-bytes が指定されたアドレスが満杯の場合に何が起こるかを決定します。	PAGE	STRING
dead-letter-address	デッドレターアドレスが指定された場合、**max-delivery-attempts** 配信試行に失敗した場合、メッセージはデッドレターアドレスに移動します。それ以外の場合、これらの未配信メッセージは破棄されます。ワイルドカードは許可されます。	jms.queue.DLQ	STRING
expiry-address	期限切れアドレスが存在する場合、期限切れのメッセージは破棄されずに、一致するアドレスに送信されます。ワイルドカードは許可されます。	jms.queue.ExpiryQueue	STRING
last-value-queue	キューが最後の値のみを使用するかどうかを定義します。	false	BOOLEAN
max-delivery-attempts	メッセージを **dead-letter-address** に送信するか、破棄する前にメッセージを再配信する最大回数。	10	INT
max-size-bytes	最大バイトサイズ。	10485760L	LONG
message-counter-history-day-limit	メッセージカウンター履歴の日数制限。	10	INT
page-max-cache-size	ページングナビゲーション中に IO を最適化するためにメモリー内に保持するページファイルの数。	5	INT

要素	説明	デフォルト値	Type
page-size-bytes	ページングサイズ。	5	INT
redelivery-delay	メッセージの再配信試行間の遅延時間（ミリ秒単位）。**0** に設定すると、再配信が無限に試行されます。	0L	LONG
redistribution-delay	メッセージを再配信する前に最後のコンシューマーがキューで閉じられたときの待機時間を定義します。	-1L	LONG
send-to-dla-on-no-route	キューにルーティングされず、そのアドレスに指定された無効なレターアドレス (DLA) に送信されたメッセージの条件を設定するアドレスのパラメーター。	false	BOOLEAN
slow-consumer-threshold	コンシューマーが「低速」とみなされる前に許可されるメッセージ消費の最小レート。1秒あたりのメッセージで測定されます。	-1	INT
slow-consumer-policy	低速なコンシューマーが検出されたときに何が起こるか。**KILL** はコンシューマーの接続を強制終了します（これは、同じ接続を使用する他のクライアントスレッドに明らかに影響を与えます）。**NOTIFY** は、アプリケーションが対応してアクションを実行できる CONSUMER_SLOW 管理通知を送信します。		STRING
slow-consumer-check-period	特定のキューで低速なコンシューマーをチェックする頻度。測定（秒単位）。	5	INT

- *アドレス設定とパターン属性の設定*

  *管理 **CLI** または管理コンソールを選択して、必要に応じてパターン属性を設定します。*

    - *管理 **CLI** を使用したアドレス設定*

- 管理 **CLI** を使用したアドレス設定

  管理 **CLI** を使用してアドレスを設定します。

  a. 新しいパターンの追加

  必要に応じて **add** 操作を使用して新規アドレス設定を作成します。管理 **CLI** セッションのルートからこのコマンドを実行できます。以下の例では、**patternname** という名前の新しいパターンを作成し、**max-delivery-attempts** 属性を **5** として宣言します。完全な プロファイルで編集しているスタンドアロンサーバーと管理対象ドメインの両方の例が表示されます。

  ```
  [standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
  ```

  ```
  [domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:add(max-delivery-attempts=5)
  ```

  b. パターン属性の編集

  書き込み 操作を使用して、新しい値を属性に書き込みます。タブ補完を使用すると、入力時のコマンド文字列の補完に役立つほか、利用可能な属性を明らかにできます。以下の例は、**max-delivery-attempts** の値を **10**に更新します。

  ```
  [standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
  ```

  ```
  [domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:write-attribute(name=max-delivery-attempts,value=10)
  ```

  c. パターン属性の確認

  値が変更されたことを確認するには、**include-runtime=true** パラメーターを指定して **read-resource** 操作を実行し、サーバーモデルでアクティブな現在の値をすべて公開します。

  ```
  [standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default/address-setting=patternname/:read-resource
  ```

  ```
  [domain@localhost:9999 /] /profile=full/subsystem=messaging/hornetq-server=default/address-setting=patternname/:read-resource
  ```

- 管理コンソールを使用したアドレスの設定

管理コンソールを使用してアドレスを設定します。

a.
管理対象ドメインまたはスタンドアロンサーバーの管理コンソールにログインします。

b.
画面上部の **Configuration** タブを選択します。ドメインモードの場合は、左上の **Profile** メニューからプロファイルを選択します。**full** および **full-ha** プロファイルのみが **messaging** サブシステムが有効になっています。

c.
Messaging メニューを展開し、**Destinations** を選択します。

d.
JMS プロバイダーの一覧が表示されます。デフォルト設定では、**default** という **1** つのプロバイダーのみが表示されます。**View** をクリックして、このプロバイダーの詳細な設定を表示します。

e.
Address Settings タブをクリックします。**Add** をクリックして新しいパターンを追加するか、既存のパターンを選択して**Edit** をクリックして設定を更新します。

f.
新しいパターンを追加する場合、**Pattern** フィールドは **address-setting** 要素の **match** パラメーターを参照します。デッドレターアドレス、**Expiry Address** 、**Redelivery Delay**、および **Max Delivery Attempts** を編集することもできます。その他のオプションは、管理 **CLI** を使用して設定する必要があります。

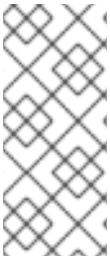
バグの報告

18.8.3. 一時キューとランタイムキュー

リクエストを送信して応答を待つクライアントを含むリクエストリプライシナリオを設計しますが、アドレス指定可能な問題は、クライアントの各ランタイムインスタンスが応答専用のキューを持っているか、ランタイムインスタンスが共有キューにアクセスするかどうかに関わらず、適切な属性に基づいて特定の応答メッセージを選択します。複数のキューが必要な場合、クライアントで使用するために動的にキューを作成する機能が必要で、**JMS** は一時キューの概念を使用してこの機能を提供します。**TemporaryQueue** は、**QueueSession** がリクエストに作成され、**QueueConnection** （**QueueConnection** が閉じられるまで）が削除されるまで存在します。つまり、**TemporaryQueue** は

特定の **QueueSession** によって作成されますが、同じ **QueueConnection** から作成されるその他の **Queue Session** で再利用でき、**QueueReceiver** を作成できます。

応答に共有キューを持つか、個別の一時キューを持つトレードオフは、アクティブなクライアントインスタンスの潜在的な数に影響されます。共有キューアプローチでは、一部のプロバイダー固有のしきい値で、キューへのアクセスの競合が懸念される可能性があります。これは、実行時にキューストレージを作成するプロバイダーに関連する追加のオーバーヘッドと、多数の一時キューをホストする必要があるマシンメモリーへの影響を比較する必要があります。



注記

一時キューの作成は、**createTemporaryQueue** メソッドで実行できます。同様に、**createTemporaryTopic** メソッドを使用して一時トピックを作成します。これらのメソッドは、共に物理キューと物理トピックを作成します。

QueueSession が終了したときに受信され、確認されていないメッセージがある場合は、これらのメッセージは保持され、コンシューマーが次にキューにアクセスするときに再配信されます。**QueueSession** は、**Point toPoint** 固有のオブジェクトを作成するために使用されます。一般的には、**Session** オブジェクトを使用します。**QueueSession** は既存のコードをサポートするために使用されます。**Session** オブジェクトを使用するとプログラミングモデルが簡素化され、トランザクションを **2** つのメッセージングドメイン全体で使用できるようになります。

TopicSession は、**Pub/Sub** 固有のオブジェクトを作成するために使用されます。通常、**Session** オブジェクトを使用し、**TopicSession** のみを使用して既存のコードをサポートします。**Session** オブジェクトを使用するとプログラミングモデルが簡素化され、トランザクションを **2** つのメッセージングドメイン全体で使用できるようになります。

バグの報告

18.8.4. last-Value キュー

last-Value キューは、明確に定義された **Last-Value** プロパティーと同じ値の新しいメッセージがキューに置かれると、メッセージを破棄する特殊なキューです。つまり、**last-Value** キューは最後の値のみを保持します。**Last-Value** キューの典型的な例は、株式価格で、特定の株式の最新値のみが対象となります。

last-Value キューの設定

last-value キューは **address-setting** 設定で定義されます。

```
<address-setting match="jms.queue.lastValueQueue">
 <last-value-queue>true</last-value-queue>
</address-setting>
```

last-Value プロパティーの使用

最後の値を識別するために使用されるプロパティー名は **"_HQ_LVQ_NAME"**（またはコア **API** の定数 **Message.HDR_LAST_VALUE_NAME** ）です。たとえば、**Last-Value** プロパティーに同じ値を持つ **2** つのメッセージが **Last-Value** キューに送信される場合、最新のメッセージのみがキューに保持されます。

例**18.3** Send 1st message with Last-Value property set to STOCK_NAME（ **STOCK_NAME**に設定された **Last-Value** プロパティーを持つ最初のメッセージを送信する）

```
TextMessage message = session.createTextMessage("1st message with Last-Value
property set");
message.setStringProperty("_HQ_LVQ_NAME", "STOCK_NAME");
producer.send(message);
```

例**18.4** Send 2nd message with Last-Value property set to STOCK_NAME

```
message = session.createTextMessage("2nd message with Last-Value property set");
message.setStringProperty("_HQ_LVQ_NAME", "STOCK_NAME");
producer.send(message);
```

例**18.5** 2 番目のメッセージのみが受信されます。最後に **Last-Value** プロパティーが設定されます。

```
TextMessage messageReceived = (TextMessage)messageConsumer.receive(5000);
System.out.format("Received message: %s\n", messageReceived.getText());
```

バグの報告

18.8.5. コアおよび JMS 宛先

HornetQ コアには **JMS** トピックの概念はありません。**JMS** トピックは、**0** 個以上のキューがバインドされたアドレス **(**トピック名**)** としてコアに実装されます。このアドレスにバインドされる各キューが、トピックサブスクリプションを表します。同様に、**JMS** キューはアドレス（**JMS** キュー名）として実装されますが、**JMS** キューを表す単一のキューがバインドされます。

JMS トピックは、ユーザーが JMS メッセージングのパブリッシュ/サブスクライブモデルでプロデューサーから特定のメッセージを受信するためにサブスクライブするチャネルです。

JMS キューは、特定のトピックでメッセージを自動的に受信するのではなく、ポイントツーポイント(p2p)モデルを使用して受信するユーザーの「プル」メッセージを示すチャネルです。プロデューサーはメッセージをキューに送信し、受信側はキューを参照し、受信するメッセージを判別できます。p2p モデルでは、配信を受け入れるかどうかを決定する前に、キューに保持されているメッセージの内容を確認できます。

名前が orders.europe の JMS Queue の設定を行うには、対応するコアキュー jms.queue.orders.europe を設定する必要があります。

```
<!-- expired messages in JMS Queue "orders.europe" will be sent to the JMS Queue
"expiry.europe" -->
<address-setting match="jms.queue.orders.europe">
 <expiry-address>jms.queue.expiry.europe</expiry-address>
 ...
</address-setting>
```

バグの報告

18.8.6. JMS メッセージセレクター

メッセージングアプリケーションが受信するメッセージをフィルターする必要がある場合は、JMS API メッセージセレクターを使用して、メッセージコンシューマーが対象のメッセージを指定できるようになります。メッセージセレクターは、メッセージをアプリケーションではなく JMS プロバイダーにフィルタリングする作業を割り当てます。

メッセージセレクターは以下のように定義できます。

```
@MessageDriven(name = "MDBMessageSelectorExample", activationConfig =
{
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
"queue/testQueue"),
  @ActivationConfigProperty(propertyName = "messageSelector", propertyValue = "color =
'RED'")
```

```
})
@TransactionManagement(value= TransactionManagementType.CONTAINER)
@TransactionAttribute(value= TransactionAttributeType.REQUIRED)
public class MDBMessageSelectorExample implements MessageListener
{
  public void onMessage(Message message)....
}
```

バグの報告

18.8.7. HornetQ でのメッセージングの設定

JBoss EAP 6 でメッセージングを設定する方法として、管理コンソールまたは管理 CLI が推奨されます。standalone.xml または domain.xml 設定ファイルを手動で編集しなくても、これらの管理ツールのいずれかで永続的な変更を行うことができます。ただし、デフォルトの設定ファイルのメッセージングコンポーネントを理解すると便利です。ここでは、管理ツールを使用したドキュメントの例では、参照用の設定ファイルスニペットが提供されます。

バグの報告

18.8.8. HornetQ のロギングの有効化

EAP 6.x で HornetQ のロギングを有効にするには、以下の方法の 1 つを使用します。

- サーバー設定ファイル（standalone-full.xml および standalone-full-ha.xml）の手動での編集
- CLI を使用してサーバー設定ファイルを編集します。

手順18.1 サーバー設定ファイルを手作業で編集して HornetQ ロギングを設定する

1. サーバー設定ファイルを開いて編集します。例： standalone-full.xml および standalone-full-ha.xml
2. ファイルの logging サブシステム設定に移動します。デフォルト設定は以下のようになります。

```
<logger category="com.arjuna">
 <level name="TRACE"/>
</logger>
...
<logger category="org.apache.tomcat.util.modeler">
 <level name="WARN"/>
</logger>
....
```

3. 以下の例のように、**org.hornetq** ロガーカテゴリーと必要なロギングレベルを追加します。

```
<logger category="com.arjuna">
  <level name="TRACE"/>
</logger>
...
<logger category="org.hornetq">
  <level name="INFO"/>
</logger>
....
```

結果

HornetQ ロギングが有効になり、設定されたログレベルを基にログメッセージが処理されます。

***CLI** を使用してサーバー設定ファイルを編集して **HornetQ** ロギングを設定する*

CLI を使用して、希望のロギングレベルとともに **org.hornetq** ロガーカテゴリーをサーバー設定ファイルに追加することもできます。詳細は以下を参照してください。 「**CLI** でのログカテゴリー設定」

バグの報告

18.8.9. HornetQ Core Bridge の設定

*例**18.6 HornetQ Core Bridge** の設定例*

この例の値は、このトピックの残りを説明するために使用されます。

```
<bridges>
  <bridge name="myBridge">
      <queue-name>jms.queue.InQueue</queue-name>
      <forwarding-address>jms.queue.OutQueue</forwarding-address>
 <ha>true</ha>
      <reconnect-attempts>-1</reconnect-attempts>
      <use-duplicate-detection>true</use-duplicate-detection>
      <static-connectors>
       <connector-ref>
            bridge-connector
            </connector-ref>
      </static-connectors>
      </bridge>
</bridges>
```

表18.8 HornetQ Core Bridge の属性

属性	説明
name	ブリッジの名前はすべてサーバー上で一意となる必要があります。
queue-name	必須パラメーターは、ブリッジが消費するローカルキューの一意の名前です。ブリッジが起動時にインスタンス化される時点でキューが存在している必要があります。
forwarding-address	これは、メッセージが転送されるターゲットサーバーのアドレスです。転送アドレスが指定されていない場合、メッセージの元のアドレスは保持されます。
ha	このオプションのパラメーターは、このブリッジが高可用性をサポートするべきかどうかを決定します。**true** これは、クラスター内の利用可能なサーバーに接続し、フェイルオーバーをサポートすることを意味します。デフォルト値は false です。
reconnect-attempts	このオプションのパラメーターは、断念してシャットダウンするまでにブリッジが行う再接続試行の総回数を決定します。-1 の値は、無制限の試行回数を示します。デフォルト値は -1 です。
use-duplicate-detection	任意のパラメーターで、ブリッジが転送する各メッセージに複製の ID プロパティーを自動的に挿入するかどうかを決定します。
static-connectors	static-connectors は、他の場所で定義されるコネクター要素を参照する connector-ref 要素のリストです。コネクターは、使用するトランスポート（TCP、SSL、HTTP など）とサーバー接続パラメーター（ホスト、ポートなど）の知識をカプセル化します。

バグの報告

18.8.10. JMS ブリッジの設定

HornetQ には、完全に機能する **JMS** メッセージブリッジが含まれています。このブリッジの機能は、ソースキューまたはトピックからメッセージを消費し、通常は別のサーバーにあるターゲットキューまたはトピックに送信することです。

ソースやターゲットサーバーは同じクラスターにある必要はないため、ブリッジングは異なるクラスターの間 **(WAN** など**)** や接続が信頼できない場所でメッセージを確実に送信するのに適しています。

ブリッジは、**HornetQ** スタンドアロンサーバーまたは **JBoss AS** インスタンス内部を使用してスタンドアロンアプリケーションとしてデプロイできます。ソースとターゲットは同じ仮想マシンまたは別の仮想マシンに置くことができます。

例**18.7 JMS** ブリッジの設定例

この例の値は、このトピックの残りを説明するために使用されます。

```
<subsystem>
  <subsystem xmlns="urn:jboss:domain:messaging:1.3">
            <hornetq-server>
            ...
            </hornetq-server>

            <jms-bridge name="myBridge">
              <source>
                 <connection-factory name="ConnectionFactory"/>
                 <destination name="jms/queue/InQueue"/>
              </source>
              <target>
                 <connection-factory name="jms/RemoteConnectionFactory"/>
                 <destination name="jms/queue/OutQueue"/>
                 <context>
                    <property key="java.naming.factory.initial"
value="org.jboss.naming.remote.client.InitialContextFactory"/>
                    <property key="java.naming.provider.url" value="remote://192.168.40.1:4447"/>
                 </context>
              </target>
              <quality-of-service>AT_MOST_ONCE</quality-of-service>
              <failure-retry-interval>1000</failure-retry-interval>
              <max-retries>-1</max-retries>
              <max-batch-size>10</max-batch-size>
              <max-batch-time>100</max-batch-time>
              <add-messageID-in-header>true</add-messageID-in-header>
```

```
        </jms-bridge>
    ...
</subsystem>
```

警告

***JMS** ブリッジは再接続処理に独自の **max-retries** パラメーターがあるため、**reconnect-attempts** パラメーターを設定（または **0**に設定）しない接続ファクトリーを使用することが推奨されます。これにより、再接続時間が長くなる可能性がある競合の発生を防ぐことができます。*

表18.9 HornetQ Core JMS 属性

属性	説明
name	ブリッジの名前はすべてサーバー上で一意となる必要があります。
source connection-factory	SourceCFF Bean（beans ファイルにも定義されます）をインジェクトします。この Bean はソースの ConnectionFactory を作成します。
source destination name	SourceDestinationFactory Bean（beans ファイルにも定義されます）をインジェクトします。この Bean はソースの Destination を作成します。
target connection-factory	TargetCFF Bean（beans ファイルにも定義されます）をインジェクトします。この Bean はターゲット ConnectionFactory を作成します。
target destination name	TargetDestinationFactory Bean（beans ファイルにも定義されます）をインジェクトします。この Bean はターゲット Destination を作成します。
quality-of-service	このパラメーターは、必要なサービスモードの品質を示しています。使用できる値は AT_MOST_ONCE、DUPLICATES_OK、ONCE_AND_ONLY_ONCE です。
failure-retry-interval	ブリッジによって接続の失敗が検出されたときに、ソースまたはターゲットサーバーへの接続を再試行する前に待機する期間 (ミリ秒単位) を表します。

属性	説明
max-retries	ブリッジで障害が発生したことを検出したときに、ソースサーバーまたはターゲットサーバーへの接続の再作成を試行する回数を表します。この回数の試行後にブリッジが試行されます。-1 は「try forever」を表します。
max-batch-size	バッチでターゲット宛先に送信する前にソース宛先から消費するメッセージの最大数を表します。この値は >= 1 にする必要があります。
max-batch-time	消費されたメッセージの数が MaxBatchSize に達していない場合でも、ターゲットへバッチを送信するまで待機する最大時間（ミリ秒単位）を表します。実際の時間を指定するには、値が -1 である必要があります。「wait forever」または 1 以上（実際の時間）を表す必要があります。
add-messageID-in-header	true の場合、元のメッセージのメッセージ ID がヘッダー HORNETQ_BRIDGE_MSG_ID_LIST 内の宛先に送信されるメッセージに追加されます。メッセージが複数回ブリッジされる場合は、各メッセージ ID が追加されます。これにより、分散リクエスト/応答パターンを使用できます。 メッセージを受信するとき、最初のメッセージ ID の相関 IDを使用して応答を送信できます。そのため、元の送信側がメッセージを受信すると簡単に関連付けできます。

注記

***quality-of-service** 属性が **ONCE_AND_ONLY_ONCE** に設定されているデプロイ済みの **JMS** ブリッジを持つサーバーをシャットダウンする場合は、予期しない例外を防ぐために **JMS** ブリッジを持つサーバーを最初にシャットダウンします。*

*詳しい手順については、「**JMS** ブリッジの作成」 を参照してください。*

バグの報告

18.8.11. 遅延再配信の設定

はじめに

*遅延再配信は、**Java Messaging Service(JMS)**サブシステム設定の **< address-setting>** 設定要素の子要素である **<redelivery - delay>** 要素で定義されます。*

```
<!-- delay redelivery of messages for 5s -->
<address-setting match="jms.queue.exampleQueue">
  <redelivery-delay>5000</redelivery-delay>
</address-setting>
```

再配信の遅延が指定されると、**JMS** システムはこの遅延期間待機した後にメッセージを再配信します。**&lt ;redelivery-delay>** を **0** に設定すると、再配信の遅延はありません。アドレスワイルドカードは **< address- match >** 要素の **match** 属性で使用し、ワイルドカードに一致するアドレスの再配信遅延を設定できます。

バグの報告

18.8.12. デッドレターアドレスの設定

はじめに

デッドレターアドレスは、**Java Messaging Service(JMS)**サブシステム設定の **<address-setting >** 要素で定義されます。

```
<!-- undelivered messages in exampleQueue will be sent to the dead letter address
deadLetterQueue after 3 unsuccessful delivery attempts
-->
<address-setting match="jms.queue.exampleQueue">
  <dead-letter-address>jms.queue.deadLetterQueue</dead-letter-address>
  <max-delivery-attempts>3</max-delivery-attempts>
</address-setting>
```

< dead-letter-address > が指定されていない場合、**< max-delivery-attempts>** 時間の配信を試行した後にメッセージが削除さ れます。デフォルトでは **10** 回メッセージの配信を試みます。**< max-delivery-attempts>** を **- 1** に 設定すると、再配信が無限に試行されます。たとえば、一致するアドレスのセットに対してグローバルにデッドレターを設定し、特定のアドレスに対して **< max-delivery-attempts >** を **- 1** に設定すると、このアドレスのみに対して無限再配信が試行されます。アドレスワイルドカードを使用して、アドレスのセットにデッドレターを設定することもできます。

バグの報告

18.8.13. メッセージ期限切れアドレス

はじめに

メッセージ期限切れアドレスは **Java Messaging Service(JMS)**の **address-setting** 設定で定義されます。以下に例を示します。

```
<!-- expired messages in exampleQueue will be sent to the expiry address expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
  <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

メッセージの期限が切れ、期限切れアドレスが指定されていない場合、メッセージは単にキューから削除されドロップされます。アドレスワイルドカード を使用して、アドレスのセットに期限切れアドレスの特定の範囲を設定することもできます。**JMX** ワイルドカード構文と例は、「**JMS** アドレスの設定」 を参照してください。

バグの報告

18.8.14. フロー制御

フロー制御は、クライアントまたはサーバーがデータでオーバーロードしないように、クライアントとサーバー間のデータフローを制限するために使用されます。

- **Consumer Flow Control:** クライアントがメッセージを消費する際にサーバーとクライアント間のデータのフローを制御します。パフォーマンス上の理由から、クライアントは通常、**receive**（） メソッドを介してコンシューマーに配信したり、メッセージリスナーを介して非同期的にコンシューマーに配信する前にメッセージをバッファーします。

  レート制限のあるフロー制御: コンシューマーがメッセージを消費できるレートを制御できます。これはスロットリングの形式であり、コンシューマーが指定のレートよりも速いメッセージを消費しないようにすることができます。この機能を有効にするには、レートは正の整数である必要があります。**1** 秒あたりのメッセージ単位で指定される必要なメッセージ消費率

の最大値です。これを **-1** に設定すると、レート制限フロー制御が無効になります。デフォルト値は **-1** です。

例**18.8 JMS** を使用したレート制限フロー制御

JNDI を使用して接続ファクトリーを検索する場合は、**standalone.xml** または **domain.xml** で最大レートを設定できます。

```
<connection-factory name="ConnectionFactory">
 <connectors>
  <connector-ref connector-name="netty-connector"/>
 </connectors>
  <entries>
   <entry name="ConnectionFactory"/>
  </entries>
  <!-- We limit consumers created on this connection factory to consume messages
 at a maximum rate of 10 messages per sec -->
  <consumer-max-rate>10</consumer-max-rate>
 </connection-factory>
```

接続ファクトリーが直接インスタンス化されている場合、**HornetQConnectionFactory.setConsumerMaxRate(int consumerMaxRate)** メソッドを使用して最大レートサイズを設定できます。

- プロデューサーフロー制御： サーバーが大きすぎないように、クライアントからサーバーに送信されるデータ量をサーバーに制限します。

ウィンドウベースのフロー制御: HornetQ プロデューサーは、ジャーナルに十分なクレジットがある限り、メッセージをアドレスに送信できます。メッセージの送信に必要なクレジットの量は、メッセージのサイズで指定されます。プロデューサーはクレジットが少ないため、サーバーがより多くのクレジットを送信すると、より多くのメッセージを送信できます。プロデューサーリクエストのクレジットの量は、ウィンドウサイズ と呼ばれます。

例**18.9 JMS** を使用したプロデューサーウィンドウサイズフロー制御

JNDI を使用して接続ファクトリーを検索する場合は、プロデューサーウィンドウサイズを **standalone.xml** または **domain.xml** で設定できます。

```
<connection-factory name="ConnectionFactory">
```

```
    <connectors>
      <connector-ref connector-name="netty-connector"/>
    </connectors>
    <entries>
      <entry name="ConnectionFactory"/>
    </entries>
    <producer-window-size>10</producer-window-size>
  </connection-factory>
```

接続ファクトリーが直接インスタンス化されている場合、**HornetQConnectionFactory.setProducerWindowSize(int producerWindowSize)** メソッドを使用してプロデューサーウィンドウサイズを設定できます。

バグの報告

18.8.15. HornetQ 設定属性のリファレンス

HornetQ の **JBoss EAP 6** 実装は設定の以下の属性を公開します。管理 **CLI** を使用すると、**read-resource** 操作で設定可能または表示可能な属性を表示できます。

例**18.10 read-resource** を使用して属性を表示します。

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default:read-resource
```

表**18.10 HornetQ** 属性

属性	デフォルト値	型	説明
allow-failback	true	BOOLEAN	元のライブサーバーが復旧したときにこのサーバーを自動的にシャットダウンするかどうか。

属性	デフォルト値	型	説明
async-connection-execution-enabled	true	BOOLEAN	サーバーの受信パケットをスレッドプールからのスレッドに渡して処理するかどうか。
address-setting			特定のキューでなく、アドレスワイルドカードに対して定義された一部の属性を定義するアドレス設定。
acceptor			アクセプターは HornetQ サーバーへ接続を確立する方法を定義します。
backup-group-name		STRING	お互いをレプリケートする必要があるライブ/バックアップのセットの名前。
backup	false	BOOLEAN	このサーバーがバックアップサーバーであるかどうか。
check-for-live-server	false	BOOLEAN	同じノード ID を持つライブサーバーがすでに存在するかを確認するため、レプリケートされたライブサーバーが現在のクラスターをチェックするかどうか。
clustered	false	BOOLEAN	[廃止済み] サーバーがクラスター化されているかどうか。
cluster-password	**やり直してください！**	STRING	クラスター化されたノードの間で通信するためにクラスター接続によって使用されるパスワード。
cluster-user	HORNETQ.CLUSTER.ADMIN.USER	STRING	クラスター化されたノードの間で通信するためにクラスター接続によって使用されるユーザー。
cluster-connection			クラスター接続はサーバーをクラスターにグループ化し、クラスターのノード間でメッセージの負荷を分散します。
create-bindings-dir	true	BOOLEAN	起動時にサーバーが bindings ディレクトリーを作成するかどうか。

属性	デフォルト値	型	説明
create-journal-dir	true	BOOLEAN	起動時にサーバーが journal ディレクトリーを作成するかどうか。
connection-ttl-override	-1L	LONG	設定された場合、ping を受信せずに接続を維持する期間 (ミリ秒単位) を上書きします。
connection-factory			接続ファクトリーを定義します。
connector			サーバーへの接続方法を定義するためクライアントはコネクターを使用できます。
connector-service			
divert			クライアントアプリケーションロジックを変更せずに、あるアドレスから別のアドレスにルーティングされたメッセージを透過的に迂回できるようにするメッセージングリソースです。
discovery-group			コネクターをアナウンスする他のサーバーからブロードキャストを受信するためリッスンするマルチキャストグループ。
failback-delay	5000	LONG	ライブサーバーの再起動でフェイルバックが発生する前に待機する期間。
failover-on-shutdown	false	BOOLEAN	このバックアップサーバー (バックアップサーバーである場合) が通常のサーバーのシャットダウン時に稼働されるかどうか。
grouping-handler			クラスターのどのノードがグループ ID が割り当てられたメッセージを処理するかを決定します。

属性	デフォルト値	型	説明
id-cache-size	20000	INT	メッセージ ID を事前作成するためのキャッシュのサイズ。
in-vm-acceptor			HornetQ サーバーへ VM 内 (in-VM) 接続を確立する方法を定義します。
in-vm-connector			サーバーへの接続方法を定義するため、VM 内のクライアントによって使用されます。
jmx-domain	org.hornetq	STRING	MBeanServer で内部 HornetQ MBean を登録するために使用される JMX ドメイン。
jmx-management-enabled	false	BOOLEAN	HornetQ が内部管理 API を JMX 経由で公開するかどうか。これらの MBean にアクセスすると設定が一貫性のない可能性があるため、この方法は推奨されません。
journal-buffer-size	501760 (490KiB)	LONG	ジャーナルの内部バッファーのサイズ。
journal-buffer-timeout	ASYNCIO ジャーナルは 500000 (0.5 ミリ秒)、NIO ジャーナルは 3333333 (3.33 ミリ秒)。	LONG	ジャーナルで内部バッファーをフラッシュするのに使用されるタイムアウト (ナノ秒単位)。
journal-compact-min-files	10	INT	圧縮開始前のジャーナルデータファイルの最小数。
journal-compact-percentage	30	INT	ジャーナルの圧縮を考慮するライブデータの比率 (パーセント)。
journal-file-size	10485760	LONG	各ジャーナルファイルのサイズ (バイト単位)。

属性	デフォルト値	型	説明
journal-max-io	1	INT	一度に AIO キューに入れることのできる書き込み要求の最大数。ASYNCIO ジャーナルが使用されると、デフォルト値は 500 に変わります。
journal-min-files	2	INT	事前作成するジャーナルファイルの数。
journal-sync-non-transactional	true	BOOLEAN	クライアントに応答を返す前に、非トランザクションデータがジャーナルに同期化されるのを待つかどうか。
journal-sync-transactional	true	BOOLEAN	クライアントに応答を返す前に、トランザクションデータがジャーナルに同期化されるのを待つかどうか。
journal-type	ASYNCIO	文字列	使用するジャーナルのタイプ。この属性は「ASYNCIO」または「NIO」の値を取ることができます。
jms-topic			JMS トピックを定義します。
live-connector-ref	reference	STRING	[廃止済み] ライブコネクターへの接続に使用されるコネクターの名前。このサーバーが共有しない HA を使用するバックアップではない場合、値は「undefined」になります。
log-journal-write-rate	false	BOOLEAN	ジャーナルの書き込み率およびフラッシュ率を周期的にログに記録するかどうか。
mask-password	true	BOOLEAN	
management-address	jms.queue.hornetq.management	STRING	管理メッセージを送信する先のアドレス。

属性	デフォルト値	型	説明
management-notification-address	hornetq.notifications	STRING	管理通知を受け取るためにコンシューマーがバインドするアドレスの名前。
max-saved-replicated-journal-size	2	INT	フェイルバック発生後に保持するバックアップジャーナルの最大数。
memory-measure-interval	-1	LONG	JVM メモリーのサンプリング頻度 (ミリ秒単位）。-1 を指定するとメモリーのサンプリングが無効になります。
memory-warning-threshold	25	INT	この値を越えると警告ログが記録される使用可能なメモリーの比率 (パーセント)。
message-counter-enabled	false	BOOLEAN	メッセージカウンターが有効であるかどうか。
message-counter-max-day-history	10	INT	メッセージカウンターの履歴を保持する日数。
message-counter-sample-period	10000	LONG	メッセージカウンターに使用するサンプル周期 (ミリ秒単位)。
message-expiry-scan-period	30000	LONG	期限切れメッセージをスキャンする頻度 (ミリ秒単位)。
message-expiry-thread-priority	3	INT	メッセージを期限切れにするスレッドの優先度
page-max-concurrent-io	5	INT	ページングで許可される同時読み取りの最大数。
perf-blast-pages	-1	INT	
persist-delivery-count-before-delivery	false	BOOLEAN	配信前に配信数が永続化されるかどうか。false は、メッセージがキャンセルされた後にのみ発生することを意味します。
persist-id-cache	true	BOOLEAN	ID がジャーナルへ永続化されるかどうか。

属性	デフォルト値	型	説明
persistence-enabled	true	BOOLEAN	サーバーがファイルベースのジャーナルを永続化に使用するかどうか。
pooled-connection-factory			管理対象接続ファクトリーを定義します。
remoting-interceptors	undefined	LIST	[廃止済み] このサーバーによって使用されるインターセプタークラスのリスト。
remoting-incoming-interceptors	undefined	LIST	このサーバーによって使用される受信インターセプタークラスのリスト。
remoting-outgoing-interceptors	undefined	LIST	このサーバーによって使用される送信インターセプタークラスのリスト。
run-sync-speed-test	false	BOOLEAN	起動時にディスクが同期できる速度で診断テストを実行するかどうか。パフォーマンスの問題を特定する際に役立ちます。
replication-clustername		STRING	レプリケート元のクラスター接続の名前 (2 つ以上のクラスターが設定されている場合)。
runtime-queue			ランタイムキュー
remote-connector			サーバーへの接続方法を定義するためにリモートクライアントによって使用されます。
remote-acceptor			HornetQ サーバーへリモート接続が確立される方法を定義します。
scheduled-thread-pool-max-size	5	INT	メインのスケジュールされたスレッドプールが持つスレッドの数。
security-domain	その他	STRING	ユーザーおよびロールの情報を検証するために使用するセキュリティードメイン。

属性	デフォルト値	型	説明
security-enabled	true	BOOLEAN	セキュリティーが有効かどうか。
security-setting			セキュリティー設定は、アドレスを基にキューに対してパーミッションのセットを定義できるようにします。
security-invalidation-interval	10000	LONG	セキュリティーキャッシュを無効にする前に待機する時間 (ミリ秒単位)。
server-dump-interval	-1	LONG	基本的なランタイム情報をサーバーログにダンプする頻度。1 未満の値を指定すると、この機能が無効になります。
共有ストア	true	BOOLEAN	このサーバーがフェイルオーバーに共有ストアを使用しているかどうか。
thread-pool-max-size	30	INT	メインのスレッドプールが持つスレッドの数。-1 は無制限を意味します。
transaction-timeout	300000	LONG	作成時の後、トランザクションをリソースマネージャーから削除できるまでの時間 (ミリ秒単位)。
transaction-timeout-scan-period	1000	LONG	タイムアウトトランザクションをスキャンする頻度 (ミリ秒単位)。
wild-card-routing-enabled	true	BOOLEAN	サーバーがワイルドカードルーティングをサポートするかどうか。

警告

journal-file-size の値は、サーバーに送信されるメッセージのサイズよりも大きくなければなりません。そうでないと、サーバーはメッセージを保存できません。

バグの報告

18.8.16. メッセージの有効期限の設定

はじめに

送信されたメッセージは、指定期間（ミリ秒単位）後にコンシューマーに配信されない場合にサーバーで期限切れになるように設定できます。**Java Messaging Service(JMS)**または **HornetQ Core API** を使用すると、有効期限をメッセージに直接設定できます。以下に例を示します。

```
// message will expire in 5000ms from now
message.setExpiration(System.currentTimeMillis() + 5000);
```

JMS MessageProducer には、送信するメッセージの有効期限を制御する **TimeToLive** パラメーターが含まれています。

```
// messages sent by this producer will be retained for 5s (5000ms) before expiration
producer.setTimeToLive(5000);
```

期限切れアドレスより消費された期限切れメッセージは次のプロパティーを持っています。

- **_HQ_ORIG_ADDRESS**

期限切れメッセージの元のアドレスが含まれる文字列プロパティー。

- **_HQ_ACTUAL_EXPIRY**

期限切れメッセージの実際の失効時間が含まれるロングプロパティー。

JMS プロデューサーで **time-to-live** パラメーターを設定する他に、メッセージごとに設定することもできます。これを行うには、メッセージの送信時にプロデューサーの送信メソッドに **TTL** パラメーターを追加します。

```
producer.send(message, DeliveryMode.PERSISTENT, 0, 5000)
```

ここで、最後のパラメーターはメッセージ固有の **TTL** です。

期限切れアドレスの設定

期限切れアドレスは **address-setting** 設定で定義されます。

```
<!-- expired messages in exampleQueue will be sent to the expiry address expiryQueue -->
<address-setting match="jms.queue.exampleQueue">
   <expiry-address>jms.queue.expiryQueue</expiry-address>
</address-setting>
```

メッセージの期限が切れ、期限切れアドレスが指定されていない場合、メッセージはキューから削除されドロップされます。

期限切れリーパー (Reaper) スレッドの設定

リーパー (reaper) スレッドは、メッセージの期限切れを検証するためにキューを定期的に検査します。

- **message-expiry-scan-period**

期限切れメッセージを検出するためにキューがスキャンされる頻度 (ミリ秒単位でデフォルト値は 30000 ミリ秒)。-1 を設定するとリーパースレッドが無効になります。

- **message-expiry-thread-priority**

リーパースレッドの優先度。優先度が最も高い 0 から 9 までの値である必要があります。デフォルトは 3 です。

バグの報告

18.9. PRE_ACKNOWLEDGE モード

JMS は次の 3 つの承認モードを指定します。

- **AUTO_ACKNOWLEDGE**
- **CLIENT_ACKNOWLEDGE**
- **DUPS_OK_ACKNOWLEDGE**

HornetQ は、PRE_ACKNOWLEDGE と INDIVIDUAL_ACKNOWLEDGE の 2 つの追加モードをサポートします。

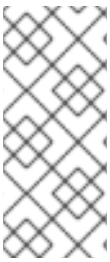
シナリオによっては、障害発生時にメッセージを失わせる可能性があるので、クライアントへ配信する前にサーバーでメッセージを確認しやすくなります。

この追加モードは **HornetQ** によってサポートされ、事前承認モードと呼ばれます。

配信前にサーバーでメッセージを承認する欠点は、サーバーでメッセージを承認した後、クライアントに配信される前にシステムがクラッシュした場合にメッセージが失われることです。この場合、システムの再起動時にメッセージが失われ、復元されません。

メッセージングのケースに応じて、事前承認モードでは、メッセージ損失による対応により、追加のネットワークトラフィックと **CPU** を回避できます。

事前承認の使用例は、株式価格更新メッセージに関連します。これらのメッセージでは、クラッシュ発生時にメッセージが失われることは適切です。これは、次の価格更新メッセージがすぐに到達し、前の価格を上書きするためです。



注記

pre-acknowledge モードを使用する場合、消費されるメッセージのトランザクションセマンティクスが失われます。これは、トランザクションのコミット時ではなく、サーバーで最初に確認されるためです。

バグの報告

18.9.1. PRE_ACKNOWLEDGE の使用

事前承認モードは、接続ファクトリーの **standalone.xml** または **domain.xml** ファイルで設定できます。

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <pre-acknowledge>true</pre-acknowledge>
</connection-factory>
```

または、**JMS API** を使用して事前承認モードを使用するには、**HornetQSession.PRE_ACKNOWLEDGE** 定数で **JMS** セッションを作成します。

```
// messages will be acknowledge on the server *before* being delivered to the client
Session session = connection.createSession(false,
HornetQJMSConstants.PRE_ACKNOWLEDGE);
```

または、**setset** メソッドを使用して **HornetQConnectionFactory** インスタンスに事前承認を直接設定できます。

コア **API** を使用して事前承認モードを使用するには、セッターメソッドを使用して **ClientSessionFactory** インスタンスに直接設定できます。

バグの報告

18.9.2. 個々の Acknowledge

個別の確認応答の有効なユースケースとして、独自のスケジュールが必要であり、メッセージ処理が完了したときにわからない場合が挙げられます。スレッドワーカーごとに **1** つのコンシューマーの使用が優先されますが、処理の複雑な設定方法によっては、状況によってはこれを行うことはできません。これにより、個別の確認応答を使用できます。

HornetQJMSConstants.INDIVIDUAL_ACKNOWLEDGE で確認モードでセッションを作成して、個別の **Acknowledge** を設定する必要があります。個々の **Acknowledge** は、クライアント **Acknowledge** からすべてのセマンティクスを継承しますが、例外はメッセージを個別に承認されます。

注記

MDB 処理における混乱を避けるため、個別の **ACKNOWLEDGE** は **MDB**（またはインバウンドリソースアダプター）ではサポートされません。これは、**MDB** 内でメッセージのプロセスを終了する必要があるためです。

バグの報告

18.10. スレッド管理

18.10.1. サーバー側のスレッド管理

各 **HornetQ** サーバーは、一般的な使用のために単一のスレッドプールを維持し、スケジュール使用のためにスケジュールスレッドプールを維持します。**Java** のスケジュールスレッドプールは標準のスレッドプールを使用するように設定できません。それ以外は、スケジュールされたアクティビティーとスケジュールされていないアクティビティーの両方に単一のスレッドプールを使用できます。

古い（ブロッキング）**IO** を使用する場合は、別のスレッドプールを使用して接続を処理します。古い **IO** は接続ごとにスレッドを必要とするため、多くの接続が行われるとプールが使い切られるため、標準プールからスレッドを取得することは推奨されません。これにより、他の処理を行うスレッドが残っていないため、サーバーがハングします。サーバーが多くの同時接続を処理する必要がある場合は、古い **IO** ではなく **NIO** を使用する必要があります。

新しい **IO(NIO)**を使用する場合、**HornetQ** はデフォルトで受信パケットを処理するために **.getRuntime().availableProcessors()Runtime** によって報告されるコア（またはハイパースレッド）の数の **3** 倍のスレッドを使用します。この値をオーバーライドするには、トランスポート設定に **nio-remoting-threads** パラメーターを指定してスレッド数を設定できます。

バグの報告

18.10.1.1. サーバースケジュールスレッドプール

サーバースケジュールスレッドプールは、定期的に実行する必要がある、または遅れて実行する必要があるサーバーサイドの多くのアクティビティーに使用されます。内部的には、**java.util.concurrent.ScheduledThreadPoolExecutor** インスタンスにマッピングします。

このプールによって使用されるスレッドの最大数は、**scheduled-thread-pool-max-size** パラメーターを使用して **standalone.xml** または **domain.xml** で設定されます。デフォルト値は **5** スレッドです。通常、このプールには少ない数のスレッドで十分です。

バグの報告

18.10.1.2. 汎用サーバースレッドプール

汎用スレッドプールは、サーバー側でほとんどの非同期操作に使用されます。内部的には、**java.util.concurrent.ThreadPoolExecutor** インスタンスにマッピングします。

このプールによって使用されるスレッドの最大数は、**thread-pool-max-size** パラメーターを使用して **standalone.xml** または **domain.xml** で設定されます。

-1 の値を使用すると、スレッドプールには上限がなく、要求を満たすのに十分なスレッドがない場合には、新しいスレッドがオンデマンドで作成されます。後でアクティビティーが従うと、スレッドはタイムアウトになり、閉じられます。

n の値（n はゼロより大きい正の整数）が使用される場合、スレッドプールはバインドされます。要求数が増え、プールに空きスレッドがなく、プールが満杯になると、スレッドが利用可能になるまでリクエストはブロックされます。バインドされたスレッドプールは、上限が低すぎると、デッドロックの状況が発生する可能性があるため、注意して使用することが推奨されます。

thread-pool-max-size のデフォルト値は 30 です。

バグの報告

18.10.1.3. 期限切れリーパースレッド

キュー内の期限が切れたメッセージをスキャンするためにサーバー側でも単一のスレッドが使用されます。このスレッドは、独自の設定可能な優先順位で実行する必要があるため、このためにスレッドプールのいずれかを使用することはできません。

バグの報告

18.10.1.4. 非同期 IO

非同期 IO には、ネイティブレイヤーからイベントを送受信するためのスレッドプールがあります。これは、接頭辞 HornetQ-AIO-poller-pool を持つスレッドダンプ上にあります。HornetQ は、ジャーナル上で開かれたファイルごとに 1 つのスレッドを使用します（通常は 1 つです）。

また、libaio で書き込みを呼び出すために使用されるスレッドも 1 つあります。これは、libaio で、パフォーマンスの問題を引き起こすコンテキスト切り替えを回避するために行われます。このスレッドは、接頭辞 HornetQ-AIO-writer-pool を持つスレッドダンプにあります。

バグの報告

18.10.2. クライアント側のスレッド管理

クライアント側で、**HornetQ** は単一の静的スケジュールスレッドプールと、その **JVM** インスタンスの同じクラスローダーを使用してすべてのクライアントによって使用される静的汎用スレッドプールを **1** つ維持します。

静的スケジュールスレッドプールの最大サイズは **5** つのスレッドで、汎用スレッドプールの最大サイズはバインドされません。

各 **ClientSessionFactory** インスタンスがこれらの静的プールを使用せず、代わりに独自のスケジュールおよび汎用プールを維持するように **HornetQ** を設定することもできます。その **ClientSessionFactory** から作成されたセッションは、代わりにそれらのプールを使用します。

ClientSessionFactory インスタンスが独自のプールを使用するように設定するには、作成直後に適切なセッターメソッドを使用します。以下に例を示します。

```
ServerLocator locator = HornetQClient.createServerLocatorWithoutHA(...)
ClientSessionFactory myFactory = locator.createClientSessionFactory();
myFactory.setUseGlobalPools(false);
myFactory.setScheduledThreadPoolMaxSize(10);
myFactory.setThreadPoolMaxSize(-1);
```

JMS API を使用している場合は、**ClientSessionFactory** に同じパラメーターを設定し、これを使用して **ConnectionFactory** インスタンスを作成できます。以下に例を示します。

```
ConnectionFactory myConnectionFactory =
HornetQJMSClient.createConnectionFactory(myFactory);
```

JNDI を使用して **HornetQConnectionFactory** インスタンスをインスタンス化する場合は、接続ファクトリーを記述する **standalone.xml** または **domain.xml** ファイルにこれらのパラメーターを設定することもできます。以下に例を示します。

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
    <entry name="XAConnectionFactory"/>
  </entries>
  <use-global-pools>false</use-global-pools>
  <scheduled-thread-pool-max-size>10</scheduled-thread-pool-max-size>
  <thread-pool-max-size>-1</thread-pool-max-size>
</connection-factory>
```

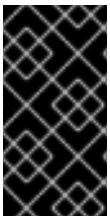
バグの報告

18.11. メッセージのグループ化

18.11.1. メッセージのグループ化

メッセージグループは、共通した特定の特徴を持つメッセージのセット/グループです。

- メッセージグループのすべてのメッセージは、共通のグループ **ID** でグループ化されます。これは、共通のグループプロパティーで識別できることを意味します。
- メッセージグループのすべてのメッセージは、キューのカスタマーの数に関係なく、同じコンシューマーによって順次処理および消費されます。そのため、一意のグループ **ID** を持つ特定のメッセージグループは、常にそのメッセージグループを開く **1** つのコンシューマーによって処理されます。コンシューマーがメッセージグループを閉じると、メッセージグループ全体がキューの別のコンシューマーに移動されます。



重要

メッセージグループは、特定のプロパティーの値 (グループ **ID**) を持つメッセージが単一のコンシューマーによって順次処理される必要がある場合に便利です。

バグの報告

18.11.2. クライアント側での HornetQ Core API の使用

_HQ_GROUP_ID プロパティーは、クライアント側で **HornetQ Core API** のメッセージグループを特定するために使用されます。ランダムな一意のメッセージグループ識別子を選択するには、**SessionFactory** で **auto-group** プロパティーを **true** に設定することもできます。

バグの報告

18.11.3. Java Messaging Service (JMS) クライアントのサーバー設定

JMSXGroupID プロパティーは、**Java Messaging Service (JMS)** クライアントのメッセージグループを特定するために使用されます。別のメッセージを持つメッセージグループを **1** つのコンシューマーに送信する場合は、異なるメッセージに同じ **JMSXGroupID** を設定できます。

```
Message message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);

message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);
```

2 つ目は、**HornetQConnectonFactory** で **auto-group** プロパティーを **true** に設定する方法です。その後、**HornetQConnectionFactory** はランダムな一意のメッセージグループ識別子を選択します。以下のようにサーバー設定ファイル（**standalone.xml** および **domain.xml**）に **auto-group** プロパティーを設定できます。

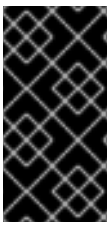
```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <auto-group>true</auto-group>
</connection-factory>
```

上記の方法の代替は、接続ファクトリーを介して特定のメッセージグループ識別子を設定することです。これにより、この接続ファクトリーを介して送信されたすべてのメッセージに対して **JMSXGroupID** プロパティーが指定された値に設定されます。接続ファクトリーに特定のメッセージグループ識別子を設定するには、サーバー設定ファイル（**standalone.xml** および **domain.xml**）の **group-id** プロパティーを以下のように編集します。

```
<connection-factory name="ConnectionFactory">
  <connectors>
    <connector-ref connector-name="netty-connector"/>
  </connectors>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
  <group-id>Group-0</group-id>
</connection-factory>
```

バグの報告

18.11.4. クラスター化されたグルーピング



重要

クラスター化されたグループ化は テクノロジープレビュー としてのみ提供されます。実稼働環境での使用はサポートされず、今後大幅に変更される可能性があります。

クラスター化されたグルーピングは、通常のメッセージのグループ化とは異なる方法に従います。クラスターの中では、特定のグループ **ID** を持つメッセージグループが、ノードのいずれかに到達できます。ノードでは、どのノードの、どのコンシューマーに、どのグループ **ID** がバインドされるかを判断することが重要です。各ノードは、メッセージグループがデフォルトで到達する場所に関係なく、それらのグループ **ID** を処理するコンシューマーを持つノードにメッセージグループを正しくルーティングします。指定したグループ **ID** を持つメッセージが、クラスター内の指定したノードに接続されている特定のコンシューマーに送信されます。この場合、これらのメッセージはコンシューマーが切断されても別のノードに送信されることはありません。

この状況は、グルーピングハンドラーによって処理されます。各ノードにグルーピングハンドラーがあります。また、このグルーピングハンドラー (と他のハンドラー) は、メッセージグループを正しいノードにルーティングする役割を担います。グルーピングハンドラーには、ローカル と リモート という **2** つのタイプがあります。

ローカルハンドラーは、メッセージグループが取るルートを決定します。リモートハンドラーはローカルハンドラーと通信し、適切に動作します。各クラスターは、特定のノードを選択してローカルグルーピングハンドラー指定するようにします。また、その他のすべてのノードには、リモートハンドラー指定する必要があります。

警告

メッセージのグループ化がクラスターで使用されると、設定されたリモートグルーピングハンドラーを持つサーバーが失敗すると破損します。リモートグルーピングハンドラーのバックアップの設定にも影響はありません。

以下のようにサーバー設定ファイル（**standalone.xml** および **domain.xml**）で「ローカル」および「リモート」グルーピングハンドラーを設定できます。

```
<grouping-handler name="my-grouping-handler">
  <type>LOCAL</type>
  <address>jms</address>
  <timeout>5000</timeout>
```

```
</grouping-handler>

<grouping-handler name="my-grouping-handler">
   <type>REMOTE</type>
   <address>jms</address>
   <timeout>5000</timeout>
</grouping-handler>
```

"timeout" 属性は、指定した時間内にルーティングを決定することを保証します。この時間内にデシジョンが行われないと、例外が発生します。

最初にメッセージグループを受け取るノードは、通常のクラスタールーティング条件（ラウンドロビンキューの可用性）に基づいてルーティングの決定を行います。ノードは、この決定をそれぞれのグルーピングハンドラーに提案します。グルーピングハンドラーが受け入れた場合は、提案されたキューにメッセージをルーティングします。

グルーピングハンドラーが提案を拒否した場合は、他のルートを提案し、それに応じてルーティングを行います。その他のノードも先例に従い、選択したキューにメッセージグループを転送します。メッセージがキューに到達すると、そのキューのカスタマーに固定されます。

バグの報告

18.11.5. クラスター化されたグルーピングのベストプラクティス

クラスター化されたグループ化のベストプラクティスには、以下のものがあります。

- コンシューマーを定期的に作成して閉じる場合は、コンシューマーが別のノードに均等に分散されていることを確認します。キューが固定されると、キューからカスタマーを削除するかどうかに関わらず、そのキューにメッセージが自動的に転送されます。
- メッセージグループがバインドされているキューを削除する場合は、メッセージを送信しているセッションによってキューが削除されていることを確認してください。これを実行することで、他のノードは削除後にこのキューにメッセージをルーティングしないようにします。
- フェイルオーバーメカニズムとして、ローカルグルーピングハンドラーを持つノードを常にレプリケートします。

バグの報告

18.12. 複製メッセージの検出

18.12.1. 複製メッセージの検出

複製メッセージの検出により、アプリケーション内で重複検出ロジックをコーディングせずに複製メッセージをフィルターできます。**HornetQ** で複製メッセージの検出を設定できます。

送信側（クライアント/サーバー）がメッセージを別のサーバーに送信する場合、メッセージの送信後に処理が成功したことを示す応答を送信する前にターゲットサーバー（受信側）または接続に失敗する可能性があります。このような状況では、送信側（クライアント）がメッセージが目的の受信者に正常に送信されたかどうかを判断することは非常に困難です。

メッセージ送信は、（メッセージの送信前または送信後）ターゲット受信側または接続が失敗したタイミングによって、成功することがあります。送信側（クライアント/サーバー）が最後のメッセージを再送信すると、アドレスに重複メッセージが送信される可能性があります。

HornetQ は、アドレスに送信されたメッセージに対して複製メッセージの検出機能を提供します。

バグの報告

18.12.2. メッセージ送信に重複メッセージ検出を使用する

送信メッセージに対して複製メッセージの検出を有効にするには、メッセージで特別なプロパティーを一意の値に設定する必要があります。この値は必要に応じて作成できますが、この値は一意でなければなりません。

ターゲットサーバーがこのメッセージを受信すると、特別なプロパティーが設定されているかどうかを確認します。プロパティーが設定されている場合、ターゲットサーバーはヘッダーのその値で受信したメッセージのメモリーキャッシュをチェックします。サーバーがヘッダーと同じ値を持つメッセージを見つけると、クライアントが送信したメッセージを無視します。

トランザクションでメッセージを送信する場合、そのトランザクションで送信するすべてのメッセージにプロパティーを設定する必要はありません。トランザクションで一度だけ設定する必要があります。サーバーがトランザクションのメッセージの重複メッセージを検出すると、トランザクション全体を無視します。

設定するプロパティーの名前は、**org.hornetq.api.core.HDR_DUPLICATE_DETECTION_ID** の値によって指定されます(**_HQ_DUPL_ID**)。このプロパティーの値は、コア **API** の **byte[]** または **SimpleString** 型です。**Java Messaging Service(JMS)**クライアントの場合、一意の値を持つ **String** タイプである必要があります。一意の **ID** を簡単に生成する方法は、**UUID** を生成することです。

下例はコア **API** のプロパティーを設定する方法を示しています。

```
...

ClientMessage message = session.createMessage(true);

SimpleString myUniqueID = "This is my unique id";  // Can use a UUID for this

message.setStringProperty(HDR_DUPLICATE_DETECTION_ID, myUniqueID);

...
```

下例は **JMS** クライアントのプロパティーを設定する方法を示しています。

```
...

Message jmsMessage = session.createMessage();

String myUniqueID = "This is my unique id";  // Could use a UUID for this

message.setStringProperty(HDR_DUPLICATE_DETECTION_ID.toString(), myUniqueID);

...
```

バグの報告

18.12.3. 複製 ID キャッシュの設定

サーバーは、各アドレスに送信された **org.hornetq.core.message.impl.HDR_DUPLICATE_DETECTION_ID** プロパティーの受信値のキャッシュを維持します。各アドレスは独自のアドレスキャッシュを維持します。

キャッシュのサイズは固定です。キャッシュの最大サイズは、サーバー設定ファイル（**standalone.xml** および **domain.xml**）の **id-cache-size** パラメーターを使用して設定されます。このパラメーターのデフォルト値は **2000** 要素です。キャッシュの最大サイズが **n** 要素である場合、**(n + 1)** 番目の **ID** が保存されていると、キャッシュ内の **0** 番目の要素が上書きされます。

キャッシュは、ディスクに永続化するように設定することもできます。これは、サーバー設定ファイル（**standalone.xml** および **domain.xml**）で **persist-id-cache** パラメーターを使用して設定できます。この値を「**true**」に設定すると、各 **ID** は受信時に永続ストレージに永続化されます。このパラメーターのデフォルト値は **true** です。

注記

メッセージを再送信した場合にキャッシュに保存されている送信済みのメッセージが上書きされないようにするには、重複 **ID** キャッシュのサイズを大きなサイズに設定します。

バグの報告

18.12.4. ブリッジおよびクラスター接続での複製検出の使用

コアブリッジは、メッセージをターゲットに転送する前に、一意の重複 **ID** 値（メッセージにまだない場合）を自動的に追加するように設定できます。重複メッセージ検出にコアブリッジを設定するには、サーバー設定ファイル（**standalone.xml** および **domain.xml**）でプロパティー **use-duplicate-detection** を **"true"** に設定します。このパラメーターのデフォルト値は「**true**」です。

クラスター接続は内部的にコアブリッジを使用してクラスターのノード間でメッセージを移動します。重複メッセージ検出にクラスター接続を設定するには、サーバー設定ファイル（**standalone.xml** および **domain.xml**）でプロパティー **use-duplicate-detection** を **"true"** に設定します。このパラメーターのデフォルト値は「**true**」です。

バグの報告

18.13. JMS ブリッジ

18.13.1. ブリッジ

ブリッジの機能は、ソースキューからメッセージを消費し、通常は異なる **HornetQ** サーバーにあるターゲットアドレスに転送することです。ブリッジは信頼できる接続に対応し、接続が再び利用可能になると自動的に再接続します。**HornetQ** ブリッジはフィルター式で設定でき、特定のメッセージのみを転送できます。

重要

JMS ブリッジは、**HornetQ** が専用のバックアップとして設定された **EAP 6** サーバーにデプロイできません。そのため、専用のバックアップサーバーのトランザクションマネージャーは、**HornetQ** ライブサーバーで以前に開始したトランザクションをリカバリーできないからです。

バグの報告

18.13.2. JMS ブリッジの作成

概要

JMS ブリッジはソースの **JMS** キューまたはトピックからメッセージを消費し、通常は異なるサーバーにあるターゲット **JMS** キューまたはトピックへ送信します。**JMS 1.1** に準拠する **JMS** サーバーの間でメッセージをブリッジするために使用できます。送信元および宛先の **JMS** リソースは、**JNDI** を使用してルックアップされ、**JNDI** ルックアップのクライアントクラスはモジュールでバンドルされる必要があります。モジュール名は **JMS** ブリッジ設定で宣言されます。

手順**18.2 JMS** ブリッジの作成

この手順では、**JMS** ブリッジを設定して、メッセージを **JBoss EAP 5.x** サーバーから **JBoss EAP 6** サーバーへ移行する方法を示します。

1. ソース **JMS** メッセージングサーバーでのブリッジの設定

   サーバータイプに記載されている手順に従って、ソースサーバーに **JMS** ブリッジを設定します。**JBoss EAP 5.x** サーバーに **JMS** ブリッジを設定する方法の例は、**JBoss EAP 6** 『**移行ガイド**』 『**の「JMS ブリッジの作成**」 」を参照してください。

2. 宛先 **JBoss EAP 6** サーバー上のブリッジの設定

   JBoss EAP 6.1 以降では、**JMS** ブリッジを使用して **JMS 1.1** 準拠のサーバーからメッセージをブリッジできます。ソースおよびターゲットの **JMS** リソースは **JNDI** を使用してルックアップされるため、ソースメッセージングプロバイダーの **JNDI** ルックアップクラスまたはメッセージブローカーは **JBoss** モジュールにバンドルされる必要があります。以下の手順では、例として特別な **'MyCustomMQ'** メッセージブローカーを使用しています。

a. メッセージプロバイダーの **JBoss** モジュールを作成します。

i. 新しいモジュールの **EAP_HOME/modules/system/layers/base/** の下にディレクトリー構造を作成します。**main/** サブディレクトリーには、クライアント **JAR** および **module.xml** ファイルが含まれます。**MyCustomMQ** メッセージングプロバイダー用に作成されたディレクトリー構造の例 **EAP_HOME/modules/system/layers/base/org/mycustommq/main/**

ii. **main/** サブディレクトリーで、メッセージングプロバイダーのモジュール定義が含まれる **module.xml** ファイルを作成します。**MyCustomMQ** メッセージングプロバイダー用に作成された **module.xml** の例を以下に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="org.mycustommq">
   <properties>
      <property name="jboss.api" value="private"/>
   </properties>

   <resources>
      <!-- Insert resources required to connect to the source or target   -->
      <resource-root path="mycustommq-1.2.3.jar" />
      <resource-root path="mylogapi-0.0.1.jar" />
   </resources>

   <dependencies>
     <!-- Add the dependencies required by JMS Bridge code                 -->
     <module name="javax.api" />
     <module name="javax.jms.api" />
     <module name="javax.transaction.api"/>
     <!-- Add a dependency on the org.hornetq module since we send         -->
     <!-- messages tothe HornetQ server embedded in the local EAP instance --
>
     <module name="org.hornetq" />
   </dependencies>
</module>
```

iii. ソースリソースの **JNDI** ルックアップに必要なメッセージングプロバイダー **JAR** をモジュールの **main/** サブディレクトリーにコピーします。**MyCustomMQ** モジュールのディレクトリー構造は以下のようになります。

```
modules/
`-- system
    `-- layers
        `-- base
            `-- org
                `-- mycustommq
```

```
`-- main
  |-- mycustommq-1.2.3.jar
  |-- mylogapi-0.0.1.jar
  |-- module.xml
```

b.

JBoss EAP 6 サーバーの **messaging** サブシステムで **JMS** ブリッジを設定します。

i.

開始する前に、サーバーを停止し、現在のサーバー設定ファイルをバックアップします。スタンドアロンサーバーを実行している場合は、**EAP_HOME/standalone/configuration/standalone-full-ha.xml** ファイルになります。管理対象ドメインを実行している場合は、**EAP_HOME/domain/configuration/domain.xml** ファイルと **EAP_HOME/domain/configuration/host.xml** ファイルの両方をバックアップします。

ii.

jms-bridge 要素をサーバー設定ファイルの **messaging** サブシステムに追加します。**source** 要素および **target** 要素は、**JNDI** ルックアップに使用される **JMS** リソースの名前を提供します。ユーザー と パスワード の認証情報が指定されている場合は、**JMS** 接続の作成時に引数として渡されます。

MyCustomMQ メッセージングプロバイダーに設定された **jms-bridge** 要素の例を以下に示します。

```
<subsystem xmlns="urn:jboss:domain:messaging:1.3">
  ...
  <jms-bridge name="myBridge" module="org.mycustommq">
    <source>
      <connection-factory name="ConnectionFactory"/>
      <destination name="sourceQ"/>
      <user>user1</user>
      <password>pwd1</password>
      <context>
        <property key="java.naming.factory.initial"
value="org.mycustommq.jndi.MyCustomMQInitialContextFactory"/>
        <property key="java.naming.provider.url"    value="tcp://127.0.0.1:9292"/>
      </context>
    </source>
    <target>
      <connection-factory name="java:/ConnectionFactory"/>
      <destination name="/jms/targetQ"/>
    </target>
    <quality-of-service>DUPLICATES_OK</quality-of-service>
    <failure-retry-interval>500</failure-retry-interval>
    <max-retries>1</max-retries>
    <max-batch-size>500</max-batch-size>
    <max-batch-time>500</max-batch-time>
```

```
        <add-messageID-in-header>true</add-messageID-in-header>
    </jms-bridge>
</subsystem>
```

上記の例では、**JNDI** プロパティーは ソース の コンテキスト 要素で定義されます。上記の例のように、**context** 要素を省略すると、**JMS** リソースはローカルインスタンスでルックアップされます。

バグの報告

18.14. 永続性

18.14.1. HornetQ の永続性

HornetQ は独自の永続性を処理します。これには、メッセージング固有のユースケース向けに最適化された高パフォーマンスジャーナルが含まれています。

HornetQ ジャーナルは設定可能なファイルサイズでのみ追加されるため、単一の書き込み操作を有効にすることでパフォーマンスが向上します。これは、ディスク上の **1** 組のファイルで構成されます。このファイルは、最初に固定サイズで事前作成され、パディングが書き込まれます。サーバーの操作（メッセージの追加、削除、更新メッセージなど）を実行すると、ジャーナルファイルが満杯になるまで操作の記録がジャーナルに追加されます。ジャーナルファイルが満杯になるまで、次のジャーナルファイルが使用されます。

すべてのデータが削除された場合、高度なガベージコレクションアルゴリズムにより、ジャーナルファイルを回収して再利用できるかどうかが決定されます。圧縮アルゴリズムにより、ジャーナルファイルから不要な領域が削除され、データが圧縮されます。

またジャーナルは、ローカルトランザクションと **XA** トランザクションの両方に完全に対応しています。

ジャーナルの大半は **Java** で記述されていますが、ファイルシステムとのやりとりは抽象化されており、さまざまなプラグ可能な実装が可能となります。**HornetQ** に同梱される **2** つの実装は次のとおりです。

- **Java New I/O (NIO)**

  *ファイルシステムとのインターフェースに標準の **Java NIO** を使用します。これにより、非常に優れたパフォーマンスを実現し、**Java 6** 以降のランタイムを備えたプラットフォーム上で稼働します。*

- **Linux 非同期 IO (AIO)**

  *ネイティブコードラッパーを使用して **Linux** 非同期 **IO** ライブラリー**(AIO)**と通信します。**AIO** では、データを永続化すると **HornetQ** はメッセージを受信します。これにより、明示的な同期の必要性がなくなります。通常、**AIO** は **Java NIO** よりも優れたパフォーマンスを提供しますが、**Linux** カーネル **2.6** 以降および **libaio** パッケージが必要です。*

  *また、**AIO** には **ext2**、**ext3**、**ext4**、**jfs**、または **xfs** タイプのファイルシステムも必要になります。*

*標準の **HornetQ** コアサーバーは次のジャーナルインスタンスを使用します。*

- **バインディングジャーナル**

  *サーバーにデプロイされたキューのセットとその属性を含む、バインディング関連のデータを格納します。また、**ID** シーケンスカウンターなどのデータも格納します。通常、バインディングジャーナルはメッセージジャーナルと比較するとスループットが低くなるため、バインディングジャーナルは常に **NIO** ジャーナルになります。*

  *このジャーナルのファイルには、**hornetq-bindings** というプレフィックスが付けられます。各ファイルには **bindings** 拡張子があります。ファイルサイズは **1048576** バイトで、**bindings** フォルダーにあります。*

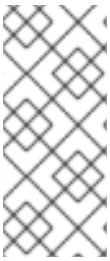
- **JMS ジャーナル**

  ***JMS** キュー、トピックまたは接続ファクトリー、これらのリソースの **JNDI** バインディングなど、**JMS** 関連のデータをすべて格納します。管理 **API** で作成された **JMS** リソースは、このジャーナルに永続化されます。設定ファイルで設定したリソースはありません。このジャーナルは、**JMS** が使用されている場合にのみ作成されます。*

- **メッセージジャーナル**

  *メッセージ自体や **duplicate-id** キャッシュを含む、メッセージ関連のデータをすべて格納します。デフォルトでは、**HornetQ** はこのジャーナルに **AIO** を使用します。**AIO** が利用できない場合は、自動的に **NIO** にフォールバックします。*

*大きなメッセージは、メッセージジャーナル外で永続化されます。メモリーが少ない状況では、メッセージをディスクにページングするように **HornetQ** を設定します。永続性が必要ない場合は、データを永続化しないよう **HornetQ** を設定できます。*



注記

__HornetQ__ ネイティブおよびジャーナルタイプが __AsyncIO__ に設定された状態で __JBoss EAP__ サーバーを実行すると、__tmpfs__ ファイルシステムで実行しているとエラーが発生します。

バグの報告

18.14.2. ジャーナルデータのインポートまたはエクスポート

***HornetQ** が使用するジャーナルの __1__ つに対して既存のレコードを確認し、その目的に __export/import__ ツールを使用できます。__export/import__ は __EAP_HOME/bin/client/jboss-client.jar__ クラスで、 __java -cp jboss-client.jar org.hornetq.core.journal.impl.ExportJournal <JournalDirectory> <JournalPrefix> <FileExtension> <FileSize> <FileOutput> <FileOutput>__*

ジャーナルでファイルをバイナリーデータとしてインポートするには、 __java -cp jboss-client.jar org.hornetq.core.journal.impl.ImportJournal <JournalDirectory> <JournalPrefix> <FileExtension> <FileSize> <FileInput> <FileInput >__

- ***__JournalDirectory:__ 選択したフォルダーに設定したフォルダーを使用します。*
  例：__./hornetq/data/journal__

- ***__JournalPrefix:__ で説明するように、選択したジャーナルに接頭辞を使用します。*

- ***__FileExtension:__ 説明しているように、選択したジャーナルの拡張を使用します。*

- ***filesize:*** *説明しているように、選択したジャーナルのサイズを使用します。*

- ***FileOutput:*** *エクスポートされたデータが含まれるテキストファイル*

バグの報告

18.15. HORNETQ クラスタリング

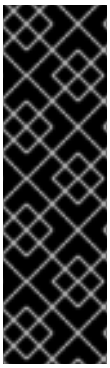
HornetQ *クラスターは、メッセージ処理負荷を共有するために* ***HornetQ*** *サーバーのグループを作成するために使用されます。クラスター内のアクティブな各ノードは独立した* ***HornetQ*** *サーバーとして機能し、独自のメッセージと接続を管理します。*

クラスターを形成するために、各ノード（独立した ***HornetQ*** *サーバー）はサーバー設定ファイル（**standalone.xml** および **domain.xml**）の設定パラメーターを使用して別のノードでクラスター接続を宣言します。*

クラスタリングでは、あるクラスターから別のクラスターへメッセージをブリッジ/ルーティングするのにコアブリッジが使用されます。コアブリッジはソースキューからメッセージを消費し、同じクラスターにある可能性があるターゲット ***HornetQ*** *サーバー（ノード）へこれらのメッセージを転送します。*

ノードが別のノードとクラスター接続を形成すると、内部的にコアブリッジが作成されます。各ノードは明示的なコアブリッジを作成するため、宣言する必要はありません。これらのクラスター接続により、メッセージ処理の負荷を分散するために、さまざまなクラスターのノード間でメッセージを送信できます。

*クラスターノードはサーバー設定ファイル（**standalone.xml** および **domain.xml**）で設定できます。*



重要

*サーバー設定ファイル（**standalone.xml** および **domain.xml**）でノードを設定し、この設定を他のノードにコピーして対称クラスターを生成することができます。ただし、サーバー設定ファイルをコピーする場合には注意が必要です。**HornetQ** データ（バインディング、ジャーナル、大きなメッセージディレクトリーなど）をあるノードから別のノードにコピーすることはできません。ノードを初めて起動すると、クラスターの適切な形式に必要なジャーナルディレクトリーに一意の識別子が永続化されます。*

バグの報告

18.15.1. サーバーディスカバリー

サーバーは一「サーバーディスカバリー」と呼ばれるメカニズムを使用して以下を行います。

- サーバーの接続詳細をメッセージングクライアントへ転送します。メッセージングクライアントは、指定の時点で稼働しているサーバーの特定の詳細を持たずにクラスターのサーバーへ接続しようとします。
- 他のサーバーへ接続します。クラスターのサーバーは、クラスターにある他のすべてのサーバーに関する特定の詳細を持たずに他のサーバーとクラスター接続を確立しようとします。

サーバーの情報は、通常の **HornetQ** 接続を介してメッセージングクライアントへ送信され、さらにクラスター接続を介して他のサーバーへ送信されます。

初回の接続を確立する必要があります。**UDP (**ユーザーデータグラムプロトコル**)** や **JGroups** などの動的サーバーディスカバリー技術を使用するか、コネクターのリストを提供して接続を確立します。

バグの報告

18.15.2. ブロードキャストグループ

コネクターはクライアントで使用され、サーバーへの接続方法および方法を定義します。サーバーはブロードキャストグループを使用してネットワーク経由でコネクターをブロードキャストします。ブロードキャストグループはコネクターのペアのセットを取得し、ネットワーク上でブロードキャストします。各コネクターペアには、ライブサーバーとバックアップサーバーの接続設定が含まれます。

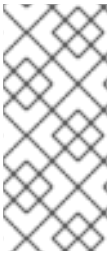
ブロードキャストグループはサーバー設定ファイル（**standalone.xml** および **domain.xml**）の **broadcast-groups** 要素で定義できます。**1** つの **HornetQ** サーバーは多くのブロードキャストグループを持つことができます。**UDP(User Datagram Protocol)**または **JGroup** ブロードキャストグループを定義できます。

バグの報告

18.15.2.1. UDP (ユーザーデータグラムプロトコル) ブロードキャストグループ

以下は **UDP** ブロードキャストグループの定義例になります。

```
<broadcast-groups>
  <broadcast-group name="my-broadcast-group">
    <local-bind-address>172.16.9.3</local-bind-address>
    <local-bind-port>5432</local-bind-port>
    <group-address>231.7.7.7</group-address>
    <group-port>9876</group-port>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```



注記

上例の **local-bind-address**、**local-bind-port**、**group-address**、および **"group-port"** の属性は非推奨になりました。これらの属性の代わりに、**socket-binding** 属性を使用できます。

以下は、廃止された属性を **socket-binding** 属性に置き換えた **UDP** ブロードキャストグループの定義例になります。

```
<broadcast-groups>
  <broadcast-group name="my-broadcast-group">
    <socket-binding>messaging-group</socket-binding>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

下表は、上記の例や通常の設定で **UDP** ブロードキャストグループを定義するために使用される重要なパラメーターを説明しています。

表18.11 **UDP** ブロードキャストグループパラメーター

属性	説明
name 属性	サーバーの各ブロードキャストグループの名前を表します。各ブロードキャストグループには一意の名前を指定する必要があります。
local-bind-address	**[廃止済み]** データグラムパケットのバインド先となるローカルバインドアドレスを指定する **UDP** 固有の属性です。ブロードキャストに使用するインターフェースを定義するには、このプロパティーを設定する必要があります。このプロパティーが指定されていない場合、ソケットはワイルドカードアドレス（ランダムなカーネル生成アドレス）にバインドします。
local-bind-port	**[廃止済み]** データグラムソケットがバインドするローカルポートを指定するために使用される **UDP** 固有の属性です。デフォルト値の「**-1**」は、使用される匿名ポートを指定します。
group-address	**[廃止済み]** メッセージがブロードキャストされる **UDP** 固有のマルチキャストアドレスです。この **IP** アドレスの範囲は **224.0.0.0** から **239.255.255.255** までです。**IP** アドレス **224.0.0** は予約されており、使用できません。

属性	説明
group-port	[廃止済み] ブロードキャスト用の **UDP** ポート番号を表します。
socket-binding	ブロードキャストグループのソケットバインディングを表します。
broadcast-period	このパラメーターは **2** つのブロードキャストの間隔（ミリ秒単位）を指定します。これはオプションです。
connector-ref	ブロードキャストされるコネクターを参照します。

バグの報告

18.15.2.2. JGroups ブロードキャストグループ

JGroups を使用してブロードキャストするには、**jgroups-stack** と **jgroups-channel** という **2** つの属性を指定します。以下は **JGroups** ブロードキャストグループの定義例になります。

```
<broadcast-groups>
  <broadcast-group name="bg-group1">
```

```
    <jgroups-stack>udp</jgroups-stack>
    <jgroups-channel>udp</jgroups-channel>
    <broadcast-period>2000</broadcast-period>
    <connector-ref>netty</connector-ref>
  </broadcast-group>
</broadcast-groups>
```

JGroups ブロードキャストグループの定義には主に *2* つの属性が使用されます。

- *jgroups-stack* 属性： これは、*org.jboss.as.clustering.jgroups* サブシステムに定義されたスタックの名前を示します。
- *jgroups-channel* 属性： *JGroups* チャネルがブロードキャストするために接続するチャネルを表します。

バグの報告

18.15.3. 検出グループ

ブロードキャストグループは、ネットワーク経由でコネクターをブロードキャストするために使用されます。一方、ディスカバリーグループでは、コネクター情報がブロードキャストエンドポイント（*UDP* または *JGroups* ブロードキャストグループ）から受信する方法を定義します。検出グループは、異なるサーバーによってブロードキャストごとにコネクターペアのリストを維持します。

検出グループが、特定サーバーのブロードキャストエンドポイントでブロードキャストを受信すると、特定のサーバーのリストにあるコネクターペアのエントリーを更新します。特定のサーバーから長期間ブロードキャストを受信しない場合は、リストからサーバーのエントリーを削除します。

ディスカバリーグループは主にクラスター接続および *Java Messaging Service (JMS)* クライアントによって使用され、必要なトポロジーをダウンロードするために最初の接続情報を取得します。

注記

対応するブロードキャストグループ *(UDP* または *JGroups)* と一致する適切なブロードキャストエンドポイントを使用して各ディスカバリーグループを設定する必要があります。

バグの報告

18.15.3.1. サーバー上での UDP (ユーザーデータグラムプロトコル) ディスカバリーグループの設定

以下は **UDP** ディスカバリーグループの定義例になります。

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <local-bind-address>172.16.9.7</local-bind-address>
    <group-address>231.7.7.7</group-address>
    <group-port>9876</group-port>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

注記

上例の **local-bind-address**、**group-address**、および **group-port** 属性は非推奨となっています。これらの属性の代わりに、**socket-binding** 属性を使用できます。

以下は、廃止された属性を **socket-binding** 属性に置き換えた **UDP** ディスカバリーグループの定義例になります。

```
<discovery-groups>
  <discovery-group name="my-discovery-group">
    <socket-binding>messaging-group</socket-binding>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

下表は、上記の例や通常の設定で ディスカバリーグループを定義するために使用される重要なパラメーターを説明しています。

表18.12 UDP ディスカバリーグループパラメーター

属性	説明
name 属性	この属性は、ディスカバリーグループの名前を示します。検出名はそれぞれ、サーバーごとに一意の名前を指定する必要があります。
local-bind-address	[廃止済み] 任意の **UDP** 固有の属性です。同じマシンで複数のインターフェースを使用する場合に、特定のインターフェースでリッスンするディスカバリーグループを設定するために使用されます。
group-address	[廃止済み] **UDP** 固有の属性です。これは、グループのマルチキャスト **IP** アドレスをリッスンするようにディスカバリーグループを設定するために使用されます。この属性の値は、リッスンするブロードキャストグループの **group-address** 属性と一致する必要があります。
group-port	[廃止済み] **UDP** 固有の属性です。マルチキャストグループの **UDP** ポートを設定するために使用されます。この属性の値は、リッスンするマルチキャストグループの **group-port** 属性と一致する必要があります。

属性	説明
socket-binding	ディスカバリーグループのソケットバインディングを表します。
refresh-timeout	これはオプションの **UDP** 固有の属性です。サーバーの最後のブロードキャストを受信した後、検出グループが待機する期間（ミリ秒単位）を設定するために使用されます。この期間（ミリ秒単位）は、そのサーバーから最後のブロードキャストを受信した後にサーバーのコネクターペアエントリーをリストから削除します。**refresh-timeout** の値は、ブロードキャストプロセスがオンのときにサーバーがリストの迅速な削除を防ぐために、ブロードキャストグループの **broadcast-period** 属性の値よりもはるかに高くする必要があります。この属性のデフォルト値は **10,000** ミリ秒です。

バグの報告

18.15.3.2. サーバー上での JGroups ディスカバリーグループの設定

以下は **JGroups** ディスカバリーグループの定義例になります。

```
<discovery-groups>
  <discovery-group name="dg-group1">
    <jgroups-stack>udp</jgroups-stack>
    <jgroups-channel>udp</jgroups-channel>
    <refresh-timeout>10000</refresh-timeout>
  </discovery-group>
</discovery-groups>
```

JGroups ディスカバリーグループの定義には主に **2** つの属性が使用されます。

- **jgroups-stack** 属性： これは、**org.jboss.as.clustering.jgroups** サブシステムに定義されたスタックの名前を示します。

- **jgroups-channel** 属性： この属性は、ブロードキャストを受信するために **JGroups** チャネルが接続するチャネルを表します。

注記

JGroup 属性と **UDP** 固有の属性は排他的です。検出グループまたはブロードキャストグループの設定で **JGroup** または **UDP** の属性セットを使用できます。

バグの報告

18.15.3.3. Java Messaging Service (JMS) クライアントに対するディスカバリーグループの設定

検出グループは、**JMS** およびコアクライアントに対して設定できます。サーバー設定ファイル（**standalone.xml** および **domain.xml**）の **JMS** 接続ファクトリーに使用されるディスカバリーグループを指定できます。

```
<connection-factory name="ConnectionFactory">
 <discovery-group-ref discovery-group-name="my-discovery-group"/>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
</connection-factory>
```

discovery-group-ref 要素は、検出グループの名前を指定するために使用されます。クライアントアプリケーションが **JNDI(Java Naming and Directory Interface)**からこの接続ファクトリーをダウンロードし、**JMS** 接続を作成すると、ディスカバリーグループ設定で指定されたマルチキャストアドレ

スでリッスンすることで、ディスカバリーグループが維持するすべてのサーバーで、これらの接続の負荷が分散されます。

JNDI ではなく **JMS** を使用して接続ファクトリーを検索する場合、**JMS** 接続ファクトリーの作成時に直接ディスカバリーグループパラメーターを指定できます。

```
final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ConnectionFactory jmsConnectionFactory = HornetQJMSClient.createConnectionFactory(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1)), JMSFactoryType.CF);
Connection jmsConnection1 = jmsConnectionFactory.createConnection();
Connection jmsConnection2 = jmsConnectionFactory.createConnection();
```

setter メソッド **setDiscoveryRefreshTimeout()** を使用して、**refresh-timeout** 属性のデフォルト値を **DiscoveryGroupConfiguration** に設定できます。接続ファクトリーが、最初の接続を作成する前に特定の時間待機するようにするには、**DiscoveryGroupConfiguration** でセッターメソッド **setDiscoveryInitialWaitTimeout()** を使用できます。

これにより、接続ファクトリーがクラスター内のすべてのノードからブロードキャストを受信するのに十分な時間を確保できます。このパラメーターのデフォルト値は **10000** ミリ秒です。

バグの報告

18.15.3.4. コア API のディスカバリー設定

コア **API** を使用して **ClientSessionFactory** インスタンスを直接インスタンス化する場合は、セッションファクトリーの作成時に直接ディスカバリーグループパラメーターを指定できます。

```
final String groupAddress = "231.7.7.7";
final int groupPort = 9876;
ServerLocator factory = HornetQClient.createServerLocatorWithHA(new
DiscoveryGroupConfiguration(groupAddress, groupPort, new
UDPBroadcastGroupConfiguration(groupAddress, groupPort, null, -1))));
ClientSessionFactory factory = locator.createSessionFactory();
ClientSession session1 = factory.createSession();
ClientSession session2 = factory.createSession();
```

setter メソッド **setDiscoveryRefreshTimeout()** を使用して、**refresh-timeout** 属性のデフォルト値を **DiscoveryGroupConfiguration** に設定できます。セッションファクトリーがセッションを作成す

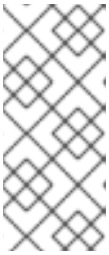
る前に、特定期間待機するために **DiscoveryGroupConfiguration** で **setDiscoveryInitialWaitTimeout()** を使用できます。

バグの報告

18.15.4. サーバー側の負荷分散

重要なクラスタートポロジーは次のとおりです。

- 対称クラスター：対称クラスターでは、各クラスターノードがクラスター内の他のすべてのノードに直接接続されます。対称クラスターを作成するには、**max-hops** 属性を **1** に設定して、クラスターの接続を定義します。



注記

対称クラスターでは、各ノードは他のすべてのノード上に存在するキューと、それらのコンシューマーを認識します。この知識により、ノード間でメッセージの負荷分散および再配布方法を決定できます。

バグの報告

18.15.4.1. クラスター接続の設定

クラスター接続は、**cluster-connection** 要素のサーバー設定ファイル（**standalone.xml** および **domain.xml**）で設定されます。**HornetQ** サーバーごとに **0** 個以上のクラスター接続を定義できます。

```
<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
    <check-period>1000</check-period>
    <connection-ttl>5000</connection-ttl>
    <min-large-message-size>50000</min-large-message-size>
    <call-timeout>5000</call-timeout>
    <retry-interval>500</retry-interval>
```

```
    <retry-interval-multiplier>1.0</retry-interval-multiplier>
    <max-retry-interval>5000</max-retry-interval>
    <reconnect-attempts>-1</reconnect-attempts>
    <use-duplicate-detection>true</use-duplicate-detection>
    <forward-when-no-consumers>false</forward-when-no-consumers>
    <max-hops>1</max-hops>
    <confirmation-window-size>32000</confirmation-window-size>
    <call-failover-timeout>30000</call-failover-timeout>
    <notification-interval>1000</notification-interval>
    <notification-attempts>2</notification-attempts>
     <discovery-group-ref discovery-group-name="my-discovery-group"/>
   </cluster-connection>
 </cluster-connections>
```

下表は設定可能な属性の説明になります。

表**18.13** クラスター接続の設定可能な属性

属性	説明	デフォルト
address	各クラスター接続は、この値で始まるアドレスに送信されるメッセージのみに適用されます。アドレスは任意の値にすることができ、アドレスの異なる値を持つ多くのクラスター接続があり、それらのアドレスのメッセージをサーバーの異なるクラスターに同時に分散できます。ワイルドカードの一致は使用されません。	
connector-ref	適切なクラスタートポロジーを持つようにするため、クラスターの他のノードへ送信されるコネクターを参照する必須の属性です。	
check-period	クラスター接続が別のサーバーからの **ping** を受信しなかったことを検証するために使用される期間 (ミリ秒単位) を表します。	**30,000** ミリ秒
connection-ttl	クラスターの特定のノードからメッセージを受信しなくなった場合に、クラスター接続が有効でなければならない期間を指定します。	**60,000** ミリ秒

属性	説明	デフォルト
min-large-message-size	メッセージのサイズ (バイト単位) がこの値を越える場合、ネットワーク上で他のクラスターメンバーへ送信されるときに複数のセグメントに分割されます。	**102400** バイト
call-timeout	例外が発生する前にクラスター接続上で送信されたパケットが応答を待つ期間 (ミリ秒単位) を指定します。	**30,000** ミリ秒
retry-interval	クラスターのノード間でクラスター接続が作成され、ターゲットノードが起動されていないか、再起動された場合、他のノードからのクラスター接続は再起動するまでターゲットへの接続を再試行します。パラメーター **retry-interval** は再試行の間隔（ミリ秒単位）を定義します。	**500** ミリ秒
retry-interval-multiplier	これは、再試行ごとに **retry-interval** をインクリメントするために使用されます。	**1**
max-retry-interval	再試行の最大遅延 (ミリ秒単位) を表します。	**2000** ミリ秒
reconnect-attempts	クラスターでシステムがノードへの接続を試行する回数を定義します。	**-1** (無限)
use-duplicate-detection	クラスター接続はブリッジを使用してノードをリンクし、ブリッジは転送される各メッセージに複製 **ID** プロパティーを追加するように設定できます。ブリッジのターゲットノードがクラッシュしてからリカバリーすると、メッセージがソースノードから送信される可能性があります。重複検出を有効にすると、重複メッセージはフィルターされ、ターゲットノードで受信時に無視されます。	**True**

属性	説明	デフォルト
forward-when-no-consumers	他のノードにコンシューマーが存在するかどうかに関わらず、クラスターの他のノードの間でメッセージがラウンドロビン方式で配布されるかどうかを決定するパラメーターです。	**False**
max-hops	このサーバーに接続された他の **HornetQ** サーバーへメッセージを負荷分散する方法を決定します。	**-1**
confirmation-window-size	接続しているサーバーから確認を送信するために使用されるウインドウのサイズ (バイト単位)。	**1048576**
call-failover-timeout	フェールオーバーの試行中に呼び出しが行われるときに使用されます。	**-1** (タイムアウトなし)
notification-interval	クラスターにアタッチするときにクラスター接続自体をブロードキャストする頻度 (ミリ秒単位) を決定します。	**1000** ミリ秒
notification-attempts	クラスターに接続するときにクラスター接続自体をブロードキャストする回数を定義します。	**2**
discovery-group-ref	現在のクラスター接続が接続するクラスターにある、他のサーバーのリストを取得するために使用されるディスカバリーグループを決定します。	

クラスターのノード間で接続を作成し、クラスター接続を形成するとき、**HornetQ** はサーバー設定ファイル（**standalone.xml** および **domain.xml**）で定義されるクラスターユーザーおよびクラスターパスワードを使用します。

```
<cluster-user>HORNETQ.CLUSTER.ADMIN.USER</cluster-user>
<cluster-password>NEW USER</cluster-password>
```

警告

リモートクライアントがデフォルト値を使用してサーバーに接続しないようにするため、これらのクレデンシャルのデフォルト値を変更することが重要になります。

バグの報告

18.16. 高可用性

18.16.1. 高可用性とは

***HornetQ** は、**1** つ以上のサーバーに障害が発生した場合でも機能を継続する機能をサポートします。これは、ライブサーバーに障害が発生した場合のクライアント接続がライブサーバーからバックアップサーバーに移行するフェイルオーバーサポートによって行われます。バックアップサーバーを最新の状態に維持するために、メッセージは共有ストアとレプリケーションの **2** つのストラテジーを通じてライブサーバーからバックアップサーバーに継続的にレプリケートされます。*

*高可用性トポロジーには次の **2** 種類があります。*

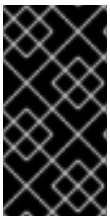
- *専用トポロジー：このトポロジーは **2** つの **EAP** サーバーで構成されます。最初のサーバーで **HornetQ** はライブサーバーとして設定されます。**2** 番目のサーバーで、**HornetQ** はバックアップサーバーとして設定されます。**HornetQ** がバックアップサーバーとして設定されている **EAP** サーバーは、**HornetQ** のコンテナーとしてのみ動作します。このサーバーは非アクティブであるため、**EJB**、**MDB**、サーブレットなどのデプロイメントをホストできません。*

- *配置トポロジー**:** このトポロジーには **2** つの **EAP** サーバーが含まれます。各 **EAP** サーバーには **2** つの **HornetQ** サーバー（ライブサーバーとバックアップサーバー）が含まれます。最初の **EAP** サーバーの **HornetQ** ライブサーバーと、**2** つ目の **EAP** サーバーの **HornetQ** バックアップサーバーはライブバックアップのペアを形成します。**2** つ目の **EAP** サーバーの*

HornetQ ライブサーバーと最初の EAP サーバーの HornetQ ライブサーバーは別のライブバックアップのペアを形成します。

配置トポロジーでは、ライブ HornetQ サーバー（ライブ/バックアップペアの一部）が失敗すると、バックアップ HornetQ サーバーが引き継ぎ、アクティブになります。フェイルバック時にバックアップ HornetQ サーバーがシャットダウンすると、バックアップサーバーに設定された宛先と接続ファクトリーは JNDI(Java Naming and Directory Interface)のバインドされません。

Java Naming and Directory Interface は、他のライブ HornetQ サーバー（他のライブ/バックアップペアの一部）と共有されます。そのため、JNDI から宛先と接続ファクトリーのバインドを解除すると、このライブ HornetQ サーバーの宛先と接続ファクトリーのバインドも解除されます。



重要

配置されたバックアップサーバーの設定に、宛先と接続ファクトリーの設定を含めることはできません。

注記

以下の情報が standalone-full-ha.xml を参照します。設定の変更は standalone-full-ha.xml に適用するか、そこから派生する設定ファイルに適用できます。

バグの報告

18.16.2. HornetQ Shared の共有ストア

共有ストアを使用する場合は、ライブサーバーとバックアップサーバーの両方が共有ファイルシステムを使用して同じデータディレクトリー全体を共有します。これには、ページングディレクトリー、ジャーナルディレクトリー、大型のメッセージ、バインディングジャーナルが含まれます。フェイルオーバーが発生し、バックアップサーバーが引き継ぐと、共有ファイルシステムから永続ストレージをロードします。その後、クライアントはこれに接続できます。

このような形式の高可用性は、ライブノードとバックアップノードの両方から共有ファイルシステムにアクセスできる必要があるため、データのレプリケーションとは異なります。通常、これは一部の時点で高パフォーマンスのストレージエリアネットワーク(SAN)になります。

共有ストアの高可用性の利点は、ライブノードとバックアップノード間でレプリケーションが発生しないことです。つまり、通常の操作中にレプリケーションのオーバーヘッドによりパフォーマンスが低下します。

共有ストアのレプリケーションの欠点は、共有ファイルシステムが必要であり、バックアップサーバーがアクティブ化する際には、共有ストアからジャーナルをロードする必要があることです。ストア内のデータサイズによっては、これには時間がかかる場合があります。

通常の操作中に最良のパフォーマンスが必要で、高速の **SAN** にアクセスでき、若干速度が遅いフェールオーバーを許可する (データの量に応じて) 場合は、共有ストアの高可用性が推奨されます。

注記

HornetQ のデータレプリケーションメカニズムは **JMS** データを複製し、バインディングを複製しません。

バグの報告

18.16.3. HornetQ ストレージの設定

HornetQ は、**Red Hat Enterprise Linux** バージョンの **NFSv4** を使用する場合、共有ストレージの **ASYNCIO** または **NIO** ジャーナルタイプを使用する場合に共有ストレージをサポートします。**Red Hat Enterprise Linux NFS** 実装は、ダイレクト **I/O**（**O_DIRECT** フラグセットでファイルを開く）とカーネルベースの非同期 **I/O** の両方をサポートします。共有ストレージ用に **NFS** を設定する場合は、可用性の高い **NFS** 設定を使用することが推奨されます。

重要

Red Hat Enterprise Linux NFSv4 を共有ストレージオプションとして使用する場合は、クライアントキャッシュを無効にする必要があります。

バグの報告

18.16.4. HornetQ のジャーナルタイプ

HornetQ では **2** つのジャーナルタイプを使用できます。

- **ASYNCIO**
- **NIO**

ASYNCIO ジャーナルタイプは **AIO** とも呼ばれる **Linux** 非同期 **IO** ライブラリー**(AIO)**に関するシンネイティブコードラッパーです。ネイティブ機能を使用すると、**NIO** よりも優れたパフォーマンスが得られます。このジャーナルタイプは **Red Hat Enterprise Linux** でのみサポートされ、**JBoss EAP 6** が実行されている **libaio** および **Native Components** パッケージをインストールする必要があります。ネイティブコンポーネントパッケージの 『**インストール方法は、『インストールガイド』**』 を参照してください。

重要

JBoss EAP 6 の起動後にサーバーログを確認し、ネイティブライブラリーが正常にロードされ、**ASYNCIO** ジャーナルタイプが使用されていることを確認します。ネイティブライブラリーの読み込みに失敗した場合、**HornetQ** は **NIO** ジャーナルタイプに戻され、サーバーログに記録されます。

NIO ジャーナルタイプは、ファイルシステムとのインターフェースに標準の **Java NIO** を使用します。非常に優れたパフォーマンスを提供し、サポートされるすべてのプラットフォームで実行されます。

HornetQ ジャーナルタイプを指定するには、メッセージング サブシステムで **& lt;journal-type>** パラメーターを設定します。

バグの報告

18.16.5. 共有ストアを持つ専用トポロジー向けの HornetQ の設定

専用トポロジーの共有ストアにライブサーバーとバックアップサーバーを設定するには、各サーバーの **standalone-X.xml** ファイルに以下が含まれるように設定します。

```
<shared-store>true</shared-store>
<paging-directory path="${shared.directory}/paging"/>
<bindings-directory path="${shared.directory}/bindings"/>
<journal-directory path="${shared.directory}/journal"/>
<large-messages-directory path="${shared.directory}/large-messages"/>
.
.
.
```

```
<cluster-connections>
  <cluster-connection name="my-cluster">
    ...
  </cluster-connection>
</cluster-connections>
```

表18.14 HornetQ サーバーの設定属性 (ライブおよびバックアップサーバー用)

属性	説明
shared-store	このサーバーが共有ストアを使用しているかどうか。デフォルトは **false** です。
paging-directory path	これは、ページングディレクトリーへのパスを示します。ライブサーバーとバックアップサーバーはこのディレクトリーを共有するため、このパスは同じです。
bindings-directory path	これは、バインディングジャーナルへのパスを示します。ライブサーバーとバックアップサーバーはこのジャーナルを共有するため、パスは同じです。
journal-directory path	これは、ジャーナルディレクトリーへのパスを示します。ライブサーバーとバックアップサーバーはこのディレクトリーを共有するため、このパスは同じです。

属性	説明
large-messages-directory path	これは、大きなメッセージディレクトリーへのパスを示します。ライブサーバーとバックアップサーバーはこのディレクトリーを共有するため、このパスは同じです。
failover-on-shutdown	ライブサーバーまたは現在アクティブなバックアップサーバーがシャットダウンすると、このサーバーがアクティブになるかどうか。

また、バックアップサーバーはバックアップとして明示的にフラグ付けする必要があります。

```
<backup>true</backup>
```

HornetQ バックアップサーバーのみの設定属性は **allow-failback** です。元のライブサーバーが復旧したときにバックアップサーバーが自動的にシャットダウンするかどうかを指定します。

バグの報告

18.16.6. HornetQ のメッセージレプリケーション

警告

永続メッセージのみがレプリケートされます。非永続的なメッセージはフェイルオーバー後も維持されません。

ライブサーバーとバックアップサーバーは同じデータストアを共有しないため、ライブサーバーとバックアップサーバー間のメッセージレプリケーションはネットワークトラフィックを介して実行されます。**2** つのサーバーが同じクラスター内にあり、クラスターユーザー名およびパスワードが同じであれば、ジャーナルはすべて **2** つのサーバー間でレプリケートされます。ライブサーバーが受信した永続データトラフィックはすべて、バックアップサーバーに複製されます。

バックアップサーバーがオンラインになると、ライブサーバーを検索し、同期を試行するためにライブサーバーに接続します。同期中、バックアップサーバーとしては使用できません。同期にかかる時間とネットワーク速度によっては、同期に時間がかかることがあります。バックアップサーバーがオンラインになり、ライブサーバーが利用できない場合、バックアップサーバーは、ライブサーバーがクラスターで利用可能になるまで待機します。

サーバーがデータを複製できるようにするには、**standalone-full-ha.xml** ファイルの間でリンクを定義する必要があります。バックアップサーバーは、同じグループ名を持つライブサーバーとのみ複製します。グループ名は、各サーバーの **standalone-full-ha.xml** ファイルの **backup-group-name** パラメーターで定義する必要があります。

ライブサーバーに障害が発生した場合、適切に設定され完全に同期されたバックアップサーバーが役割を引き継ぎます。バックアップサーバーは、ライブサーバーが失敗し、バックアップサーバーがクラスター内の半分を超えるサーバーに接続できる場合にのみアクティベートされます。クラスター内の他のサーバーが半分以上でも応答しない場合は、一般的なネットワーク障害を示し、バックアップサーバーはライブサーバーへの接続を再試行するのを待機します。

フェイルオーバー後に元の状態になるには、ライブサーバーを起動し、バックアップサーバーと完全に同期されるまで待つ必要があります。これを行うと、元のライブサーバーのバックアップサーバーをシャットダウンして、再度アクティブ化することができます。**allow-failback** 属性が **true** に設定されている場合に自動的に実行されます。

バグの報告

18.16.7. レプリケーションに対する HornetQ サーバーの設定

ライブサーバーとバックアップサーバーをレプリケートペアとして設定するには、各サーバーの **standalone-full-ha.xml** ファイルに以下の設定が含まれるように設定します。

```
<shared-store>false</shared-store>
<backup-group-name>NameOfLiveBackupPair</backup-group-name>
<check-for-live-server>true</check-for-live-server>
.
.
.
<cluster-connections>
```

```
    <cluster-connection name="my-cluster">
      ...
    </cluster-connection>
  </cluster-connections>
```

警告

管理者は、共有ストアとレプリケートされた設定の設定を混在させないように注意する必要があります。たとえば、レプリケーションに使用される ***backup-group-name*** *属性は、共有ストアを示す* ***shared-store*** *が* ***true*** *に設定されている場合は設定できません。*

表18.15 HornetQ レプリケーション設定属性

属性	説明
shared-store	このサーバーが共有ストアを使用しているかどうか。レプリケートされた設定では、この値を false に設定する必要があります。デフォルトは false です。
backup-group-name	お互いをレプリケートするライブ/バックアップペアを識別する一意名です。
check-for-live-server	レプリケートされたライブサーバーが現在のクラスターを確認し、同じノード ID を持つライブサーバーがあるかどうかを確認します。デフォルトは false です。
failover-on-shutdown	このバックアップサーバー（バックアップサーバーである場合）が通常のサーバーのシャットダウン時にライブサーバーになるかどうか。デフォルトは false です。

また、バックアップサーバーはバックアップとして明示的にフラグ付けする必要があります。

```
<backup>true</backup>
```

表18.16 HornetQ バックアップサーバー設定属性

属性	説明
allow-failback	元のライブサーバーが復旧したときにこのサーバーを自動的にシャットダウンするかどうか。デフォルトは true です。

属性	説明
max-saved-replicated-journal-size	フェイルバック発生後に保持するバックアップジャーナルの最大数。allow-failback が true の場合のみこの属性を指定する必要があります。デフォルト値は 2 です。つまり、ライブサーバーからジャーナルを複製し、再度バックアップになるように 2 回フェイルバック後にバックアップサーバーを再起動する必要があります。

バグの報告

18.16.8. 高可用性 (HA) フェールオーバー

*高可用性フェイルオーバーは、ライブバックアップ構造を介して、自動クライアントフェイルオーバーまたはアプリケーションレベルのフェイルオーバーのいずれかで利用できます。ライブサーバーにはそれぞれバックアップサーバーがあります。ライブサーバーごとに **1** つのバックアップのみがサポートされます。*

*バックアップサーバーは、ライブサーバーがクラッシュし、フェイルオーバーがある場合にのみ引き継ぎます。ライブサーバーが再起動され、**allow-failback** 属性が **true** に設定されている場合、ライブサーバーは再度ライブサーバーになります。元のライブサーバーが引き継ぎすると、バックアップサーバーはライブサーバーのバックアップに戻ります。*

重要

*クラスタリング機能を使用していない場合でも、クラスタリングを有効にする必要があります。これは、他のサーバーとロールをネゴシエートするために、**HA** クラスターの各ノードに他のノードすべてへの **cluster-connection** が必要であるためです。*

*高可用性クラスタートポロジーは、**IP** マルチキャストを使用して接続の詳細に関する情報を送信するため、ライブサーバーおよびバックアップサーバーによって実現されます。**IP** マルチキャストを使用できない場合は、初期接続の静的設定を使用することもできます。最初の接続後、クライアントはトポロジーについて通知されます。現在の接続が古くなると、クライアントは別のノードへの新しい接続を確立します。*

*ライブサーバーが失敗し、バックアップサーバーが引き継いした後、ライブサーバーを再起動してクライアントをフェイルバックする必要があります。これには、元のライブサーバーを再起動して、新しいライブサーバーを強制終了します。これは、プロセス自体を強制終了するか、またはサーバー自体でクラッシュするのを待つことで実行できます。また、通常のサーバーのシャットダウンでフェイルオーバーが発生し、**standalone.xml** 設定ファイルで **failover-on-shutdown** プロパティーを **true** に設定することもできます。*

```
<failover-on-shutdown>true</failover-on-shutdown>
```

デフォルトでは、**failover-on-shutdown** プロパティーは **false** に設定されます。

また、**standalone.xml** 設定ファイルで **allow-failback** プロパティーを **true** に設定すると、元のライブサーバーが復旧したときに新しいライブサーバーを強制的にシャットダウンし、元のライブサーバーが自動的に引き継げるように強制できます。

```
<allow-failback>true</allow-failback>
```

レプリケーション **HA** モードでは、古いライブサーバーが復旧したときに新しいライブサーバーを強制的にシャットダウンするには、**standalone.xml** 設定ファイルで **check-for-live-server** プロパティーを **true** に設定します。

```
<check-for-live-server>true</check-for-live-server>
```

バグの報告

18.16.9. HornetQ バックアップサーバー上のデプロイメント

専用の **HA** 環境では、**HornetQ** がバックアップとして設定された **JBoss EAP 6** サーバーを使用して、そのサーバー上の **HornetQ** バックアップを使用または接続するデプロイメントをホストしないでください。これには、**Enterprise Java Bean**（ステートレスセッション **Bean**、メッセージ駆動型 **Bean**）やサーブレットなどのデプロイメントが含まれます。

JBoss EAP 6 サーバーに **HornetQ** が配置されたバックアップ設定がある場合 **(**メッセージングサブシステムにライブサーバーとして設定された **HornetQ** サーバーがあり、別の **HornetQ** サーバーがバックアップとして設定されている場合**)**、デプロイメントが**HornetQ** のライブサーバーに接続するように設定されていれば、**JBoss EAP 6** サーバーはデプロイメントをホストできます。

バグの報告

18.16.10. HornetQ フェイルオーバーモード

HornetQ は、**2** 種類のクライアントフェイルオーバーを定義します。

- 自動クライアントフェイルオーバー
- アプリケーションレベルのクライアントフェイルオーバー

HornetQ は、一時的なネットワークの問題が発生した場合など、同じサーバーへの接続を透過的に再割り当てします。これはフェイルオーバーと似ていますが、同じサーバーに再接続されます。

フェイルオーバー中に、クライアントに非永続キューまたは一時キューにコンシューマーがある場合、バックアップノードには非永続キューについての情報がないため、これらのキューはバックアップノード上のフェイルオーバー中に自動的に再作成されます。

バグの報告

18.16.11. 自動クライアントフェイルオーバー

HornetQ クライアントは、ライブサーバーとバックアップサーバーに関する情報を受信するように設定できます。この情報は、クライアント接続に障害が発生した場合（ライブサーバー接続）、クライアントがフェイルオーバーを検出し、バックアップサーバーに再接続する場合に役立ちます。バックアップサーバーは、フェイルオーバー前に各接続に存在したセッションとコンシューマーを自動的に再作成します。そのため、ユーザーは再接続ロジックを手動でコーディングする必要がなくなります。

client-failure-check-period で指定された時間内にサーバーからパケットが受信されない場合、**HornetQ** クライアントは接続障害を検出します。クライアントが時間内にデータを受信しない場合、クライアントは接続が失敗し、フェイルオーバーを試みます。ソケットがオペレーティングシステムによって閉じられている場合、サーバープロセスはマシン自体がクラッシュしずに強制終了され、クライアントは即座にフェイルオーバーを開始します。

HornetQ クライアントを設定するには、ライブバックアップサーバーグループのリストを検出できます。クライアントは明示的に設定できるか、クライアントのサーバー検出を使用してリストを自動的

に検出できます。または、クライアントは特定のサーバーに明示的に接続し、現在のサーバーとバックアップをダウンロードできます。

自動クライアントフェイルオーバーを有効にするには、ゼロでない再接続試行を許可するようにクライアントを設定する必要があります。

デフォルトでは、フェイルオーバーは、ライブサーバーへの接続が少なくとも **1** つ確立された後にのみ発生します。クライアントは **reconnect-attempts** プロパティーで指定したライブサーバーへの接続を試み、指定した試行回数後に失敗します。

バグの報告

18.16.12. アプリケーションレベルのフェイルオーバー

状況に応じて、カスタム障害ハンドラーで再接続ロジックを指定して、接続障害を手動で処理できることがあります。フェイルオーバーはユーザーアプリケーションレベルで処理されるため、これをアプリケーションレベルのフェイルオーバーとして定義できます。

アプリケーションレベルのフェイルオーバーを実装するには、**JMS** を使用している場合は、**JMS** 接続で **ExceptionListener** クラスを設定する必要があります。接続の障害が検出されると、**ExceptionListener** クラスは **HornetQ** によって呼び出されます。**ExceptionListener** で、古い **JMS** 接続を閉じ、**JNDI** から新しい接続ファクトリーインスタンスを検索し、新しい接続を作成します。

コア **API** を使用している場合、手順は非常に似ています。コア **ClientSession** インスタンスで **FailureListener** を設定します。

バグの報告

18.17. パフォーマンスチューニング

18.17.1. 永続性の調整

- メッセージジャーナルを独自の物理ボリュームに配置します。ディスクが他のプロセスと共有されている場合（トランザクションコーディネーター、データベース、または他のジャーナルも読み取り/書き込み）、ディスクヘッドが異なるファイル間でスキップされる可能性があるため、パフォーマンスが大幅に低下する可能性があります。追記のみのジャーナルの利点の **1** つは、ディスクヘッド移動が最小限に抑えられることです。ディスクが共有されている場合

は、この利点は失われます。ページングまたは大きなメッセージを使用している場合は、それらが別のボリュームに配置されていることを確認します。

- ジャーナルファイルの最小数。**journal-min-files** パラメーターを、平均的に持続可能なレートに対応するファイル数に設定します。ジャーナルデータディレクトリーに新しいファイルがよく作成されますが、大量のデータが永続化される場合は、ファイルの最小数を増やす必要があります。これにより、ジャーナルが新規データファイルを作成するのではなく、より多くのファイルを再利用できます。

- ジャーナルファイルのサイズ。ジャーナルファイルのサイズは、ディスク上のシリンダーの容量に合わせて調整する必要があります。ほとんどのシステムでは、デフォルト値の **10MiB** で十分である必要があります。

- **AIO** ジャーナルを使用します。**Linux** オペレーティングシステムの場合、ジャーナルタイプは **AIO** のままにします。**AIO** は、**Java NIO** よりも優れています。

- **journal-buffer-timeout** を調整します。タイムアウトを増やしてスループットを増やすことができますが、待ち時間が代入されます。

- **AIO** を実行している場合は、**journal-max-io** パラメーター値を増やしてパフォーマンスを向上させることができる可能性があります。**NIO** を実行している場合は、このパラメーターを変更しないでください。

バグの報告

18.17.2. JMS の調整

JMS API を使用している場合は、いくつかの調整を行うことができます。

- メッセージ **ID** を無効にします。**MessageProducer** クラスで **setDisableMessageID**（） メソッドを使用して、メッセージ **ID** が必要ない場合に無効にします。これにより、メッセージのサイズが減少し、一意の **ID** の作成のオーバーヘッドも回避されます。

- メッセージのタイムスタンプを無効にします。**MessageProducer** クラスで

setDisableMessageTimeStamp（） メソッドを使用して、メッセージのタイムスタンプが必要ない場合にそれらを無効にします。

- **ObjectMessage** を使用しません。**ObjectMessage** は便利ですが、コストがかかります。**ObjectMessage** の本文は、**Java** のシリアライゼーションを使用してこれをバイトにシリアライズします。小さいオブジェクトでも **Java** のシリアライズ形式は非常に冗長であるため、ネットワーク上の多くのスペースを占有します。また、カスタムマーシャリング技術と比較して、**Java** のシリアライズも遅くなります。他のメッセージタイプの **1** つを使用できない場合にのみ **ObjectMessage** のみを使用します。これは、ランタイムまでペイロードのタイプが分からない場合です。

- **AUTO_ACKNOWLEDGE** を使用しません。**AUTO_ACKNOWLEDGE** モードでは、クライアント上で受信される各メッセージについてサーバーから確認応答を送信する必要があります。つまり、ネットワーク上のトラフィックが多いことを意味します。可能であれば **DUPS_OK_ACKNOWLEDGE** を使用するか、**CLIENT_ACKNOWLEDGE** またはトランザクションセッションを使用し、**1** つの確認応答/コミットで多くの確認応答をバッチアップします。

- 永続メッセージを使用しません。デフォルトでは、**JMS** メッセージは永続化されます。永続メッセージが必要ない場合は、それらを 非永続に設定します。永続メッセージにより、ストレージへの永続化のより多くのオーバーヘッドが発生します。

- **1** つのトランザクションでの多数の送信または確認応答。**HornetQ** では、送信または確認応答ごとにではなく、コミット時にネットワークラウンドトリップのみが必要になります。

バグの報告

18.17.3. その他のチューニング

HornetQ には、チューニングを実行できるさまざまな場所があります。

- 非同期送信応答を使用します。トランザクション以外の永続メッセージを送信する必要があり、**send** （） が返すときにサーバーに到達したことを保証する必要がある場合は、永続メッセージをブロックに送信 するように設定しないでください。代わりに非同期送信の確認応答を使用して、別のストリームで送信の応答を取得します。

- **pre-acknowledge** モードを使用します。**pre-acknowledge** モードでは、メッセージはクライアントに送信される前に確認応答されます。これにより、ネットワーク上の確認応答トラフィックの量が減少します。

- セキュリティーを無効にします。**standalone.xml** または **domain.xml** で **security-enabled** パラメーターを **false** に設定してセキュリティーを無効にすると、パフォーマンスが若干向上します。

- 永続性を無効にします。**standalone.xml** または **domain.xml** で **persistence-enabled** を **false** に設定すると、メッセージの永続性を完全にオフにすることができます。

- トランザクションを遅れて同期します。**standalone.xml** または **domain.xml** で **journal-sync-transactional** を **false** に設定すると、トランザクションの永続的なパフォーマンスが向上しますが、障害時にトランザクションが失われる可能性があります。

- トランザクション以外の遅延を同期します。**standalone.xml** または **domain.xml** で **journal-sync-non-transactional** を **false** に設定すると、障害時に永続メッセージが失われる可能性が高まり、非トランザクションの永続的なパフォーマンスが向上します。

- ブロック以外のメッセージを送信します。**standalone.xml** または **domain.xml**（**JMS** および **JNDI** を使用している場合は）または **ServerLocator** に直接、**block-on-durable-send** および **block-on-non-durable-send** を **false** に設定します。つまり、送信されるすべてのメッセージに対してネットワークラウンドトリップ全体を待機する必要はありません。

- 高速なコンシューマーがある場合は、**consumer-window-size** を増やすことができます。これにより、コンシューマーフローの制御が無効になります。

- ソケット **NIO** とソケット古い **IO** の比較デフォルトでは、**HornetQ** はサーバーとクライアント側で古い（ブロッキング）を使用します。**NIO** は非常にスケーラビリティーが高くなりますが、古いブロッキング **IO** と比較してレイテンシーのヒットが発生することがあります。サーバーで数千の接続を多数処理するには、サーバーで **NIO** を使用する必要があります。ただし、サーバーに数千の接続がない場合、サーバーアクセプターは古い **IO** を使用して維持でき、パフォーマンスが小さい可能性があります。

- **JMS** ではなくコア **API** を使用します。**JMS API** を使用すると、サーバーが処理する前にすべての **JMS** 操作をコア操作に変換する必要があるため、コア **API** を使用する場合よりもパフォーマンスが低下します。コア **API** を使用する場合は、可能な限り **SimpleString** を取得するメソッドの使用を試みます。**SimpleString** は、**java.lang.String** とは異なり、ネットワークに書き込まれる前にコピーする必要がないため、呼び出し間で **SimpleString** インスタンスを再利用する場合に不要なコピーを回避できます。

バグの報告

18.17.4. トランスポート設定のチューニング

- **TCP** バッファーサイズ。高速ネットワークおよび高速マシンがある場合、**TCP** の送受信バッファーサイズを増やしてパフォーマンスが向上する可能性があります。

注記

新しいバージョンの **Linux** のようなオペレーティングシステムには、**TCP** の自動チューニングや **TCP** バッファーサイズの手動設定が含まれているため、自動チューニングが機能しなくなり、実際にはパフォーマンスが低下する可能性があります。

- サーバーのファイルハンドルの制限を増やします。サーバーで多くの同時接続が必要な場合や、クライアントが接続を迅速に開閉する場合は、サーバーを実行しているユーザーに十分なファイルハンドルを作成するパーミッションがあることを確認する必要があります。

これはオペレーティングシステムによって異なります。**Linux** システムでは、**/etc/security/limits.conf** ファイルで許可可能なオープンファイルハンドルの数を増やすことができます。たとえば、以下の行を追加します。

```
serveruser     soft    nofile  20000
serveruser     hard    nofile  20000
```

これにより、ユーザー **serveruser** が最大 **20000** のファイルハンドルを開くことができます。

- **batch-delay** を使用し、非常に小さいメッセージに最適なスループットを得るために、**direct-deliver** を **false** に設定します。**HornetQ** には、**standalone.xml** または **domain.xml**、および **JMS** 接続ファクトリー **(ThroughputConnectionFactory)** に事前設定されたコネクター/アクセプターペア **(netty-throughput)** が含まれており、これは **standalone.xml** または **domain.xml** で、特に小さいメッセージに対して非常に最適なスループットを提供するために使用できます。

バグの報告

18.17.5. 仮想マシンのチューニング

最適なパフォーマンスを得るには、最新の **Java JVM** を使用することを強く推奨します。内部テストは **Sun JVM** を使用して実行されるため、これらのチューニングの一部は、**IBM** や **JRockit** などの他のプロバイダーからの **JDK** には適用されないことがあります。

- ガベッジコレクション。サーバーの操作をスムーズに行うには、並列ガベージコレクションアルゴリズムを使用することが推奨されます。たとえば、**Sun JDK** で **JVM** 引数 **-XX:+UseParallelGC** を使用します。

- メモリー設定。サーバーに可能な限り多くのメモリーを指定します。**HornetQ** はページングを使用して低メモリーで実行できますが、**RAM** のすべてのキューで実行できるとパフォーマンスが向上します。必要なメモリー量は、キューのサイズおよび数とメッセージのサイズおよび数によって異なります。**JVM** 引数 **-Xms** および **-Xmx** を使用して、サーバーが利用できる **RAM** を設定します。同じ高い値に設定することを推奨します。

- アグレッシブオプション。**JVM** が異なる **JVM** チューニングパラメーターのセットを提供します。**-XX:+AggressiveOpts** および **-XX:+Use FastAccessorMethods** を使用することが推奨されます。オペレーティングシステムプラットフォームおよびアプリケーションの使用パターンによっては、他のチューニングパラメーターと一部のミッテージを取得する場合があります。

バグの報告

18.17.6. アンチパターンの回避

- 接続/セッション/コンシューマー/プロデューサーを再利用します。おそらく最も一般的なメッセージングアンチパターンとは、送信または消費するすべてのメッセージに対して新しい **connection/session/producer** を作成するユーザーです。これはリソースの使用率が低くなります。これらのオブジェクトは作成に時間がかかり、複数のネットワークラウンドトリップを伴う場合があります。常に再利用してください。

注記

Spring JMS テンプレートなどの一般的なライブラリーは、これらのアンチパターンを使用します。**Spring JMS** テンプレートを使用している場合は、パフォーマンスが低下する可能性があります。**Spring JMS** テンプレートは、たとえば **JCA** を使用して **JMS** セッションをキャッシュするアプリケーションサーバーでのみ安全に使用でき、その後メッセージを送信するためにのみ使用できます。アプリケーションサーバーであっても、同期的に消費するメッセージに安全に使用することはできません。

-

*サイズの大きいメッセージを使用しません。**XML** のような詳細形式は、ネットワーク上の多くのスペースを占有し、結果としてパフォーマンスが低下します。可能な場合は、メッセージ本文では **XML** を使用しません。*

- *各リクエストに一時キューを作成しません。この一般的なアンチパターンには、一時キューの要求/応答パターンが含まれます。一時キュー要求/応答パターンでは、メッセージはターゲットに送信され、返信先ヘッダーはローカル一時キューのアドレスで設定されます。受信者がメッセージを受信すると、メッセージを処理すると、返信先で指定されたアドレスに応答を返信します。このパターンでよくある間違いは、送信された各メッセージに新しい一時キューを作成することです。これにより、パフォーマンスが大幅に低下します。代わりに、一時キューは多くのリクエストに再使用される必要があります。*

- *メッセージ駆動 **Bean** は使用しないでください。**MDB** の使用を開始すると、多くのアプリケーションサーバーコードが実行されるため、受信した各メッセージのコードパスを簡単なメッセージコンシューマーと比較すると、すぐに増加します。*

バグの報告

第19章 TRANSACTION サブシステム

19.1. トランザクションサブシステムの設定

19.1.1. トランザクション設定の概要

はじめに

次の手順は、**JBoss EAP 6** のトランザクションサブシステムを設定する方法を示しています。

- 「**JTA Transaction API** を使用するようデータソースを設定」
- 「**XA Datasource** の設定」
- 「トランザクションマネージャーの設定」
- 「トランザクションサブシステムのログ設定」

バグの報告

19.1.2. トランザクションマネージャーの設定

トランザクションマネージャー**(TM)**は、**Web** ベースの管理コンソールまたはコマンドライン管理 **CLI** を使用して設定できます。各コマンドまたはオプションについて、**JBoss EAP 6** を管理対象ドメインとして実行していると仮定します。スタンドアロンサーバーを使用する場合や、**default** 以外のプロファイルを変更する場合は、以下の方法で手順およびコマンドを変更しなければならない場合があります。

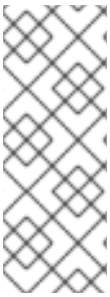
例のコマンドに関する注意点

- 管理コンソールの場合、**default** プロファイルは、最初にコンソールにログインする際に選択されるプロファイルです。別のプロファイルでトランザクションマネージャーの設定を変更

する必要がある場合は、各指示で **default** の代わりにプロファイルを選択します。

同様に、**CLI** コマンド例で、プロファイルを **default** プロファイルに置き換えます。

- スタンドアロンサーバーを使用する場合は、存在するプロファイルは **1** つだけです。命令を無視して、特定のプロファイルを選択します。**CLI** コマンドで、サンプルコマンドの **/profile=default** 部分を削除します。



注記

管理コンソールまたは管理 **CLI** で **TM** オプションを表示するには、**transactions** サブシステムを有効にする必要があります。これはデフォルトで有効にされており、他の多くのサブシステムが適切に機能するために必要であるため、無効にされる可能性が非常に低くなります。

管理コンソールを使用した **TM** の設定

Web ベースの管理コンソールを使用して **TM** を設定するには、画面上部の **Configuration** タブを選択します。管理対象ドメインを使用する場合は、左上の **Profile** 選択ボックスから正しいプロファイルを選択します。**Container** メニューを展開し、**Transactions** を選択します。

ほとんどのオプションは、トランザクションマネージャーの設定ページに表示されます。リカバリー オプションはデフォルトで非表示になっています。**Recovery** タブをクリックしてリカバリーオプションを表示します。**Edit** をクリックしてオプションを編集します。変更は直ちに反映されます。

Need Help? ラベルをクリックして、インラインヘルプテキストを表示します。

管理 **CLI** を使用した **TM** の設定

管理 **CLI** では、一連のコマンドを使用して **TM** を設定できます。このコマンドはすべて、プロファイルが **/profile=default/subsystem=transactions/** の管理対象ドメインの場合は **default**、スタンドアロンサーバーの場合は **/subsystem=transactions** で始まります。

重要

*トランザクションログのストレージタイプとして **hornetq** ジャーナルを使用するよう **transaction** サブシステムが設定されている場合、**JBoss EAP** の **2** つのインスタンスは同じディレクトリーを使用してジャーナルを保存することはできません。アプリケーションサーバーインスタンスは同じ場所を共有することはできず、アプリケーションサーバーインスタンスは一意な場所を設定する必要があります。*

*表**19.1 TM** 設定オプション*

オプション	説明	CLI コマンド
統計の有効化 (Enable Statistics)	トランザクションの統計を有効にするかどうか。これらの統計は、**Runtime** タブの **Subsystem Metrics** セクションで管理コンソールで確認できます。	**/profile=default/subsystem=transactions/:write-attribute(name=enable-statistics,value=true)**
TSM ステータスの有効化	アウトオブプロセスのリカバリーに使用される TSM (トランザクションステータスマネージャー) サービスを有効にするかどうか。プロセスリカバリーマネージャーを実行して異なるプロセスから ActionStatusService にアクセスすることはサポートされません（通常はメモリーで接続されます）。	この設定オプションはサポート対象外です。
デフォルトのタイムアウト (Default Timeout)	デフォルトのトランザクションタイムアウトです。デフォルトでは **300** 秒に設定されています。トランザクションごとにプログラムで上書きできます。	**/profile=default/subsystem=transactions/:write-attribute(name=default-timeout,value=300)**
オブジェクトストアパス (Object Store Path)	TM オブジェクトストアがデータを格納するファイルシステムの相対または絶対パス。デフォルトでは、**object-store-relative-to** パラメーターの値に相対的です。	**/profile=default/subsystem=transactions/:write-attribute(name=object-store-path,value=tx-object-store)**
オブジェクトストアパスに相対的 (Object Store Path Relative To)	ドメインモデルのグローバルなパス設定を参照します。デフォルト値は JBoss EAP 6 のデータディレクトリーで、デフォルト値は **jboss.server.data.dir** プロパティーで、デフォルト値は ***EAP_HOME*/domain/data/**、スタンドアロンサーバーインスタンスの ***EAP_HOME*/standalone/data/** です。オブジェクトストア **object-store-path** TM 属性の値はこのパスに相対的です。	**/profile=default/subsystem=transactions/:write-attribute(name=object-store-relative-to,value=jboss.server.data.dir)**

オプション	説明	CLI コマンド
ソケットバインディング	ソケットベースのメカニズムが使用される場合に、トランザクションマネージャーがリカバリーおよびトランザクション識別子を生成するために使用されるソケットバインディングの名前を指定します。一意の識別子生成の詳細は、**process-id-socket-max-ports** を参照してください。ソケットバインディングは、管理コンソールの **Server** タブでサーバーグループごとに指定されます。	**/profile=default/subsystem=transactions/:write-attribute(name=socket-binding,value=txn-recovery-environment)**
ソケットバインディングのステータス	Transaction Status マネージャーに使用するソケットバインディングを指定します。	この設定オプションはサポート対象外です。
リカバリーリスナー (Recovery Listener)	トランザクションリカバリーのプロセスがネットワークソケットをリッスンするかどうかを指定します。デフォルトは **false** です。	**/profile=default/subsystem=transactions/:write-attribute(name=recovery-listener,value=false)**

*以下のオプションは高度な使用を目的としており、管理 **CLI** を使用した場合のみ変更できます。デフォルト設定から変更する場合は注意してください。詳細は、**Red Hat** グローバルサポートサービスにお問い合わせください。*

*表**19.2** 高度な **TM** 設定オプション*

オプション	説明	CLI コマンド
jts	Java Transaction Service (JTS) トランザクションを使用するかどうかを指定します。デフォルト値は **false** で、JTA トランザクションのみを使用します。	**/profile=default/subsystem=transactions/:write-attribute(name=jts,value=false)**

オプション	説明	CLI コマンド
node-identifier	トランザクションマネージャーのノード識別子。このオプションは以下の場合に必要になります。 ● JTS 対 JTS の通信 ● 2 つのトランザクションマネージャーが共有のリソースマネージャーにアクセスする場合 ● 2 つのトランザクションマネージャーが共有のオブジェクトマネージャーにアクセスする場合 リカバリー中にデータの整合性を強制する必要があるため、各トランザクションマネージャーの **node-identifier** は一意である必要があります。複数のノードが同じリソースマネージャーと対話したり、トランザクションオブジェクトストアを共有したりするため、**node-identifier** は JTA に対しても一意である必要があります。	**/profile=default/subsystem=transactions/:write-attribute(name=node-identifier,value=1)**
process-id-socket-max-ports	トランザクションマネージャーは、各トランザクションログに対して一意の識別子を作成します。一意の識別子を生成するメカニズムは 2 種類あります。 ソケットベースのメカニズムとプロセスのプロセス識別子をベースにしたメカニズムです。 ソケットベースの識別子の場合、あるソケットを開くと、そのポート番号が識別子に使用されます。ポートがすでに使用されている場合は、空きのポートが見つかるまで次のポートがプローブされます。**process-id-socket-max-ports** は、TM が失敗する前に試行するソケットの最大数を表します。デフォルト値は **10** です。	**/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-max-ports,value=10)**
process-id-uuid	プロセス識別子を使用して各トランザクションに一意の識別子を作成するには、**true** に設定します。そうでない場合は、ソケットベースのメカニズムが使用されます。デフォルトは **true** です。詳細は、**process-id-socket-max-ports** を参照してください。**process-id-socket-binding** を有効にするには、**process-id-uuid** を **false** に設定します。	**/profile=default/subsystem=transactions/:write-attribute(name=process-id-uuid,value=true)**

オプション	説明	CLI コマンド
process-id-socket-binding	トランザクションマネージャーがソケットベースのプロセス ID を使用する必要がある場合に使用するソケットバインディング設定の名前。**undefined** が **process-id-uuid** の場合、**true** になります。それ以外の場合には設定する必要があります。	**/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-binding,value=true)**
use-hornetq-store	トランザクションログには、ファイルベースのストレージの代わりに HornetQ のジャーナルストレージメカニズムを使用します。デフォルトでは無効になっていますが、I/O パフォーマンスが改善されます。個別のトランザクションマネージャーの JTS トランザクションを使用することは推奨されません。このオプションを変更する場合は、**shutdown** コマンドを使用してサーバーを再起動して変更を反映する必要があります。	**/profile=default/subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)**

バグの報告

19.1.3. JTA Transaction API を使用するようデータソースを設定

概要

*ここでは、データソースで **Java Transaction API (JTA)** を有効にする方法を説明します。*

前提条件

このタスクを続行するには、次の条件を満たしている必要があります。

-

データベースまたはその他のリソースが **Java Transaction API** をサポートする必要があります。不明な場合は、データベースまたはその他のリソースについてのドキュメントを参照してください。

- データソースを作成します。「管理インターフェースによる非 **XA** データソースの作成」を参照してください。
- **JBoss EAP 6** を停止します。
- テキストエディターで設定ファイルを直接編集できる権限を持たなければなりません。

手順**19.1 Java Transaction API** を使用するようデータソースを設定

1. テキストエディターで設定ファイルを開きます。

   JBoss EAP 6 を管理対象ドメインまたはスタンドアロンサーバーで実行するかによって、設定ファイルの場所は異なります。

   - 管理対象ドメイン

     管理対象ドメインのデフォルトの設定ファイルは、**Red Hat Enterprise Linux** の場合は **EAP_HOME/domain/configuration/domain.xml**、**Microsoft Windows Server** の場合は **EAP_HOME\domain\configuration\domain.xml** にあります。

   - スタンドアロンサーバー

     スタンドアロンサーバーのデフォルト設定ファイルは、**Red Hat Enterprise Linux** の場合は **EAP_HOME/standalone/configuration/standalone.xml**、**Microsoft Windows Server** の場合は **EAP_HOME\standalone\configuration\standalone.xml** にあります。

2. お使いのデータ ソースに対応する **<datasource >** タグを見つけます。

   データソースの **jndi-name** 属性は、作成時に指定した属性に設定されます。たとえば、**ExampleDS** データソースは以下のようになります。

```
<datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="H2DS"
enabled="true" jta="true" use-java-context="true" use-ccm="true">
```

3. **jta** 属性を **true** に設定します。

   直前の手順のように、**<datasource>** タグの内容に以下を追加します： **jta="true"**

   特定のユースケース（読み取り専用のデータソースの定義など）がない場合、**Red Hat** は、デフォルト値の **jta=true** を上書きすることを推奨していません。この設定は、データソースが **Java Transaction API** を反映し、**JCA** 実装による接続の追跡を改善できることを示します。

4. 設定を保存します。

   設定ファイルを保存しテキストエディターを終了します。

5. **JBoss EAP 6** を起動します。

   JBoss EAP 6 サーバーを再起動します。

結果

JBoss EAP 6 が起動し、データソースが **Java Transaction API** を使用するように設定されます。

バグの報告

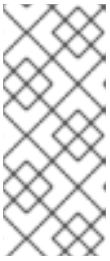
19.1.4. XA Datasource の設定

前提条件

管理コンソールへのログイン

1. 新しいデータソースを追加します。

   新しいデータソースを **JBoss EAP 6** に追加します。上部の **XA** データソース タブをクリックします。



注記

JBoss EAP 6 に新しいデータソース『を追加する方法については、『『Administration and Configuration Guide』』の「Creating an XA Datasource with the Management Interfaces』」を参照してください。

2. 必要に応じて他のプロパティーを設定します。

すべてのデータソースパラメーターは「データソースのパラメーター」に記載されています。

結果

XA Datasource が設定され、使用する準備ができます。

バグの報告

19.1.5. トランザクションログメッセージ

ログファイルが読み取り可能な状態でトランザクションの状態を追跡するには、トランザクションロガーに **DEBUG** ログレベルを使用します。詳細なデバッグには、**TRACE** ログレベルを使用します。トランザクションロガーの設定に関する詳細は、「トランザクションサブシステムのログ設定」を参照してください。

TRACE ログレベルにログインするように設定すると、トランザクションマネージャーは多くのロギング情報を生成できます。最も一般的なメッセージの一部は次のとおりです。このリストは包括的なものではないため、他のメッセージが表示されることもあります。

表19.3 トランザクション状態の変更

トランザクションの開始	トランザクションが開始されると、次のコードが実行されます。
	`com.arjuna.ats.arjuna.coordinator.BasicAction::Begin:1342`
	`tsLogger.logger.trace("BasicAction::Begin() for action-id "+ get_uid());`

トランザクションのコミット	トランザクションがコミットすると、次のコードが実行されます。 com.arjuna.ats.arjuna.coordinator.BasicAction::End:1342 tsLogger.logger.trace("BasicAction::End() for action-id "+ get_uid());
トランザクションのロールバック	トランザクションがロールバックすると、次のコードが実行されます。 com.arjuna.ats.arjuna.coordinator.BasicAction::Abort:1575 tsLogger.logger.trace("BasicAction::Abort() for action-id "+ get_uid());
トランザクションのタイムアウト	トランザクションがタイムアウトすると、次のコードが実行されます。 com.arjuna.ats.arjuna.coordinator.TransactionReaper::doCancellations:349 tsLogger.logger.trace("Reaper Worker " + Thread.currentThread() + " attempting to cancel " + e._control.get_uid()); この結果、上記のように同じスレッドがトランザクションをロールバックすることが示されます。

バグの報告

19.1.6. トランザクションサブシステムのログ設定

概要

*この手順を使用して、**JBoss EAP 6** の他のロギング設定とは関係なく、トランザクションに関するログに記録される情報量を制御します。メイン手順では、**Web** ベースの管理コンソールでこれを行う方法を説明します。管理 **CLI** コマンドは後で指定されます。*

手順19.2 管理コンソールを使用したトランザクションロガーの設定

1. *ログ設定領域に移動します。*

   *管理コンソールで、**Configuration** タブをクリックします。管理対象ドメインを使用する場合は、左上の **Profile** 選択ボックスから、設定するサーバープロファイルを選択します。*

Core メニューを展開し、*Logging* を選択します。

2. *com.arjuna* 属性を編集します。

Log Categories タブを選択します。*com.arjuna* を選択し、*Details* セクションで *Edit* を選択します。ここでは、クラス固有のロギング情報を追加できます。*com.arjuna* クラスがすでに存在します。ログレベルや親ハンドラーを使用するかどうかを変更できます。

ログレベル

ログレベルはデフォルトで *WARN* です。トランザクションは大量のロギング出力を生成する可能性があるため、標準のロギングレベルの意味はトランザクションロガーでは若干異なります。通常、選択したレベルよりも重大度の低いレベルでタグ付けされたメッセージは破棄されます。

トランザクションログのレベル (詳細度の高い順)

- *TRACE*
- *DEBUG*
- *INFO*
- *WARN*
- *ERROR*
- *FAILURE*

親ハンドラーの使用

ロガーが出力を親ロガーに送信するかどうか。デフォルトの動作は *true* です。

3. 変更は直ちに反映されます。

バグの報告

19.2. トランザクション管理

19.2.1. トランザクションの参照と管理

管理 **CLI** では、トランザクションレコードを参照および操作する機能がサポートされます。この機能は、トランザクションマネージャーと **JBoss EAP 6** の管理 **API** 間の対話によって提供されます。

トランザクションマネージャーは、保留中の各トランザクションとトランザクションに関連する参加者に関する情報を オブジェクトストア と呼ばれる永続ストレージに格納します。管理 **API** は、オブジェクトストアを **log-store** と呼ばれるリソースとして公開します。**probe** という **API** 操作はトランザクションログを読み取り、ログごとにノードを作成します。**probe** を更新する必要がある場合は、**log-store** コマンドを手動で呼び出すことができます。トランザクションログが即座に表示され非表示になるのは、正常な挙動です。

例**19.1** ログストアの更新

このコマンドは、管理対象ドメインでプロファイル **default** を使用するサーバーグループに対してログストアを更新します。スタンドアローンサーバーの場合は、コマンドから **profile=default** を削除します。

```
/profile=default/subsystem=transactions/log-store=log-store/:probe
```

例**19.2** 準備済みトランザクションすべての表示

準備済みトランザクションをすべて表示するには、最初にログストアを更新し（ 例**19.1**「ログストアの更新」を参照）、ファイルシステムの **ls** コマンドと同様に機能する以下のコマンドを実行します。

```
ls /profile=default/subsystem=transactions/log-store=log-store/transactions
```

各トランザクションが一意の識別子とともに表示されます。個々の操作は、各トランザクションに対して実行できます（ トランザクションの管理を参照）。

トランザクションの管理

トランザクションの属性を表示します。

JNDI 名、EIS 製品名およびバージョン、ステータスなどのトランザクションに関する情報を表示するには、:read-resource CLI コマンドを使用します。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource
```

トランザクションの参加者を表示します。

各トランザクションログには、participants と呼ばれる子要素が含まれます。この要素で read-resource CLI コマンドを使用して、トランザクションの参加者を確認します。参加者は JNDI 名によって識別されます。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=java\:\/JmsXA:read-resource
```

結果は以下のようになります。

```
{
    "outcome" => "success",
    "result" => {
        "eis-product-name" => "HornetQ",
        "eis-product-version" => "2.0",
        "jndi-name" => "java:/JmsXA",
        "status" => "HEURISTIC",
        "type" => "/StateManager/AbstractRecord/XAResourceRecord"
    }
}
```

ここで示された結果は HEURISTIC 状態にあり、リカバリーの対象となります。詳細は、トランザクションをリカバリーします。 を参照してください。

特別な場合では、ログに対応するトランザクションレコードがないオーファンレコード (XAResourceRecords) をオブジェクトストアに作成できます。たとえば、準備済みの XA リソースが TM の記録前にクラッシュし、ドメイン管理 API はアクセス不可能である場合などです。このようなレコードにアクセスするには、管理オプション expose-all-logs を true に設定する必要があります。このオプションは管理モデルには保存されず、サーバーが再起動されると false に戻ります。

```
/profile=default/subsystem=transactions/log-store=log-store:write-attribute(name=expose-all-logs, value=true)
```

トランザクションを削除します。

各トランザクションログは、トランザクションを表すトランザクションログを削除するために **:delete** 操作をサポートします。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete
```

トランザクションをリカバリーします。

各トランザクションの参加者は、**:recover CLI** コマンドを使用したリカバリーをサポートします。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=2:recover
```

ヒューリスティックトランザクションおよび参加者のリカバリー

- トランザクションの状態が **HEURISTIC** の場合、リカバリー操作は状態を **PREPARE** に変更し、リカバリーをトリガーします。
- トランザクションの参加者の **1** つがヒューリスティックな場合、リカバリー操作は **commit** 操作を再生しようとします。成功すると、参加者はトランザクションログから削除されます。これを確認するには、**:probe** で **log-store** 操作を再実行し、参加者がリストされていないことを確認します。最後の参加者が削除されると、トランザクションも削除されます。

リカバリーが必要なトランザクションの状態を更新します。

トランザクションをリカバリーする必要がある場合は、リカバリーを試行する前に **:refresh CLI** コマンドを使用してリカバリーが必要なことを確認できます。

```
/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=2:refresh
```

トランザクション統計情報の表示

トランザクションマネージャーの統計が有効になっていると、トランザクションマネージャーおよびトランザクションサブシステムに関する統計を表示できます。トランザクションマネージャーの統計を有効にする方法は、「トランザクションマネージャーの設定」 を参照してください。

管理コンソールまたは管理 **CLI** を使用して統計を表示できます。管理コンソールでは、**Runtime** → **Status** → サブシステム → ます。管理対象ドメインの各サーバーでは、トランザクションの統計を利用

*できます。別のサーバーの状態を表示するには、左側のメニューで **Change Server** を選択し、一覧からサーバーを選択します。*

*以下の表は、統計を表示する利用可能な各統計、その説明、および管理 **CLI** コマンドを示しています。*

*表**19.4** トランザクションサブシステム統計情報*

統計	説明	CLI コマンド
合計	このサーバー上でトランザクションマネージャーにより処理されるトランザクションの合計数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-transactions,include-defaults=true)
Committed	このサーバー上でトランザクションマネージャーにより処理されるコミット済みトランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-committed-transactions,include-defaults=true)
強制終了	このサーバー上でトランザクションマネージャーにより処理されるアボートされたトランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-aborted-transactions,include-defaults=true)

統計	説明	CLI コマンド
Timed Out	このサーバー上でトランザクションマネージャーにより処理されるタイムアウトのトランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-timed-out-transactions,include-defaults=true)
Heuristics	管理コンソールでは利用できません。ヒューリスティック状態のトランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-heuristics,include-defaults=true)
フライト状態のトランザクション	管理コンソールでは利用できません。開始済みであるが終了されていないトランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-inflight-transactions,include-defaults=true)
障害の原因 - アプリケーション	障害の原因がアプリケーションであった失敗トランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-application-rollbacks,include-defaults=true)

統計	説明	CLI コマンド
障害の原因 - リソース	障害の原因がリソースであった失敗トランザクションの数。	/host=*master*/server=*server-one*/subsystem=transactions/:read-attribute(name=number-of-resource-rollbacks,include-defaults=true)
参加者 ID	参加者の ID。	/host=*master*/server=*server-one*/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-children-names(child-type=participants)
トランザクションすべてのリスト	トランザクションの完全リスト。	/host=*master*/server=*server-one*/subsystem=transactions/log-store=log-store:read-children-names(child-type=transactions)

バグの報告

19.3. トランザクションに関するリファレンス

19.3.1. JBoss Transactions エラーと例外

***UserTransaction** クラスのメソッドによって発生する例外に関する詳細は、の 『**UserTransaction API**』 仕様 **http://docs.oracle.com/javaee/6/api/javax/transaction/UserTransaction.html** を参照してください。*

バグの報告

19.3.2. JTA トラザクションの制限

JTA トランザクションは、**JBoss EAP 6** の複数のインスタンス全体で完全にトランザクション分散を認識しません。現在の実装では、トランザクションコンテキストはリモート **EJB** 呼び出しとともに渡されます。ただし、これは、あるサーバーが別のサーバーを呼び出す単純なシナリオでのみ機能します。トランザクションディストリビューションに接続する複数のサーバーがある場合は、コンテキスト伝播が完全に安全ではありません。

完全なトランザクションディストリビューションのサポート動作については、**JTS** トランザクションを使用する必要があります。以下を含む **JTS** トランザクションを使用するには **ORB** を設定する必要があります。

- **JacORB** サブシステムでのトランザクションの有効化
- **JTS** トランザクションを使用するよう **Transaction** サブシステムを設定
- **IIOP** プロトコルを使用して **EJB** を呼び出します。
- 「**JTS** トランザクション用 **ORB** の設定」

バグの報告

19.4. ORB 設定

19.4.1. Common Object Request Broker Architecture (CORBA)

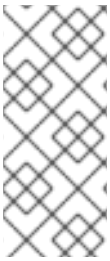
Common Object Request Broker Architecture(CORBA) は、アプリケーションとサービスが複数の互換性がない言語で記述され、異なるプラットフォームでホストされる場合でも、アプリケーションとサービスが連携できるようにする標準です。**CORBA** リクエストは **Object Request Broker(ORB)** と

呼ばれるサーバー側のコンポーネントによってブローカー化されます。**JBoss EAP 6** は、**JacORB** コンポーネントを用いて **ORB** インスタンスを提供します。

ORB は **Java Transaction Service(JTS)** トランザクションに対して内部的に使用され、独自のアプリケーションが使用することもできます。

バグの報告

19.4.2. jacorb の設定



注記

管理対象ドメインでは、**JacORB** サブシステムは **full** および **full-ha** プロファイルでのみ利用できます。スタンドアロンサーバーでは、**standalone-full.xml** または **standalone-full-ha.xml** 設定を使用する場合に利用できます。

jacorb プロパティーは、管理 **CLI** を使用して最も簡単に設定できます。一部の属性はサブシステムで直接設定され、その他の属性はシステムレベルで設定する必要があります。

サブシステムで直接設定できる設定を表示するには、以下の管理 **CLI** コマンドを使用します。

```
/subsystem=jacorb:read-resource(include-runtime=true, recursive=true)
```

現在の設定が一覧表示されます。

```
"add-component-via-interceptor" => "on",
"cache-poa-names" => "off",
"cache-typecodes" => "off",
"chunk-custom-rmi-valuetypes" => "on",
"client-requires" => "None",
"client-supports" => "MutualAuth",
"client-timeout" => 0,
"comet" => "off",
"export-corbaloc" => "on",
"giop-minor-version" => 2,
"indirection-encoding-disable" => "off",
"iona" => "off",
"lax-boolean-encoding" => "off",
"max-managed-buf-size" => 24,
"max-server-connections" => 2147483647,
"max-threads" => 32,
```

```
"monitoring" => "off",
"name" => "JBoss",
"outbuf-cache-timeout" => -1,
"outbuf-size" => 2048,
"pool-size" => 5,
"print-version" => "off",
"properties" => undefined,
"queue-max" => 100,
"queue-min" => 10,
"queue-wait" => "off",
"retries" => 5,
"retry-interval" => 500,
"root-context" => "JBoss/Naming/root",
"security" => "identity",
"security-domain" => undefined,
"server-requires" => "None",
"server-supports" => "MutualAuth",
"server-timeout" => 0,
"socket-binding" => "jacorb",
"ssl-socket-binding" => "jacorb-ssl",
"strict-check-on-tc-creation" => "off",
"sun" => "on",
"support-ssl" => "off",
"transactions" => "spec",
"use-bom" => "off",
"use-imr" => "off",
"ior-settings" => undefined
```

*その他の設定はシステムレベルで設定する必要があります。システムレベルの設定は **JBoss EAP** モデルでは透過的ではなく、**JacORB** サブシステムを相互に干渉すると表示されないことに注意してください。たとえば、上記の **jacorb:read-resource** コマンドを使用すると表示されません。*

*以下のコマンド例を使用して、システムレベルのプロパティーを使用して **JacORB** 属性を設定します。*

```
/system-property=jacorb.connection.client.pending_reply_timeout:add(value=600000)
```

```
/system-property=jacorb.connection.client.idle_timeout:add(value=120000)
```

```
/system-property=jacorb.connection.server.timeout:add(value=300000)
```

```
/system-property=jacorb.native_char_codeset:add(value=UTF8)
```

```
/system-property=jacorb.native_wchar_codeset:add(value=UTF16)
```

バグの報告

19.4.3. JTS トランザクション用 ORB の設定

JBoss EAP 6 のデフォルトインストールでは ORB が無効になります。コマンドライン管理 CLI を使用して ORB を有効にできます。

手順19.3 管理コンソールを使用した ORB の設定

1. プロファイル設定の表示

   管理コンソールの上部から Configuration を選択します。管理対象ドメインを使用する場合は、左上の選択ボックスから full または full-ha プロファイルを選択します。

2. Initializers 設定の変更

   Subsystems メニューを展開します。Container メニューを展開し、JacORB を選択します。

   メイン画面に表示されるフォームで、Initializers タブを選択し、Edit ボタンをクリックします。

   security の値を on に設定して、セキュリティーインターセプターを 有効 に します。

   JTS に対して ORB を有効にするには、Transaction Interceptors の値をデフォルトの 仕様 ではなく on に設定します。

   これらの値の詳細については、フォームの Need Help? リンクを参照してください。値の編集が完了したら、Save をクリックします。

3. 高度な ORB 設定

   高度な設定オプションについては、フォームの他のセクションを参照してください。各セクションには Need Help? リンクと、パラメーターに関する詳細情報が含まれています。

管理 CLI を使用して ORB を設定

管理 CLI を使用して ORB を設定できます。以下のコマンドは、管理コンソールのイニシャライザーを上記の手順と同じ値に設定します。これは、JTS と使用するために行う ORB の最低限の設定です。

これらのコマンドは、full プロファイルを使用して管理対象ドメインに対して設定されます。必要な

場合は設定に応じてプロファイルを変更してください。スタンドアロンサーバーを使用する場合は、コマンドの **/profile=full** 部分を省略します。

例**19.3** セキュリティーインターセプターの有効化

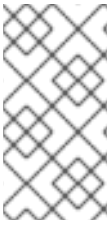
```
/profile=full/subsystem=jacorb/:write-attribute(name=security,value=on)
```

例**19.4 JacORB** サブシステムでのトランザクションの有効化

```
/profile=full/subsystem=jacorb/:write-attribute(name=transactions,value=on)
```

例**19.5** トランザクションサブシステムでの **JTS** の有効化

```
/profile=full/subsystem=transactions:write-attribute(name=jts,value=true)
```



注記

JTS をアクティベートするにはリロードでは不十分なため、サーバーを再起動する必要があります。

バグの報告

19.5. JDBC オブジェクトストアのサポート

19.5.1. トランザクションの JDBC ストア

要件:

- 「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」

トランザクションは **JDBC** データソースをオブジェクトストアとして使用できます。使用するデータベースがフェイルオーバーとリカバリー用に設定されている場合、アプリケーションサーバーでディスク領域を使用することよりも優れたオプションとなる可能性があります。生の **JDBC** オブジェクトストアは特別なオブジェクトストアであり、ファイルシステムまたは **HornetQ** ジャーナルオブジェクトストアと同様に動作しない可能性があるという点に対して、短所で重み付けする必要があります。

注記

***Transactions** オブジェクトストアとして使用される **JDBC** データソースは、サーバーの設定ファイルの データソース セクションに **jta="false"** を指定* **する必要があり** *ます。*

*手順**19.4** **JDBC** データソースをトランザクションオブジェクトストアとして使用*

1. ***use-jdbc-store** を **true** に設定します。*

```
/subsystem=transactions:write-attribute(name=use-jdbc-store, value=true)
```

2. ***jdbc-store-datasource** を使用するデータソースの **JNDI** 名に設定します。*

```
/subsystem=transactions:write-attribute(name=jdbc-store-datasource,
value=java:jboss/datasources/TransDS)
```

3. ***JBoss EAP** サーバーを再起動し、変更を反映します。*

```
shutdown --restart=true
```

すべての属性は次のとおりです。

*表**19.5** トランザクション **JDBC** ストアのプロパティー*

プロパティー	説明
use-jdbc-store	true に設定すると、トランザクションの JDBC ストアが有効になります。
jdbc-store-datasource	ストレージに使用される JDBC データソースの JNDI 名。
jdbc-action-store-drop-table	起動時にアクションストアテーブルをドロップおよび再作成します。任意設定、デフォルトは「false」です。
jdbc-action-store-table-prefix	アクションストアテーブル名の接頭辞。オプション。
jdbc-communication-store-drop-table	起動時にコミュニケーションストアテーブルがドロップおよび再作成されます。任意設定、デフォルトは「false」です。

プロパティー	説明
jdbc-communication-store-table-prefix	コミュニケーションストアテーブル名の接頭辞。オプション。
jdbc-state-store-drop-table	起動時にステートストアテーブルをドロップおよび再作成します。任意設定、デフォルトは「false」です。
jdbc-state-store-table-prefix	ステートストアテーブル名の接頭辞。オプション。

以下も参照してください。

- *「**JTA Transaction API** を使用するようデータソースを設定」*

バグの報告

第20章 メールサブシステム

20.1. メールサブシステムでのカスタムトランスポートの使用

標準的なメールサーバー (POP3、IMAP) を使用する場合、定義可能な属性が複数あり、その一部は必須の属性になります。

最も重要な属性は outbound-socket-binding-ref で、アウトバウンドメールソケットバインディングへの参照で、ホストアドレスとポート番号で定義されます。

ロードバランシングの目的でホスト設定で複数のホストが使用されるため、一部のユーザーは最も効果的なソリューションではありません。ただし、この設定は標準の JavaMail ではサポートされないため、一部のユーザーがカスタムメールトランスポートを実装する必要があります。

これらのカスタムトランスポートは outbound-socket-binding-ref を必要とせず、カスタムのホストプロパティー形式を許可します。

以下のコマンドを使用すると、CLI よりカスタムトランスポートを設定できます。

手順20.1

1. 新しいメールセッションを追加します。以下のコマンドは mySession という新しいセッションを作成し、JNDI を java:jboss/mail/MySession に設定します。

```
/subsystem=mail/mail-session=mySession:add(jndi-name=java:jboss/mail/MySession)
```

2. アウトバウンドソケットバインディングを追加します。以下のコマンドは、localhost:25 を参照する my-smtp-binding という名前のソケットバインディングを追加します。

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-binding=my-smtp-binding:add(host=localhost, port=25)
```

3. outbind-socket-binding-ref を使用して SMTP サーバーを追加します。以下のコマンドは、my-smtp-binding という SMTP を追加し、ユーザー名、パスワード、および TLS 設定を定義します。

```
/subsystem=mail/mail-session=mySession/server=smtp:add(outbound-socket-
binding-ref= my-smtp-binding, username=user, password=pass, tls=true)
```

4. *POP3* と *IMAP* に対して、同じ処理を行います。

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=my-pop3-binding:add(host=localhost, port=110)
```

```
/subsystem=mail/mail-session=mySession/server=pop3:add(outbound-socket-
binding-ref=my-pop3-binding, username=user, password=pass)
```

```
/socket-binding-group=standard-sockets/remote-destination-outbound-socket-
binding=my-imap-binding:add(host=localhost, port=143)
```

```
/subsystem=mail/mail-session=mySession/server=imap:add(outbound-socket-
binding-ref=my-imap-binding, username=user, password=pass)
```

5. カスタムサーバーを使用するには、アウトバウンドソケットバインディングのないカスタムメールサーバーを新規作成し (アウトバウンドソケットバインディングは任意であるため)、代わりにホスト情報をプロパティーの一部として指定します。

```
/subsystem=mail/mail-
session=mySession/custom=myCustomServer:add(username=user,password=pass,
properties={"host" => "myhost", "my-property" =>"value"})
```

カスタムプロトコルを定義するとき、ピリオド(.)が含まれるプロパティー名は完全修飾名とみなされ、指定時に渡されます。その他の形式（例：**my-property** ）は、**mail.server-name.my-property** の形式に変換されます。

custom-server 属性のカスタム形式を強調する、完全な **XML** 設定の例は次のとおりです。

```
<subsystem xmlns="urn:jboss:domain:mail:1.1">
  <mail-session jndi-name="java:/Mail" from="user.name@domain.org">
    <smtp-server outbound-socket-binding-ref="mail-smtp" tls="true">
      <login name="user" password="password"/>
    </smtp-server>
    <pop3-server outbound-socket-binding-ref="mail-pop3"/>
    <imap-server outbound-socket-binding-ref="mail-imap">
      <login name="nobody" password="password"/>
    </imap-server>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Default">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
  <mail-session debug="true" jndi-name="java:jboss/mail/Custom">
```

```
        <custom-server name="smtp">
            <login name="username" password="password"/>
            <property name="host" value="mail.example.com"/>
        </custom-server>
        <custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
            <property name="custom_prop" value="some-custom-prop-value"/>
            <property name="some.fully.qualified.property" value="fully-qualified-prop-name"/>
        </custom-server>
    </mail-session>
    <mail-session debug="true" jndi-name="java:jboss/mail/Custom2">
        <custom-server name="pop3" outbound-socket-binding-ref="mail-pop3">
            <property name="custom_prop" value="some-custom-prop-value"/>
        </custom-server>
    </mail-session>
</subsystem>
```

バグの報告

第21章 ENTERPRISE JAVABEANS 3.2

21.1. はじめに

21.1.1. Enterprise JavaBeans の概要

Enterprise JavaBeans(EJB)3.1 は、**Enterprise Beans** と呼ばれるサーバー側のコンポーネントを使用して、分散型のトランザクション型のセキュアで移植可能な **Java EE** アプリケーションを開発するための **API** です。エンタープライズ **Bean** は、再利用を促すような分離方法でアプリケーションのビジネスロジックを実装します。**Enterprise JavaBeans 3.1** は、**Java EE** 仕様 **JSR-318** として文書化されています。

JBoss EAP 6 では、**Enterprise JavaBeans 3.1** 仕様を使用してビルドされたアプリケーションが完全にサポートされます。

バグの報告

21.1.2. 管理者向け Enterprise JavaBeans の概要

JBoss 管理者は、**JBoss EAP 6** でのエンタープライズ **Bean** のパフォーマンスを制御する多くの設定オプションを使用できます。これらのオプションは、管理コンソールまたはコマンドライン設定ツールを使用してアクセスできます。変更を適用するために **XML** サーバー設定ファイルを編集することもできますが、推奨されません。

EJB 設定オプションは、サーバーがどのように実行されているかによって、管理コンソールでの場所が若干異なります。

1. 管理コンソールの上部にある **Configuration** タブをクリックします。
2. ドメインモードで実行している場合は、左上の **Profiles** ドロップダウンメニューからプロファイルを選択します。
3. **Subsystems** メニューを展開します。
4. **Container** メニューを展開し、**EJB 3** を選択します。

バグの報告

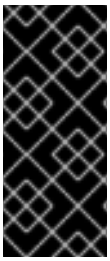
21.1.3. エンタープライズ Bean

エンタープライズ **Bean** は、**Enterprise JavaBeans(EJB)3.1** 仕様 **JSR-318** に定義されているサーバー側のアプリケーションコンポーネントです。エンタープライズ **Bean** は、再利用を促すために、分離された方法でアプリケーションのビジネスロジックを実装するように設計されています。

エンタープライズ **Bean** は **Java** クラスとして記述され、適切な **EJB** アノテーションが付けられます。**Java EE** アプリケーションの一部としてデプロイすることも、独自のアーカイブ（**JAR** ファイル）でアプリケーションサーバーにデプロイできます。アプリケーションサーバーは、各エンタープライズ **bean** のライフサイクルを管理し、セキュリティー、トランザクション、同時実行管理などのサービスを提供します。

また、エンタープライズ **Bean** は任意の数のビジネスインターフェースを定義できます。ビジネスインターフェースは、クライアントで利用できる **Bean** のメソッドの制御を強化し、リモート **JVM** で実行しているクライアントへのアクセスも許可します。

エンタープライズ **Bean** には、セッション **Bean**、メッセージ駆動型 **Bean**、およびエンティティー **Bean** の **3** 種類があります。



重要

エンティティー **bean** は **EJB 3.1** で非推奨となり、**Red Hat** は代わりに **JPA** エンティティーの使用を推奨します。**Red Hat** は、レガシーシステムとの後方互換性のためにエンティティー **Bean** のみを使用することを推奨します。

バグの報告

21.1.4. セッション Bean

セッション **Bean** は、関連するビジネスプロセスまたはタスクのセットをカプセル化し、それらを要求するクラスに挿入されるエンタープライズ **Bean** です。セッション **bean** のタイプには、ステートレス、ステートフル、シングルトンの **3** つがあります。

バグの報告

21.1.5. メッセージ駆動 Bean

メッセージ駆動 Bean (MDB) は、アプリケーション開発のイベント駆動モデルを提供します。MDB のメソッドは、クライアントコードにインジェクトされず、クライアントコードから呼び出しませんが、Java Messaging Service (JMS) サーバーなどのメッセージングサービスからのメッセージの受信によってトリガーされます。Java EE 6 仕様では JMS がサポートされる必要がありますが、他のメッセージングシステムもサポートされます。

バグの報告

21.2. BEAN プールの設定

21.2.1. Bean プール

JBoss EAP 6 は、パフォーマンスを強化するために、デプロイされたステートレスエンタープライズ Bean の複数のインスタンスをメモリーに保持します。この手法は Bean プーリング と呼ばれます。Bean が必要な場合、アプリケーションサーバーは新しい Bean をインスタンス化する代わりに、すでに利用可能な Bean の適切なプールから Bean を 1 つ取ります。Bean が不要になると、再利用するためにプールに戻されます。

ステートレスセッション Bean と メッセージ駆動型 Bean の Bean プールは別々に設定および維持されます。

@org.jboss.ejb3.annotation.Pool アノテーションを EJB で使用して、その EJB に使用する必要のあるプールを特定することができます。このアノテーションは、そのプールの名前を参照します。

バグの報告

21.2.2. Bean プールの作成

管理コンソールと CLI ツールを使用すると Bean プールを作成できます。

テキストエディターを使用して、必要な Bean プール設定をサーバー設定ファイルに追加して Bean プールを作成することもできます。例21.2「XML 設定の例」 は、この設定の一例です。

手順21.1 管理コンソールを使用した Bean プールの作成

1.

管理コンソールへログインします。「管理コンソールへのログイン」 を参照してください。

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Bean Pools** タブを選択します。

3. **Add** をクリックします。**Add EJB3 Bean Pools** ダイアログが表示されます。

4. 必要な詳細、名前、最大 プールサイズ、タイムアウト、およびタイムアウト 単位を指定します。

5. **Save** ボタンをクリックして終了します。

手順**21.2 CLI** を使用した **Bean** プールの作成

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. 以下の構文で **add** 操作を使用します。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:add(max-pool-size=MAXSIZE, timeout=TIMEOUT, timeout-unit="UNIT")
```

- **BEANPOOLNAME** を **Bean** プールに必要な名前に置き換えます。
- **MAXSIZE** を **Bean** プールの最大サイズに置き換えます。
- **TIMEOUT**の置き換え
- **UNIT** を必要な時間単位に置き換えます。使用できる値は **NANOSECONDS**、**MICROSECONDS**、**MILLISECONDS**、**SECONDS**、**MINUTES**、**HOURS**、**DAYS** です。

3. ***read-resource*** *操作を使用して* ***Bean*** *プールの作成を確認します。*

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

*例**21.1 CLI** を使用した **Bean** プールの作成*

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=ACCTS_BEAN_POOL:add(max-pool-size=500, timeout=5000, timeout-unit="SECONDS")
{"outcome" => "success"}
```

*例**21.2 XML** 設定の例*

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">

  <pools>

    <bean-instance-pools>

      <strict-max-pool  name="slsb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />

      <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20"
        instance-acquisition-timeout="5"
        instance-acquisition-timeout-unit="MINUTES" />

    </bean-instance-pools>

  </pools>

</subsystem>
```

バグの報告

*21.2.3. **Bean** プールの削除*

*管理コンソールを使用して未使用の **Bean** プールを削除することが可能です。*

要件:

- *使用中の **Bean** プールを削除することはできません。使用されていないことを確認するには、「セッションおよびメッセージ駆動型 **Bean** に対する **Bean** プールの割り当て」 を参照してください。*

手順21.3 管理コンソールを使用した Bean プールの削除

1. *管理コンソールへログインします。「管理コンソールへのログイン」 を参照してください。*

2. *画面上部の Configuration タブをクリックします。Container メニューを展開し、EJB 3 を選択します。Bean Pools タブを選択します。*

3. *一覧より削除する Bean プールを選択します。*

4. *Remove をクリックします。Remove Item ダイアログが表示されます。*

5. *確認 をクリックして確定します。*

手順21.4 CLI を使用した Bean プールの削除

1. *CLI ツールを起動し、サーバーに接続します。「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」 を参照してください。*

2. *以下の構文で remove 操作を使用します。*

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:remove
```

- *BEANPOOLNAME を Bean プールに必要な名前に置き換えます。*

例21.3 CLI を使用した Bean プールの削除

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=ACCTS_BEAN_POOL:remove
{"outcome" => "success"}
```

バグの報告

21.2.4. Bean プールの編集

管理コンソールを使用して **Bean** プールを編集することが可能です。

手順**21.5** 管理コンソールを使用した **Bean** プールの編集

1. 管理コンソールへログインします。「管理コンソールへのログイン」

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Bean Pools** タブを選択します。

3. 編集する **Bean** プールを選択します。

4. **Edit** をクリックします。

5. 変更する詳細を編集します。**Max Pool Size**、**timeout** 値、および **Timeout Unit** のみを変更できます。

6. **Save** をクリックして終了します。

手順**21.6 CLI** を使用した **Bean** プールの編集

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. **Bean** プールの各属性を変更するために、次の構文で **write-attribute** 操作を使用します。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:write-attribute(name="ATTRIBUTE", value="VALUE")
```

- **BEANPOOLNAME** を **Bean** プールに必要な名前に置き換えます。
- **ATTRIBUTE** を、編集する属性の名前に置き換えます。この方法で編集できる属性は、**max-pool-size**、**timeout**、および **timeout-unit** です。

- *VALUE は、属性の必要な値に置き換えます。*

3. *read-resource 操作を使用して Bean プールへの変更を確認します。*

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

例21.4 CLI を使用した Bean プールのタイムアウト値の設定

```
[standalone@localhost:9999 /] /subsystem=ejb3/strict-max-bean-instance-
pool=HSBeanPool:write-attribute(name="timeout", value="1500")
{"outcome" => "success"}
```

バグの報告

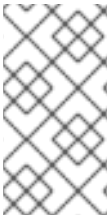
21.2.5. セッションおよびメッセージ駆動型 Bean に対する Bean プールの割り当て

JBoss 管理者は、セッション Bean およびメッセージ駆動型 Bean によって使用される個別の Bean プールを割り当てることができます。管理コンソールまたは管理 CLI を使用すると Bean プールを割り当てることができます。

デフォルトでは、ステートレスセッション Bean とメッセージ駆動型 Bean の slsb-strict-max-pool と mdb-strict-max-pool の 2 つの Bean プールが提供されます。

Bean プールを作成または編集するには、「Bean プールの作成」 および 「Bean プールの編集」を参照してください。

また、EJB で @Pool アノテーションを使用して、その EJB に使用するプールを特定できます。



注記

特定の EJB で @Pool アノテーションを使用すると、管理インターフェースを使用して指定されたデフォルト設定が上書きされます。

手順21.7 管理コンソールを使用したセッションおよびメッセージ駆動型 Bean に対する Bean プールの割り当て

1. 管理コンソールへログインします。「管理コンソールへのログイン」

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Container** タブを選択します。

3. **Edit** をクリックします。

4. 適切なコンボボックスから、**Bean** の各タイプに使用する **Bean** プールを選択します。

5. **Save** をクリックして終了します。

手順21.8 CLI を使用したセッションおよびメッセージ駆動型 Bean に対する Bean プールの割り当て

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. 次の構文で **write-attribute** 操作を使用します。

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value="BEANPOOL")
```

   - **BEANTYPE** は、メッセージ駆動型 **Bean** の **default-mdb-instance-pool**、ステートレスセッション **Bean** の **default-slsb-instance-pool** に置き換えます。
   - **BEANPOOL** を割り当てる **Bean** プールの名前に置き換えます。

3. **read-resource** 操作を使用して変更を確認します。

```
/subsystem=ejb3:read-resource
```

例21.5 CLI を使用したセッション Bean に対する Bean プールの割り当て

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-slsb-instance-pool", value="LV_SLSB_POOL")
{"outcome" => "success"}
```

例**21.6 XML** 設定の例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>
```

手順**21.9 @Pool** アノテーションを使用したセッションまたはメッセージ駆動型 **Bean** に対する **Bean** プールの割り当て

1. **@Pool** アノテーションを **Bean** に追加し、使用される **Bean** プールの名前を指定します。

```
@Stateless
@Pool("slsb-strict-max-pool")
public class HelloBean implements HelloBeanRemote {
```

これにより、管理インターフェースで作成されたデフォルト設定が上書きされます。

2. **@org.jboss.ejb3.annotation.Pool** アノテーションは **JBoss EJB3** 外部 **API** の一部であり、依存関係として追加する必要があります。**Maven** を使用している場合は、以下の依存関係を **pom.xml** ファイルに追加する必要があります。

```
<dependency>
  <groupId>org.jboss.ejb3</groupId>
  <artifactId>jboss-ejb3-ext-api</artifactId>
  <version>2.1.0</version>
</dependency>
```

バグの報告

21.3. EJB スレッドプールの設定

21.3.1. エンタープライズ Bean スレッドプール

JBoss EAP 6 は、数多くの Java スレッドオブジェクトインスタンスをメモリー内に維持します。これらは、リモート呼び出し、タイマーサービス、非同期呼び出しなどの Enterprise Bean サービスによって使用されます。

この手法は スレッドプール と呼ばれます。スレッド作成のオーバーヘッドを排除することでパフォーマンスを向上し、リソースの使用状況を制御するメカニズムをシステム管理者に提供します。

異なるパラメーターを使用して複数のスレッドプールを作成し、各サービスを異なるスレッドプールに割り当てることが可能です。

バグの報告

21.3.2. スレッドプールの作成

管理コンソールまたは CLI を使用して EJB スレッドプールを作成することが可能です。

手順21.10 管理コンソールを使用した EJB スレッドプールの作成

1. 管理コンソールへログインします。「管理コンソールへのログイン」
2. 画面上部の Configuration タブをクリックします。
3. Container メニューを展開し、EJB 3 を選択します。
4. Thread Pools タブを選択し、Add をクリックします。
5. Name および Max Threads 値を指定します。

6. **Save** をクリックして終了します。

手順**21.11 CLI** を使用したスレッドプールの作成

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. 以下の構文で **add** 操作を使用します。

```
/subsystem=ejb3/thread-pool=THREAD_POOL_NAME:add(max-threads=MAX_SIZE)
```

- **THREAD_POOL_NAME** をスレッドプールの名前に置き換えます。
- **MAX_SIZE** はスレッドプールの最大サイズに置き換えます。

3. **read-resource** 操作を使用して **Bean** プールの作成を確認します。

```
/subsystem=ejb3/thread-pool=THREAD_POOL_NAME:read-resource
```

例**21.7 CLI** を使用したスレッドプールの作成

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=my-test-pool:add(max-threads=20)
{"outcome" => "success"}
```

例**21.8 XML** 設定の例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.5">
  ...
  <thread-pools>
    ...
    <thread-pool name="my-test-pool" max-threads="20"/>
  </thread-pools>
  ...
</subsystem>
```

バグの報告

21.3.3. スレッドプールの削除

管理コンソールを使用して未使用の **EJB** スレッドプールを削除することが可能です。

前提条件

- 使用中のスレッドプールを削除することはできません。以下のタスクを参照して、スレッドプールが使用されていないことを確認します。
  - 「**EJB3** タイマーサービスの設定」
  - 「**EJB3** 非同期呼び出しサービスのスレッドプールの設定」
  - 「**EJB3** リモートサービスの設定」

手順**21.12** 管理コンソールを使用した **EJB** スレッドプールの削除

1. 管理コンソールへログインします。「管理コンソールへのログイン」.
2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Thread Pools** タブを選択します。
3. 削除するスレッドプールを選択します。
4. **Remove** をクリックします。**Remove Item** ダイアログが表示されます。
5. **Confirm** をクリックします。

手順**21.13 CLI** を使用したスレッドプールの削除

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. 以下の構文で **remove** 操作を使用します。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:remove
```

- **THREADPOOLNAME** をスレッドプールの名前に置き換えます。

例**21.9 CLI** を使用したスレッドプールの削除

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=ACCTS_THREADS:remove
{"outcome" => "success"}
```

バグの報告

21.3.4. スレッドプールの編集

管理コンソールと **CLI** を使用すると、**JBoss** の管理者によるスレッドプールの編集が可能になります。

手順**21.14** 管理コンソールを使用したスレッドプールの編集

1. 管理コンソールへログインします。「管理コンソールへのログイン」.

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Thread Pools** タブを選択します。

3. 編集するスレッドプールを選択します。

4. **Edit** をクリックします。

5. 変更する詳細を編集します。**Thread Factory**、**Max Threads**、**Keepalive Timeout**、および **Keepalive Timeout Unit** の値のみを編集できます。

6. **Save** をクリックして終了します。

手順21.15 CLI を使用したスレッドプールの編集

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. スレッドプールの各属性を変更するために、次の構文で **write_attribute** 操作を使用します。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-
attribute(name="ATTRIBUTE", value="VALUE")
```

- **THREADPOOLNAME** をスレッドプールの名前に置き換えます。
- **ATTRIBUTE** を、編集する属性の名前に置き換えます。この方法で編集できる属性は、**keepalive-time**、**max-threads**、および **thread-factory** です。
- **VALUE** は、属性の必要な値に置き換えます。

3. **read-resource** 操作を使用して、スレッドプールへの変更を確認します。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:read-resource
```

重要

keepalive-time 属性の値を **CLI** で変更する場合、必要な値はオブジェクト表現です。構文は以下のようになります。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-
attribute(name="keepalive-time", value={"time" => "VALUE","unit" => "UNIT"}
```

例21.10 CLI を使用したスレッドプールの最大値の設定

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-
attribute(name="max-threads", value="50")
{"outcome" => "success"}
```

例21.11 CLI を使用したスレッドプールの keepalive-time 時間値の設定

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=HSThreads:write-
attribute(name="keepalive-time", value={"time"=>"150"})
{"outcome" => "success"}
```

バグの報告

21.4. セッション BEAN の設定

21.4.1. セッション Bean のアクセスタイムアウト

ステートフルおよびシングルトンセッション Bean には、同時アクセスを管理するためにアクセスタイムアウト値が指定されます。この値は、タイムアウトするまでにセッション Bean メソッドへのリクエストをブロックできる期間です。

タイムアウト値と時間の単位は、メソッドの @javax.ejb.AccessTimeout アノテーションを使用して指定できます。セッション Bean（すべての Bean のメソッドに適用）および特定のメソッドに指定して、Bean の設定をオーバーライドすることができます。

タイムアウトの値が指定されていない場合、JBoss EAP 6 はデフォルトのタイムアウト値である 5000 ミリ秒を使用します。

AccessTimeout の Javadocs を参照してください。
http://docs.oracle.com/javaee/6/api/javax/ejb/AccessTimeout.html

バグの報告

21.4.2. デフォルトセッション Bean アクセスタイムアウト値の設定

JBoss 管理者は、シングルトンおよびステートフルセッション Bean のデフォルトのタイムアウト値を指定できます。デフォルトのタイムアウト値は管理コンソールまたは CLI を使用して変更できます。デフォルト値は 5000 ミリ秒です。

手順21.16 管理コンソールを使用してデフォルトのセッション Bean アクセスタイムアウト値を設定

1. 管理コンソールへログインします。「管理コンソールへのログイン」を参照してください。

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Container** タブを選択します。

3. **Edit** をクリックします。**Details** エリアのフィールドを編集できるようになりました。

4. **Stateful Access Timeout** や **Singleton Access Timeout** のテキストボックスに、必要な値を入力します。

5. **Save** をクリックして終了します。

手順21.17 CLI を使用したセッション Bean のアクセスタイムアウト値の設定

1. **CLI** ツールを起動し、サーバーに接続します。「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」を参照してください。

2. 次の構文で **write-attribute** 操作を使用します。

```
/subsystem=ejb3:write-attribute(name="BEANTYPE", value=TIME)
```

   - **BEANTYPE** は、ステートフルセッション **Bean** の **default-stateful-bean-access-timeout**、シングルトンセッション **Bean** の **default-singleton-bean-access-timeout** に置き換えます。
   - **TIME** を必要なタイムアウト値に置き換えます。

3. **read-resource** 操作を使用して変更を確認します。

```
/subsystem=ejb3:read-resource
```

例**21.12 CLI** を使用してデフォルトのステートフル **Bean** のアクセスタイムアウト値を **9000** に設定する

```
[standalone@localhost:9999 /] /subsystem=ejb3:write-attribute(name="default-stateful-bean-access-timeout", value=9000)
{"outcome" => "success"}
```

例**21.13 XML** 設定の例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <session-bean>
    <stateless>
      <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
    </stateless>
    <stateful default-access-timeout="5000" cache-ref="simple"/>
    <singleton default-access-timeout="5000"/>
  </session-bean>

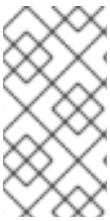
</subsystem>
```

バグの報告

21.4.3. セッション Bean トランザクションタイムアウト

TransactionTimeout アノテーションは、特定のメソッドのトランザクションタイムアウトを指定するために使用されます。アノテーションの値は指定のユニット要素で使用されるタイムアウトです。正の整数または **0** である必要があります。**0** が指定されている場合は常に、デフォルトのドメイン設定タイムアウトが使用されます。

unit 要素は、値の計測値を指定します。



注記

計算された値が正しければ、秒未満の測定がエラーとみなされます。以下に例を示します。 **@TransactionTimeout(value = 1000, unit=TimeUnit.MILISECONDS)**

デプロイメント記述子でのトランザクションタイムアウトの指定

trans-timeout 要素は、ビジネス、ホーム、コンポーネント、およびメッセージリスナーインターフェースのメソッドのトランザクションタイムアウトを定義するために使用されます。インターフェース表示メソッド、**Web** サービスエンドポイントメソッド、およびタイムアウトコールバックメソッド。**trans-timeout** 要素は **urn:trans-timeout** 名前空間にあり、**jboss** 名前空間に定義されている標準の **container-transaction** 要素に含まれます。

例**21.14 TRANS-timeout XML** 設定例

```
<ejb-name>*</ejb-name>
<tx:trans-timeout>
<tx:timeout>2</tx:timeout>
<tx:unit>Seconds</tx:unit>
</tx:trans-timeout>
```

ejb-name は、特定の **EJB** 名またはワイルドカード**(*)**に指定できます。**ejb-name** にワイルドカード**(*)**を指定すると、この特定のトランザクションタイムアウトはアプリケーション内のすべての **EJB** のデフォルトになります。

バグの報告

21.4.4. ステートフルセッション Bean キャッシュの設定

JBoss EAP 6 では、ステートフル **EJB** キャッシュはサーバー設定ファイルの **ejb3** サブシステムに設定されます。以下の手順では、ステートフル **EJB** キャッシュおよびステートフルタイムアウトを設定する方法を説明します。

手順**21.18** ステートフル **EJB** キャッシュの設定

1. サーバー設定ファイルの **ejb3** サブシステムで **<caches>** 要素を見つけます。**<cache>** 要素を追加します。以下の例では、「**my=cache**」という名前のキャッシュを作成します。

```
<cache name="my-cache" passivation-store-ref="my-cache-file" aliases="my-custom-cache"/>
```

2. サーバー設定ファイルの **ejb3** サブシステムで **<passivation-stores>** 要素を見つけます。前のステップで定義したキャッシュ用に **<file-passivation-store>** を作成します。

```
<file-passivation-store name="my-cache-file" idle-timeout="1260" idle-timeout-
unit="SECONDS" max-size="200"/>
```

3.

ejb3 サブシステムの設定が以下の例のようになるはずです。

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.4">
 ...
 <caches>
  <cache name="simple" aliases="NoPassivationCache"/>
  <cache name="passivating" passivation-store-ref="file" aliases="SimpleStatefulCache"/>
  <cache name="clustered" passivation-store-ref="infinispan" aliases="StatefulTreeCache"/>
  <cache name="my-cache" passivation-store-ref="my-cache-file" aliases="my-custom-
cache"/>
 </caches>
 <passivation-stores>
  <file-passivation-store name="file" idle-timeout="120" idle-timeout-unit="SECONDS" max-
size="500"/>
  <cluster-passivation-store name="infinispan" cache-container="ejb"/>
  <file-passivation-store name="my-cache-file" idle-timeout="1260" idle-timeout-
unit="SECONDS" max-size="200"/>
 </passivation-stores>
 ...
</subsystem>
```

パッシベーションキャッシュ **"my-cache"** は「**my-cache-file**」パッシベーションストアに設定されたようにステートフルセッション **Bean** をファイルシステムに渡します。これには、**idle-timeout**、**idle-timeout-unit**、および **max-size** オプションが含まれます。

4.

EJB JAR の **META -INF/** ディレクトリーに **jboss-ejb3.xml** ファイルを作成します。以下の例は、前のステップで定義されたキャッシュを使用するように **EJB** を設定します。

```
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:c="urn:ejb-cache:1.0"
    xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
    version="3.1"
    impl-version="2.0">
 <assembly-descriptor>
  <c:cache>
   <ejb-name>*</ejb-name>
   <c:cache-ref>my-cache</c:cache-ref>
  </c:cache>
 </assembly-descriptor>
</jboss:ejb-jar>
```

5.

タイムアウト値を設定する方法は、**EJB 2** または **EJB 3** を実装するかどうかによって異なります。

- **EJB 3** ではアノテーションが導入されるため、以下のように **EJB** コードで **javax.ejb.StatefulTimeout** アノテーションを指定できます。

```
@StatefulTimeout(value = 1320, unit=java.util.concurrent.TimeUnit.SECONDS)
@Stateful
@Remote(MyStatefulEJBRemote.class)
public class MyStatefulEJB implements MyStatefulEJBRemote {
  ...
}
```

@StatefulTimeout の値は以下のいずれかに設定できます。

- 値が **0** の場合は、**Bean** がすぐに削除できます。
- **0** より大きい値は、**unit** パラメーターで指定された単位のタイムアウト値を示します。デフォルトのタイムアウト単位は **MINUTES** です。パッシベーションキャッシュ設定を使用し、**idle-timeout** の値が **StatefulTimeout** 値よりも小さい場合、指定された **idle-timeout** 期間に対してアイドル状態になると、**JBoss EAP** は **Bean** をパッシベートします。その後、**Bean** は **StatefulTimeout** の期間後に削除することができます。
- **-1** を値と し て指定すると、タイムアウトにより **bean** が削除されません。パッシベーションキャッシュ設定を使用し、**Bean** が **idle-timeout** でアイドル状態である場合、**JBoss EAP** は **Bean** インスタンスを **passivation-store** に渡します。
- **-1** 未満の値は有効ではありません。

- *EJB 2* と *EJB 3* の両方に対して、*ejb-jar.xml* ファイルでステートフルタイムアウトを設定できます。

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
         version="3.1">
  <enterprise-beans>
    <session>
      <ejb-name>HelloBean</ejb-name>
        <session-type>Stateful</session-type>
        <stateful-timeout>
          <timeout>1320</timeout>
          <unit>Seconds</unit>
        </stateful-timeout>
    </session>
  </enterprise-beans>
</ejb-jar>
```

- *EJB 2* と *EJB 3* の両方に対して、*jboss-ejb3.xml* ファイルでステートフルタイムアウトを 設定できます。

```
<jboss:ejb-jar xmlns:jboss="http://www.jboss.com/xml/ns/javaee"
      xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:c="urn:ejb-cache:1.0"
      xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-ejb3-2_0.xsd http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/ejb-jar_3_1.xsd"
      version="3.1"
      impl-version="2.0">
  <enterprise-beans>
    <session>
      <ejb-name>HelloBean</ejb-name>
        <session-type>Stateful</session-type>
        <stateful-timeout>
          <timeout>1320</timeout>
          <unit>Seconds</unit>
        </stateful-timeout>
    </session>
  </enterprise-beans>
  <assembly-descriptor>
```

```
    <c:cache>
      <ejb-name>*</ejb-name>
      <c:cache-ref>my-cache</c:cache-ref>
    </c:cache>
  </assembly-descriptor>
</jboss:ejb-jar>
```

追加情報

- *ステートフルセッション **Bean** のパッシベーションを無効にするには、以下のいずれかを行います。*
  - ***EJB 3** アノテーションを使用してステートフルセッション **Bean** を実装した場合、アノテーションのステートフルセッション **Bean** のパッシベーションを無効にできます。**@org.jboss.ejb3.annotation.Cache("NoPassivationCache")***
  - *ステートフルセッション **Bean** が **jboss-ejb3.xml** ファイルで設定されている場合、**<c:cache-ref>** 要素の値を **NoPassivationCache** と同等である「**simple**」に設定します。*

    ```
    <c:cache-ref>simple</c:cache-ref>
    ```

- ***JBoss EAP 6** では **EJB** キャッシュポリシー「**LRUStatefulContextCachePolicy**」が変更になったため、**JBoss EAP 6** では **1** 対 **1** の設定マッピングを行うことができません。*
- ***JBoss EAP 6** では、以下のキャッシュプロパティーを設定できます。*
  - ***Bean** のライフサイクル時間は、**EJB 3.1** の **@StatefulTimeout** を使用して設定さ*

れます。

- **<file-passivation-store>** 要素の **idle-timeout** 属性を使用して、サーバー設定ファイルの **ejb3** サブシステムのディスクに **Bean** のパッシベーションを設定します。

- **<file-passivation-store>** 要素の **max-size** 属性を使用して、サーバー設定ファイルの **ejb3** サブシステムでパッシベーションストアの最大サイズを設定します。

- **JBoss EAP 6** では、以下のキャッシュプロパティーを設定することはできません。

  - メモリーキャッシュの最小および最大数。

  - パッシベーションストアの最小番号。

  - キャッシュ操作の頻度を制御する ***-period** 設定。

バグの報告

21.5. メッセージ駆動型 BEAN の設定

21.5.1. メッセージ駆動型 Bean のデフォルトリソースアダプターの設定

JBoss 管理者は、メッセージ駆動型 **Bean** によって使用されるデフォルトのリソースアダプターを指定できます。デフォルトのリソースアダプターは管理コンソールおよび **CLI** を使用して指定できます。**JBoss EAP 6** で提供されるデフォルトのリソースアダプターは **hornetq-ra** です。

手順21.19 管理コンソールを使用したメッセージ駆動型 Bean のデフォルトリソースアダプターの設定

1. 管理コンソールへログインします。「管理コンソールへのログイン」

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Container** タブを選択します。

3. **Edit** をクリックします。**Details** エリアのフィールドを編集できるようになりました。

4. **Default Resource Adapter** テキストボックスに使用するリソースアダプターの名前を入力します。

5. **Save** をクリックして終了します。

手順21.20 CLI を使用したメッセージ駆動型 Bean のデフォルトリソースアダプターの設定

1. **CLI** ツールを起動し、サーバーに接続します。「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」 を参照してください。

2. 次の構文で **write-attribute** 操作を使用します。

```
/subsystem=ejb3:write-attribute(name="default-resource-adapter-name",
value="RESOURCE-ADAPTER")
```

RESOURCE-ADAPTER は、使用されるリソースアダプターの名前に置き換えます。

3. **read-resource** 操作を使用して変更を確認します。

```
/subsystem=ejb3:read-resource
```

例21.15 CLI を使用したメッセージ駆動型 Bean のデフォルトリソースアダプターの設定

```
[standalone@localhost:9999 subsystem=ejb3] /subsystem=ejb3:write-attribute(name="default-
resource-adapter-name", value="EDIS-RA")
{"outcome" => "success"}
[standalone@localhost:9999 subsystem=ejb3]
```

例21.16 XML 設定の例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">
  <mdb>
    <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
    <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
  </mdb>
</subsystem>
```

バグの報告

21.6. EJB3 タイマーサービスの設定

21.6.1. EJB3 タイマーサービス

EJB3 タイマーサービスは、エンタープライズ Bean からメソッドの呼び出しをスケジュールする標準の Java EE 6 サービスです。ステートレスセッション Bean、シングルトンセッション Bean、およびメッセージ駆動型 Bean はすべて、指定された時間にコールバック用にメソッドをスケジュールすることができます。メソッドコールバックは、特定の時間、期間の後、繰り返しの間隔、またはカレンダーベースのスケジュールで発生する可能性があります。

バグの報告

21.6.2. EJB3 タイマーサービスの設定

EJB3 タイマーサービスは、管理コンソールまたは管理 CLI で設定できます。スケジュールされた Bean 呼び出しに使用されるスレッドプールと、Timer Service データの保存に使用されるディレクトリーまたはデータソースのいずれかを設定できます。デフォルトのディレクトリーよりも高速ストレージが利用可能な場合は、デフォルトの Timer Service ディレクトリーを変更することができます。

手順21.21 管理コンソールを使用した EJB3 タイマーサービススレッドプールの設定

前提条件

- EJB3 タイマーサービスによって使用されるスレッドプールが作成済みである必要があります。

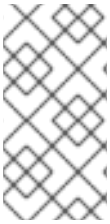
1. 管理コンソールへログインします。

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Services** タブを選択し、**Timer Service** をクリックします。**Edit** をクリックします。

3. **EJB3 Thread Pool** ドロップダウンリストをクリックし、優先スレッドプールの名前をクリックします。

4. **JBoss EAP** インスタンスを再起動します。

手順**21.22** 管理 **CLI** での **EJB3** タイマーサービススレッドプールの設定



注記

管理対象ドメインのコマンドに **/profile=PROFILE_NAME** というプレフィックスを追加します。

1. 以下の管理 **CLI** コマンドを実行します。

```
/subsystem=ejb3/service=timer-service:write-attribute(name=thread-pool-
name,value="thread-pool-name")
```

2. **JBoss EAP** インスタンスを再起動します。

手順**21.23** 管理コンソールを介した **EJB3** タイマーサービスディレクトリーの設定

1. 管理コンソールへログインします。

2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Services** タブを選択し、**Timer Service** をクリックします。**Edit** をクリックします。

3. *希望する値を **Path** フィールドおよび **Relative To** フィールドに入力します。*

4. ***Save** をクリックします。*

5. ***JBoss EAP** インスタンスを再起動します。*

*手順**21.24** 管理 **CLI** を介した **EJB3** タイマーサービスディレクトリーの設定*

1. *変更するパスに応じて、以下の管理 **CLI** コマンドの **1** つまたは両方を実行します。いずれのパスでもシステム値を使用できます（例： **${jboss.server.data.dir}** ）。*

注記

*管理対象ドメインのコマンドに **/profile=PROFILE_NAME** というプレフィックスを追加します。*

```
/subsystem=ejb3/service=timer-service/file-data-store=default-file-store:write-attribute(name=path,value="path")
```

```
/subsystem=ejb3/service=timer-service/file-data-store=default-file-store:write-attribute(name=relative-to,value="relative-path")
```

2. ***JBoss EAP** インスタンスを再起動します。*

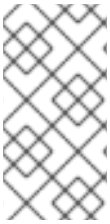
*手順**21.25** 管理 **CLI** よりデータソースを使用するよう **EJB3** タイマーサービスを設定*

***JBoss EAP 6.4** 以降、**EJB3** タイマーサービスをローカルディレクトリーの代わりにデータソースを使用するように設定できます。このオプションにはパフォーマンスコストが若干ありますが、ローカルストレージの問題の発生時にタイマーデータに対するリスクを減らすという利点があります。*

***EJB3** タイマーサービスがデータソースを使用するよう設定したら、データストアを使用するように **EJB** デプロイメントを設定するか、すべてのデプロイメントのデフォルトとして設定する必要があります。手順は、「データソースを『**使用するよう 1 つまたはすべての EJB3 デプロイメントを設定」の手順を参照してください**』。*

前提条件

- *EJB3 タイマーサービスによって使用されるデータソースがすでに存在し、基礎となるデータベースが READ_COMMITTED または SERIALIZABLE 分離モード用にサポートされ、設定する必要があります。*



注記

管理対象ドメインのコマンドに /profile=PROFILE_NAME というプレフィックスを追加します。

- *以下の管理 CLI コマンドを実行します。*

  - *datastore_name: 任意の名前。*
  - *DATASOURCE_NAME: ストレージに使用されるデータソースの名前。*
  - *データベース - postgresql、mssql、sybase、mysql、oracle、db2、または hsql のいずれか。*
  - *partition_name - 任意の名前。この属性は、複数の JBoss EAP インスタンスが EJB タイマーを保存するために同じデータベースを共有する場合に、特定のサーバーインスタンスに関連するタイマーを区別するために使用されます。この場合、すべてのサーバーインスタンスには独自のパーティション名が必要です。データベースが 1 つのサーバーインスタンスによってのみ使用される場合は、この属性を空白のままにすることができます。*

```
/subsystem=ejb3/service=timer-service/database-data-
store=datastore_name:add(datasource-jndi-name='java:/datasource_name',
database='database', partition='partition_name')
```

手順21.26 データソースを使用するように 1 つまたはすべての EJB3 デプロイメントを設定

EJB3 デプロイメントを、Timer サービスのデータソースを使用するように設定するか、すべて のデプロイメントのデフォルトとして設定します。

-
  - *EJB3 デプロイメントをデータソースを使用するように設定するには、デプロイメントの jboss-ejb3.xml を編集して、タイマー セクションが以下のようにします。datastore_name をデータストアの名前に置き換えます。*

```
[<assembly-descriptor>
  <timer:timer>
    <ejb-name>*</ejb-name>
    <timer:persistence-store-name>datastore_name</timer:persistence-store-
name>
  </timer:timer>
</assembly-descriptor>
```

  - *すべて のデプロイメントのデフォルトとしてデータソースを設定するには、以下の管理 CLI コマンドを実行してから JBoss EAP インスタンスを再起動します。datastore_name をデータストアの名前に置き換えます。*

注記

管理対象ドメインのコマンドに /profile=PROFILE_NAME というプレフィックスを追加します。

```
[/subsystem=ejb3/service=timer-service:write-attribute(name=default-data-
store,value=datastore_name)
```

バグの報告

21.7. EJB3 非同期呼び出しサービスの設定

21.7.1. EJB3 非同期呼び出しサービス

非同期呼び出しサービスは、セッション Bean メソッドの非同期呼び出しを管理する Enterprise JavaBeans コンテナーサービスです。このサービスは、非同期メソッドの実行に割り当てられる設定可能な数のスレッド（スレッドプール）を維持します。

Enterprise JavaBeans 3.1 では、セッション Bean (ステートフル、ステートレス、シングルトン) の任意のメソッドにアノテーションを付けて非同期実行を許可することができます。

バグの報告

21.7.2. EJB3 非同期呼び出しサービスのスレッドプールの設定

JBoss の管理者は、特定のスレッドプールを使用するように JBoss EAP 6 の管理コンソールに EJB3 非同期呼び出しサービスを設定することができます。

手順21.27 EJB3 非同期呼び出しサービスのスレッドプールの設定

1. 管理コンソールへログインします。「管理コンソールへのログイン」を参照してください。
2. 画面上部の Configuration タブをクリックします。Container メニューを展開し、EJB 3 を選択します。Services タブを選択し、Async Service をクリックします。
3. Edit をクリックします。
4. リストから使用する EJB3 スレッドプールを選択します。スレッドプールがすでに作成されている必要があります。
5. Save をクリックして終了します。

バグの報告

21.8. EJB3 リモート呼び出しサービスの設定

21.8.1. EJB3 リモートサービス

EJB3 リモートサービスは、リモートビジネスインターフェースでエンタープライズ **Bean** のリモート実行を管理します。

バグの報告

21.8.2. EJB3 リモートサービスの設定

JBoss 管理者は、**JBoss EAP 6** の管理コンソールで **EJB3** リモートサービスを設定できます。設定可能な機能は、リモート **Bean** 呼び出しに使用されるスレッドプールと、**EJB3** リモーティングチャネルが登録されるコネクターです。

手順21.28 EJB3 リモートサービスの設定

1. 管理コンソールへログインします。「管理コンソールへのログイン」を参照してください。
2. 画面上部の **Configuration** タブをクリックします。**Container** メニューを展開し、**EJB 3** を選択します。**Services** タブを選択し、**Remote Service** をクリックします。
3. **Edit** をクリックします。
4. 追加のスレッドプールが設定されている場合は、**Remote Service** に使用される別の **EJB3** スレッドプールを選択できます。**EJB** リモーティングチャネルの登録に使用されるコネクターを変更できます。
5. **Save** をクリックして終了します。

バグの報告

21.9. EJB 2.X エンティティー BEAN の設定

21.9.1. EJB エンティティー Bean

EJB エンティティー Bean は、データベースで保持された永続データを表す EJB 仕様のバージョン 2.x からのエンタープライズ bean のタイプです。エンティティー Bean は JPA エンティティーによって置き換えられ、今後の仕様バージョンから削除(pruning)するために正式にリストされています。Red Hat は、後方互換性を除いて、エンティティー Bean の使用を推奨していません。

JBoss EAP 6 ではエンティティー Bean のサポートはデフォルトで無効になっています。

注記

JBoss EAP 6.x は EJB 2.0 以降をサポートしますが、EJB 2.0 デプロイメント記述子は JBoss EAP 6.4 以上でのみ動作します。

バグの報告

21.9.2. コンテナ管理による永続性

コンテナ管理による永続性 (CMP) は、エンティティー Bean のデータ永続性を提供する、アプリケーションサーバーが提供するサービスです。

バグの報告

21.9.3. EJB 2.x のコンテナ管理による永続性の有効化

コンテナー管理による永続性(CMP)は org.jboss.as.cmp 拡張によって処理されます。CMP は、管理対象ドメインおよびスタンドアロンサーバーの完全な設定（例： standalone-full.xml ）でデフォルトで有効になっています。

別の設定で CMP を有効にするには、org.jboss.as.cmp モジュールをサーバー設定ファイルの有効な拡張の一覧に追加します。

```
<extensions>
    <extension module="org.jboss.as.cmp"/>
</extensions>
```

エクステンションを追加したら、<profile> 要素の下にあるプロファイルの XML 設定ファイルに以下の要素を追加する必要があります。

```
<subsystem xmlns="urn:jboss:domain:cmp:1.1"/>
```

サーバー設定で **CMP** を無効にするには、**org.jboss.as.cmp** モジュールの拡張エントリーを削除します。

バグの報告

21.9.4. EJB 2.x のコンテナ管理による永続性の設定

EJB 2.x Container Managed Persistence(CMP)サブシステムを設定して、任意の数のキージェネレーターを指定できます。キージェネレーターは、**CMP** サービスが永続化する各エンティティーを識別する一意の鍵を生成するために使用されます。

UUID ベースのキージェネレーターと **HiLo** キージェネレーターの **2** つの種類のキージェネレーターを定義できます。

UUID ベースのキージェネレーター

UUID ベースのキージェネレーターは、**Universally Unique Identifier** を使用してキーを作成します。**UUID** キージェネレーターは、一意の名前のみを指定するだけで、他の設定はありません。

UUID ベースのキージェネレーターは、以下のコマンド構文を使用して **CLI** で追加できます。

```
/subsystem=cmp/uuid-keygenerator=UNIQUE_NAME:add
```

例**21.17 UUID** キージェネレーターの追加

uuid_identities の名前で **UUID** ベースのキージェネレーターを追加するには、以下の **CLI** コマンドを使用します。

```
/subsystem=cmp/uuid-keygenerator=uuid_identities:add
```

このコマンドで作成される **XML** 設定は以下のとおりです。

```
<subsystem xmlns="urn:jboss:domain:cmp:1.0">
  <key-generators>
    <uuid name="uuid_identities" />
  </key-generators>
</subsystem>
```

HiLo キージェネレーター

HiLo キージェネレーターは、データベースを使用してエンティティー **ID** キーを作成および保存します。**HiLo** キージェネレーターは一意の名前を持ち、データの保存に使用するデータソースを指定するプロパティーと、キーを格納するテーブルおよび列の名前を設定する必要があります。

HiLo キージェネレーターは、以下のコマンド構文を使用して **CLI** で追加できます。

```
/subsystem=cmp/hilo-keygenerator=UNIQUE_NAME/:add(property=value, property=value, ...)
```

例**21.18 HiLo** キージェネレーターの追加

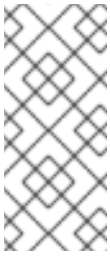
```
/subsystem=cmp/hilo-keygenerator=HiLoKeyGeneratorFactory:add(create-table=true,create-table-ddl="create table HILOSEQUENCES (SEQUENCENAME varchar(50) not null, HIGHVALUES integer not null, constraint hilo_pk primary key (SEQUENCENAME))",data-source=java:jboss/datasources/ExampleDS, id-column=HIGHVALUES,sequence-column=SEQUENCENAME,table-name=HILOSEQUENCES,sequence-name=general,block-size=10)
```

このコマンドで作成される **XML** 設定は以下のとおりです。

```
<subsystem xmlns="urn:jboss:domain:cmp:1.1">
    <key-generators>
        <hilo name="HiLoKeyGeneratorFactory">
            <block-size>10</block-size>
            <create-table>true</create-table>
            <create-table-ddl>create table HILOSEQUENCES (SEQUENCENAME varchar(50) not null, HIGHVALUES integer not null, constraint hilo_pk primary key (SEQUENCENAME))</create-table-ddl>
            <data-source>java:jboss/datasources/ExampleDS</data-source>
            <id-column>HIGHVALUES</id-column>
            <sequence-column>SEQUENCENAME</sequence-column>
            <sequence-name>general</sequence-name>
            <table-name>HILOSEQUENCES</table-name>
        </hilo>
    </key-generators>
</subsystem>
```

注記

***block-size** を **!=0** の値に設定しないと、生成された **PKey** がインクリメントされないため、**DuplicateKeyException** が発生し、エンティティーの作成に失敗します。*



注記

*一貫性を確保するために、クラスターの場合は **select-hi-ddl** を **'FOR UPDATE'** として設定する必要があります。すべてのデータベースはロック機能をサポートしません。*

バグの報告

21.9.5. HiLo キージェネレーター用の CMP サブシステムプロパティー

表21.1 HiLo キージェネレーター用の CMP サブシステムプロパティー

プロパティー	データ型	説明
block-size	Long	ブロックサイズ。
create-table	boolean	**TRUE** に設定されている場合、そのテーブルが見つからない場合には、**table-name** というテーブルが **create-table-ddl** の内容を使用して作成されます。
create-table-ddl	string	テーブルが見つからない場合に、**table-name** に指定されたテーブルを作成し、**create-table** が **TRUE** に設定されている場合に、指定したテーブルを作成するために使用される DDL コマンド。
data-source	token	データベースへの接続に使用するデータソース。
drop-table	boolean	テーブルをドロップするかどうかを決定します。
id-column	token	ID カラム名。
select-hi-ddl	string	現在保管されている中で最も大きなキーを返す SQL コマンド。

プロパティー	データ型	説明
sequence-column	token	シーケンスカラム名。
sequence-name	token	シーケンスの名前。
table-name	token	キー情報の格納に使用されるテーブルの名前。

バグの報告

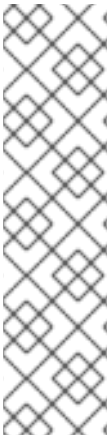
第22章 JAVA CONNECTOR ARCHITECTURE (JCA)

22.1. はじめに

22.1.1. Java EE Connector API (JCA)

JBoss EAP 6 は、Java EE Connector API(JCA)1.6 仕様の完全サポートを提供します。JCA 仕様の詳細は、「JSR 322: Java EE Connector Architecture 1.6」を参照してください。

リソースアダプターは Java EE Connector API アーキテクチャーを実装するコンポーネントです。ただし、データソースオブジェクトと似ていますが、エンタープライズ情報システム(EIS)から、データベース、メッセージングシステム、トランザクション処理、エンタープライズリソースプランニング(enterprise Resource Planning)システムなどの幅広い異種のシステムへの接続を提供します。



注記

Java Platform Enterprise Edition 6 には javax.resource.cci パッケージが同梱されています。javax.resource.cci パッケージは、Common Client Interface (CCI)をサポートするリソースアダプターによって実装する必要のある API で構成されます。javax.resource.cci.ResultSet このパッケージのメンバーで、java.sql.ResultSet を拡張します。java.sql.ResultSet のインターフェースは使用される Java バージョンに依存するため、Common Client Interface (CCI)を使用する場合は、すべてのアプリケーションはデータの対話に Java 6 の java.sql.ResultSet メソッドのみを使用できることを想定する必要があります。

バグの報告

22.1.2. Java Connector Architecture (JCA)

Java EE Connector Architecture (JCA) は外部にある異種のエンタープライズ情報システム (EIS) に対して Java EE システムの標準アーキテクチャーを定義します。EIS の例として、エンタープライズリソースプランニング(Warehouse)システム、商用トランザクション処理(TP)、データベース、メッセージングシステムなどが挙げられます。

JCA 1.6 は以下の管理機能を提供します。

- connections

- *transactions*
- *security*
- *life-cycle*
- *work instances*
- *transaction inflow*
- *message inflow*

JCA 1.6 は Java Community Process で JSR-322 として開発されました http://jcp.org/en/jsr/detail?id=313。

代替の管理接続プール

JBoss EAP 6.4 は以下の代替プール実装を特長としています。

- **org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreArrayListManagedConnectionPool:** *これはデフォルトの接続プールです。*
- **org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool:** *この接続プールはより優れたパフォーマンスプロファイルを提供し、システムプロパティー* ***-Dironjacamar.mcp=org.jboss.jca.core.connectionmanager.pool.mcp.SemaphoreConcurrentLinkedQueueManagedConnectionPool*** *を使用して有効にされることがありました。*
- **org.jboss.jca.core.connectionmanager.pool.mcp.LeakDumperManagedConnectionPool:** *この接続プールはデバッグ目的でのみ使用されます。* ***LeakDetectorPool*** *の詳細は、を参照してください。* ***http://www.ironjacamar.org/doc/userguide/1.2/en-US/html/ch04.html#configuration_ironjacamar_leakpool***

バグの報告

22.1.3. リソースアダプター

リソースアダプターは、 **Java Connector Architecture (JCA)** 仕様を使用して **Java EE** アプリケーションとエンタープライズ情報システム **(EIS)** との間の通信を提供するデプロイ可能な **Java EE** コンポーネントです。**EIS** ベンダーの製品と **Java EE** アプリケーションの統合を容易にするため、リソースアダプターは通常 **EIS** ベンダーによって提供されます。

エンタープライズ情報システムは、組織内における他のあらゆるソフトウェアシステムのことです。例としては、エンタープライズリソースプランニング **(ERP)** システム、データベースシステム、電子メールサーバー、商用メッセージングシステムなどが挙げられます。

リソースアダプターは、**JBoss EAP 6** にデプロイできる **Resource Adapter Archive(RAR)**ファイルでパッケージ化されます。また、**RAR** ファイルは、**Enterprise Archive (EAR)** デプロイメントにも含めることができます。

バグの報告

22.2. JAVA CONNECTOR ARCHITECTURE (JCA) サブシステムの設定

JBoss EAP 6 設定ファイルの **JCA** サブシステムは、**JCA** コンテナおよびリソースアダプターデプロイメントの一般的な設定を制御します。

JCA サブシステムの主な要素

アーカイブの検証

- この設定はデプロイメントユニット上でアーカイブの検証が実行されるかどうかを決定します。
- アーカイブの検証に設定できる属性は下表のとおりです。

表**22.1** アーカイブ検証の属性

属性	デフォルト値	説明
enabled	true	アーカイブバリデーションが有効であるかどうかを指定します。
fail-on-error	true	アーカイブバリデーションのエラーレポートによってデプロイメントが失敗するかどうかを指定します
fail-on-warn	false	アーカイブバリデーションの警告レポートによってデプロイメントが失敗するかどうかを指定します。

- *アーカイブが **Java EE Connector Architecture** 仕様を正しく実装せず、アーカイブ検証が有効になっている場合、デプロイメント中に問題を説明するエラーメッセージが表示されます。以下に例を示します。*

```
Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public int hashCode()"
method.
Code: com.mycompany.myproject.ResourceAdapterImpl

Severity: ERROR
Section: 19.4.2
Description: A ResourceAdapter must implement a "public boolean equals(Object)"
method.
Code: com.mycompany.myproject.ResourceAdapterImpl
```

- *アーカイブバリデーションが指定されていない場合は、アーカイブ検証が指定されているとみなされ、**enabled** 属性のデフォルトが **true** に設定されます。*

***Bean** バリデーション*

- *この設定はデプロイメントユニット上で **Bean** の検証 **(JSR-303)** が実行されるかどうかを決定します。*

- ***Bean** の検証に設定できる属性は下表のとおりです。*

*表**22.2 Bean** 検証の属性*

属性	デフォルト値	説明
enabled	true	Bean バリデーションが有効であるかどうかを指定します。

- *__Bean__ バリデーションが指定されていない場合は、__Bean__ バリデーションが指定されているとみなされ、__enabled__ 属性のデフォルトが __true__ に設定されます。*

ワークマネージャー

- *ワークマネージャーには次の __2__ 種類があります。*

  デフォルトワークマネージャー

  デフォルトのワークマネージャーおよびそのスレッドプール。

  カスタムワークマネージャー

  カスタムワークマネージャーの定義およびそのスレッドプール。

- *ワークマネージャーに設定できる属性は下表のとおりです。*

表__22.3__ ワークマネージャーの属性

属性	説明
name	ワークマネージャーの名前を指定します。これはカスタムワークマネージャーに必要です。
short-running-threads	標準の Work インスタンスのスレッドプール。ワークマネージャーごとに短時間実行されるスレッドプールが 1 つあります。
long-running-threads	**LONG_RUNNING** ヒントを設定する JCA 1.6 Work インスタンスのスレッドプール。各ワークマネージャーはオプションの長期スレッドプールを 1 つ持てます。

- *ワークマネージャーのスレッドプールに設定できる属性は下表のとおりです。*

表22.4 スレッドプールの属性

属性	説明
allow-core-timeout	コアスレッドがタイムアウトするかどうかを決定するブール値の設定。デフォルト値は false です。
core-threads	コアスレッドプールのサイズ。スレッドプールの最大サイズ以下である必要があります。
queue-length	キューの最大長。
max-thread	スレッドプールの最大サイズ。
keepalive-time	ワーク実行後にスレッドプールが保持される期間を指定します。
thread-factory	スレッドファクトリーへの参照。

ブートストラップコンテキスト

- *カスタムのブートストラップコンテキストを定義するために使用されます。*

- *ブートストラップコンテキストに設定できる属性は下表のとおりです。*

表22.5 ブートストラップコンテキストの属性

属性	説明
name	ブートストラップコンテキストの名前を指定します。
workmanager	このコンテキストに使用するワークマネージャーの名前を指定します。

キャッシュ済み接続マネージャー

- *接続のデバッグ、およびトランザクションにおける接続の **lazy enlistment** のサポートに使用されます。また、接続がアプリケーションによって適切に使用およびリリースされるかどうかを追跡します。*

- *キャッシュ済みの接続マネージャーに設定できる属性は下表のとおりです。*

表22.6 キャッシュ済み接続マネージャーの属性

属性	デフォルト値	説明
debug	false	接続を明示的に閉じるため、障害時に警告を出力します。
error	false	接続を明示的に閉じるため、障害時に例外が発生します。

手順22.1 管理コンソールを使用した JCA サブシステムの設定

JBoss EAP 6 の JCA サブシステムは管理コンソールで設定できます。JCA 設定オプションは、サーバーの実行方法に応じて、管理コンソールで若干異なる場所に置かれます。

1. *画面上部の Configuration タブをクリックします。Connector メニューを展開し、JCA を選択します。*

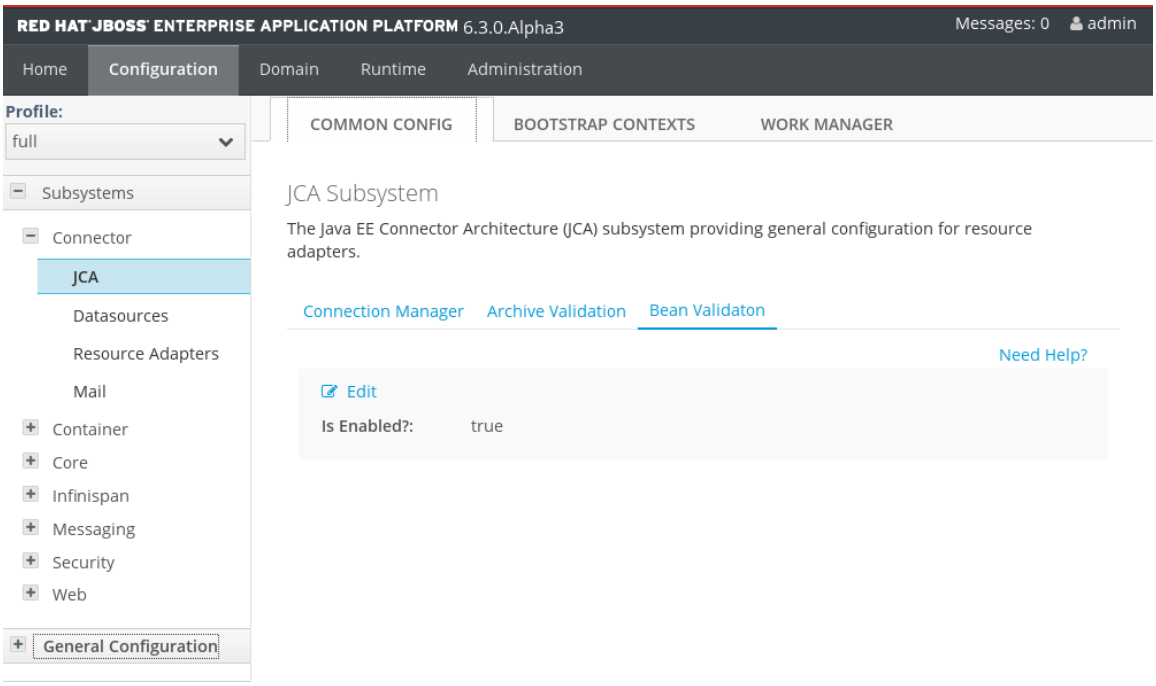
2. *サーバーがドメインモードで稼働している場合は、左上の Profile ドロップダウンメニューからプロファイルを選択します。*

3. *3 つのタブを使用して JCA サブシステムを設定します。*

    a. *共通設定*

    Common Config タブには、キャッシュされた接続マネージャー、アーカイブ検証、および Bean 検証(JSR-303)の設定が含まれます。これらの項目は独自のタブに含まれます。これらの設定は、タブを開き、編集ボタンをクリックして必要な変更を行い、保存ボタンをクリックします。

図22.1 JCA 共通設定

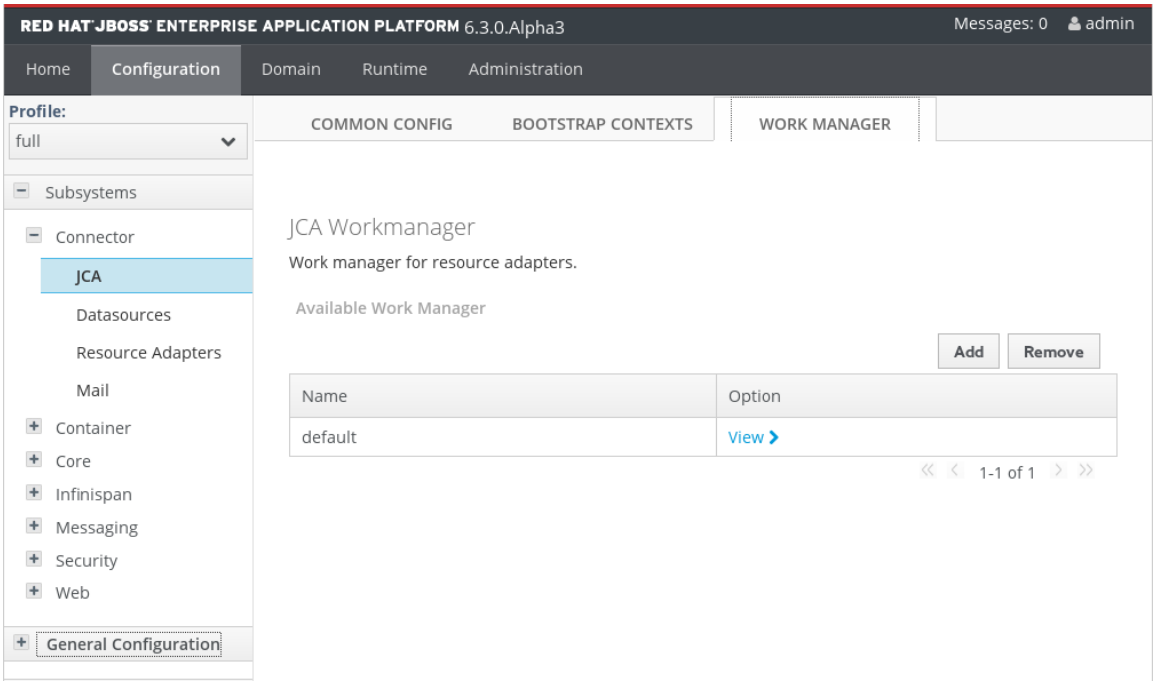


[D]

b. *ワークマネージャー*

***Work Manager** タブには、設定したワークマネージャーのリストが含まれます。新しいワークマネージャーを追加および削除でき、スレッドプールをここで設定できます。各 **Work Manager** は短時間実行されるスレッドプールを **1** つ持つことができ、任意で長時間実行されるスレッドプールを **1** つ持つことができます。*

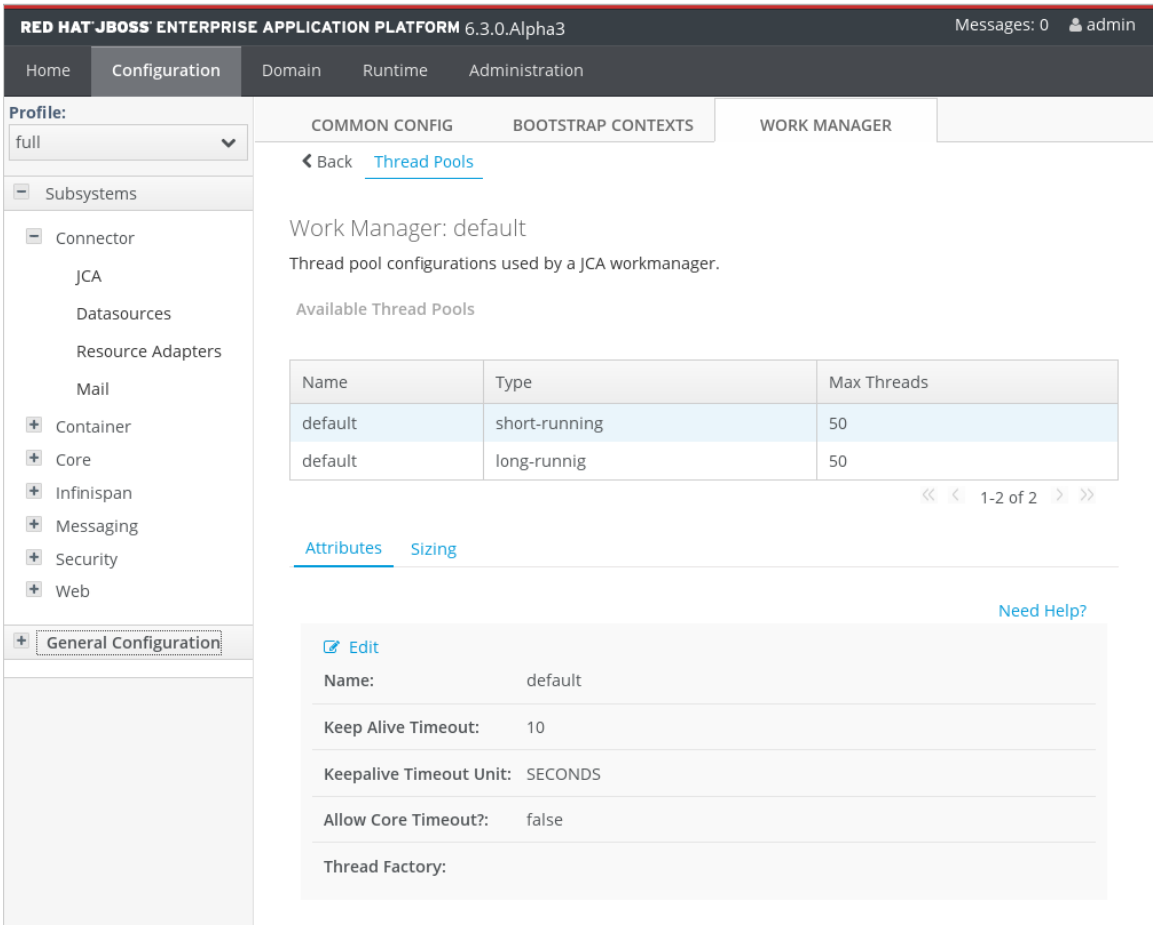
図22.2 ワークマネージャー



[D]

スレッドプール属性を設定するには、選択したリソースアダプターの 表示 をクリックします。

図22.3 ワークマネージャースレッドプール

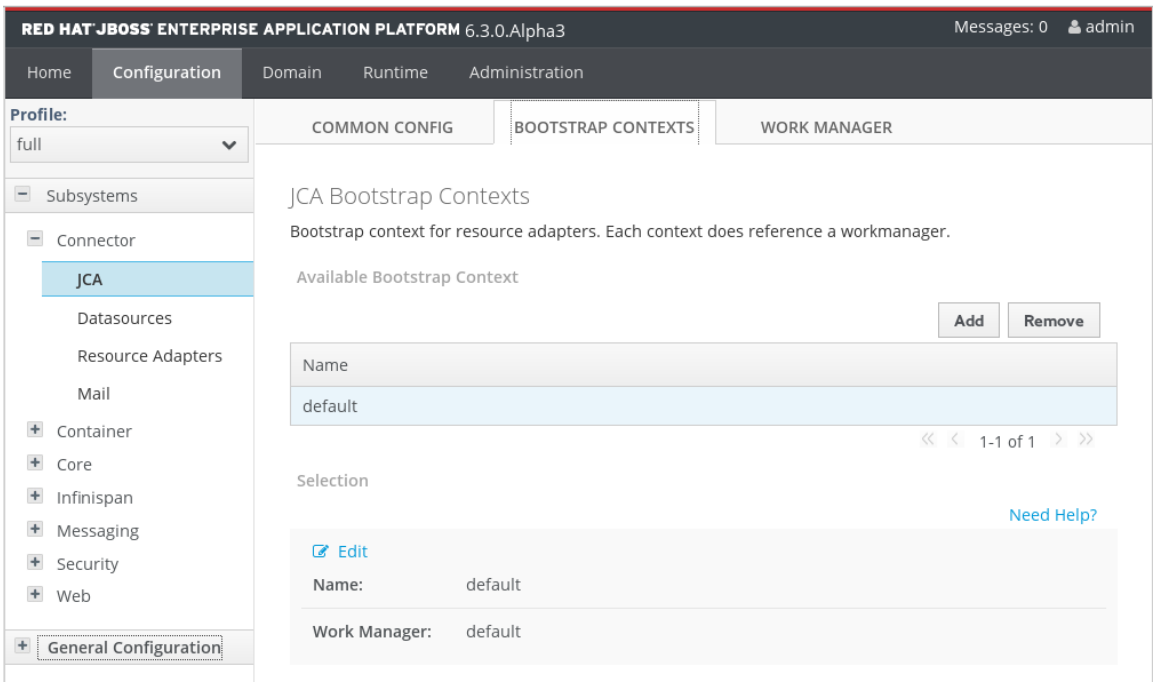


[D]

c. *ブートストラップコンテキスト*

*ブートストラップコンテキスト タブに は、設定されたブートストラップコンテキストのリストが含まれます。新しいブートストラップコンテキストオブジェクトを追加、削除、および設定できます。各ブートストラップコンテキストに **Work Manager** を割り当てる必要があります。*

図22.4 ブートストラップコンテキスト



[D]

バグの報告

22.3. リソースアダプターのデプロイ

*リソースアダプターは、管理 **CLI** ツールまたは **Web** ベースの管理コンソールを使用して **JBoss EAP 6** にデプロイするか、ファイルを手作業でコピーしてデプロイできます。このプロセスは、他のデプロイ可能なアーティファクトと同じです。*

*手順22.2 管理 **CLI** を使用したリソースアダプターのデプロイ*

1. *オペレーティングシステムのコマンドプロンプトを開きます。*
2. *管理 **CLI** へ接続します。*
   - ***Linux** の場合は、コマンドラインで以下を入力します。*

```
$ EAP_HOME/bin/jboss-cli.sh --connect
$ Connected to standalone controller at localhost:9999
```

- ***Windows*** *の場合は、コマンドラインで以下を入力します。*

```
C:\>EAP_HOME\bin\jboss-cli.bat --connect
C:\> Connected to standalone controller at localhost:9999
```

3. *リソースアダプターをデプロイします。*

- *リソースアダプターをスタンドアロンサーバーへデプロイするには、次のコマンドラインを入力します。*

```
$ deploy path/to/resource-adapter-name.rar
```

- *リソースアダプターを管理対象ドメインのすべてのサーバーグループにデプロイするには、次のコマンドラインを入力します。*

```
$ deploy path/to/resource-adapter-name.rar --all-server-groups
```

*手順**22.3** 管理コンソールを使用したリソースアダプターのデプロイ*

1. *管理コンソールへログインします。「管理コンソールへのログイン」を参照してください。*

2. *画面上部の **Runtime** タブをクリックします。**Manage Deployments** を選択し、**Add** をクリックします。*

3. *リソースアダプターアーカイブを閲覧して選択します。次に、**Next** をクリックします。*

4. *デプロイメント名を確認してから、**Save** をクリックします。*

5. *この時点で、リソースアダプターアーカイブが無効の状態でリストに表示されるはずです。*

6. *リソースアダプターを有効にします。*

    - *ドメインモードでは、割り当て をクリックします。リソースアダプターを割り当てるサーバーグループを選択します。**Save** をクリックして終了します。*

    - *スタンドアロンモードでは、一覧から **Application Component** を選択します。**En/Disable** をクリックします。**Are You Sure?** ダイアログで 確認 をクリックし、コンポーネントを有効にします。*

*手順**22.4** 手作業によるリソースアダプターのデプロイ*

- *リソースアダプターアーカイブをサーバーデプロイメントディレクトリーへコピーします。*

    - *スタンドアロンサーバーの場合は、リソースアダプターアーカイブを **EAP_HOME/standalone/deployments/** ディレクトリーにコピーします。*

    - *管理対象ドメインでは、管理コンソールまたは管理 **CLI** を使用してリソースアダプターアーカイブをサーバーグループにデプロイする必要があります。*

バグの報告

22.4. デプロイされたリソースアダプターの設定

JBoss 管理者は、管理 CLI ツールまたは Web ベースの管理コンソールを使用して JBoss EAP 6 のリソースアダプターを設定できます。また、設定ファイルを手作業で編集して設定することも可能です。

サポートされるプロパティーと他の詳細については、リソースアダプターのベンダードキュメントを参照してください。

注記

以下の手順では、**[standalone@localhost:9999 /]** プロンプトに従ってコマンドラインを入力する必要があります。テキストを中括弧内に入力しないでください。これは、**{"outcome" => "success"}** のようにコマンドの結果として表示さ れる出力です。

手順22.5 管理 CLI を使用したリソースアダプターの設定

1. オペレーティングシステムのコマンドプロンプトを開きます。
2. 管理 CLI へ接続します。
   - Linux の場合は、コマンドラインで以下を入力します。

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

次のような出力が表示されるはずです。

```
$ Connected to standalone controller at localhost:9999
```

- ***Windows*** *の場合は、コマンドラインで以下を入力します。*

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

次のような出力が表示されるはずです。

```
C:\> Connected to standalone controller at localhost:9999
```

3. *リソースアダプター設定を追加します。*

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar:add(archive=eis.rar, transaction-support=XATransaction)
{"outcome" => "success"}
```

4. *サーバー リソースアダプターレベル* ***<config-property>*** *を設定します。*

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/config-properties=server/:add(value=localhost)
{"outcome" => "success"}
```

5. *ポート リソースアダプターレベル* ***<config-property>*** *を設定します。*

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/config-properties=port/:add(value=9000)
{"outcome" => "success"}
```

6. *管理接続ファクトリーの接続定義を追加します。*

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-
adapter=eis.rar/connection-definitions=cfName:add(class-
name=com.acme.eis.ra.EISManagedConnectionFactory, jndi-
name=java:/eis/AcmeConnectionFactory)
{"outcome" => "success"}
```

7. 管理接続ファクトリーレベルの **<config-property>** を設定します。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/connection-definitions=cfName/config-properties=name/:add(value=Acme Inc)
{"outcome" => "success"}
```

8. 管理オブジェクトを追加します。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName:add(class-name=com.acme.eis.ra.EISAdminObjectImpl, jndi-name=java:/eis/AcmeAdminObject)
{"outcome" => "success"}
```

9. **threshold** 管理オブジェクトプロパティーを設定します。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar/admin-objects=aoName/config-properties=threshold/:add(value=10)
{"outcome" => "success"}
```

10. リソースアダプターをアクティベートします。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar:activate
{"outcome" => "success"}
```

11. 新しく設定されアクティベートされたリソースアダプターを表示します。

```
[standalone@localhost:9999 /] /subsystem=resource-adapters/resource-adapter=eis.rar:read-resource(recursive=true)
{
    "outcome" => "success",
    "result" => {
        "archive" => "eis.rar",
        "beanvalidationgroups" => undefined,
        "bootstrap-context" => undefined,
        "transaction-support" => "XATransaction",
        "admin-objects" => {"aoName" => {
            "class-name" => "com.acme.eis.ra.EISAdminObjectImpl",
            "enabled" => true,
            "jndi-name" => "java:/eis/AcmeAdminObject",
            "use-java-context" => true,
            "config-properties" => {"threshold" => {"value" => 10}}
        }},
```

```
        "config-properties" => {
            "server" => {"value" => "localhost"},
            "port" => {"value" => 9000}
        },
        "connection-definitions" => {"cfName" => {
            "allocation-retry" => undefined,
            "allocation-retry-wait-millis" => undefined,
            "background-validation" => false,
            "background-validation-millis" => undefined,
            "blocking-timeout-wait-millis" => undefined,
            "class-name" => "com.acme.eis.ra.EISManagedConnectionFactory",
            "enabled" => true,
            "flush-strategy" => "FailingConnectionOnly",
            "idle-timeout-minutes" => undefined,
            "interleaving" => false,
            "jndi-name" => "java:/eis/AcmeConnectionFactory",
            "max-pool-size" => 20,
            "min-pool-size" => 0,
            "no-recovery" => undefined,
            "no-tx-separate-pool" => false,
            "pad-xid" => false,
            "pool-prefill" => false,
            "pool-use-strict-min" => false,
            "recovery-password" => undefined,
            "recovery-plugin-class-name" => undefined,
            "recovery-plugin-properties" => undefined,
            "recovery-security-domain" => undefined,
            "recovery-username" => undefined,
            "same-rm-override" => undefined,
            "security-application" => undefined,
            "security-domain" => undefined,
            "security-domain-and-application" => undefined,
            "use-ccm" => true,
            "use-fast-fail" => false,
            "use-java-context" => true,
            "use-try-lock" => undefined,
            "wrap-xa-resource" => true,
            "xa-resource-timeout" => undefined,
            "config-properties" => {"name" => {"value" => "Acme Inc"}}
        }}
    }
}
```

*手順**22.6 Web** ベースの管理コンソールを使用したリソースアダプターの設定*

1. *管理コンソールへログインします。「管理コンソールへのログイン」を参照してください。*

2. *画面上部の **Configuration** タブをクリックします。**Connectors** メニューを展開し、**Resource Adapters** を選択します。*

a.

ドメインモードでは、左上のドロップダウンから プロファイル を選択します。

***Add** をクリックします。*

3.

*アーカイブ名を入力し、**TX:** ドロップダウンメニューからトランザクションタイプ **XATransaction** を選択します。次に、**Save** をクリックします。*

4.

***Properties** タブを選択します。**Add** をクリックします。*

5.

*値 に **server** を、**Name** およびホスト名（例： **localhost** ）を入力し ます。次に、**Save** をクリックして終了します。*

6.

*もう一度 追加 をクリックします。値 の **name** および ポート 番号（例： **9000** ）に **port** を入力し ます。次に、**Save** をクリックして終了します。*

7.

*サーバー および ポート プロパティー が **Properties** パネルに表示されるようになりました。一覧表示されたリソースアダプターの **Option** 列にある **View** リンクをクリックして、**Connection Definitions** を表示します。*

8.

*利用可能な接続 定義の表の上にある **Add** をクリックして、接続定義を追加します。*

9.

***JNDI Name** および **Connection Class** の完全修飾クラス名を入力します。次に、**Save** をクリックして終了します。*

10.

*新しい **Connection Definition** を選択し、**Properties** タブを選択します。**Add** をクリックして、このコネクション定義の **Key** および **Value** データを入力します。**Save** をクリックして終了します。*

11.

*接続定義が完了しましたが、無効になっています。接続定義を選択し、**Enable** をクリックして接続定義を有効にします。*

12.

***JNDI** 名の場合は、実際のダイアログで接続定義を変更しますか？" **for the JNDI name.Confirm** をクリックします。接続定義が **Enabled** と表示されるはずです。*

13.
*ページ上部の **Admin Objects** タブをクリックして、管理オブジェクトを作成および設定します。次に、**Add** をクリックします。*

14.
*管理オブジェクトの **JNDI Name** および完全修飾 **Class Name** を入力します。次に、**Save** をクリックします。*

15.
***Properties** タブを選択してから **Add** をクリックして管理オブジェクトプロパティーを追加します。*

16.
***Name** フィールドに しきい値 などの管理オブジェクト設定プロパティーを 入力 します。**Value** フィールドに設定プロパティーの値を入力します（例： **10** ）。次に、**Save** をクリックしてプロパティーを保存します。*

17.
***admin** オブジェクトが完了し、無効になっています。**Enable** をクリックして **admin** オブジェクトを有効にします。*

18.
***JNDI** 名の 「**Admin Ojbect?** 」というダイアログが出されます。**Confirm** をクリックします。**admin** オブジェクトは **Enabled** と表示されるはずです。*

19.
*このプロセスを完了するには、サーバー設定をリロードする必要があります。**Runtime** タブをクリックします。**Server** メニューを展開します。左側のナビゲーションパネルで **Overview** を選択します。*

a.
サーバーをリロードします。

- *ドメインモードでは、サーバーグループにマウスを合わせます。**Restart Group** を選択します。*

- *スタンドアロンモードでは、リロード ボタンを 使用できます。**Reload** をクリックします。*

20. 指定のサーバーに対して、**Do you want to reload the server configuration?** というダイアログが表示されます。**Confirm** をクリックします。サーバー設定が最新の状態である。

手順**22.7** 手作業によるリソースサーバーの設定

1. **JBoss EAP 6** サーバーを停止します。

重要

サーバーの再起動後に変更が維持されるようにするには、サーバーを停止してからサーバー設定ファイルを編集する必要があります。

2. 編集するため、サーバー設定ファイルを開きます。

   - スタンドアロンサーバーの場合、これは **EAP_HOME/standalone/configuration/standalone.xml** ファイルです。

   - 管理対象ドメインの場合、これは **EAP_HOME/domain/configuration/domain.xml** ファイルです。

3. 設定ファイルで **urn:jboss:domain:resource-adapters** サブシステムを見つけます。

4. このサブシステムに対して定義されているリソースアダプターがない場合、最初に以下を置き換えます。

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

上記を以下のように置き換えます。

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
   <resource-adapters>
      <!-- <resource-adapter> configuration listed below -->
   </resource-adapters>
</subsystem>
```

5. **<!-- <resource-adapter> configuration listed below -->** をリソースアダプターの **XML** 定義に置き換えます。以下は、上記の管理 **CLI** および **Web** ベースの管理コンソールを使用して作成されたリソースアダプター設定の **XML** 表現です。

```
<resource-adapter id="NAME">
   <archive>
      eis.rar
   </archive>
   <transaction-support>XATransaction</transaction-support>
   <config-property name="server">
      localhost
   </config-property>
   <config-property name="port">
      9000
   </config-property>
   <connection-definitions>
      <connection-definition class-name="com.acme.eis.ra.EISManagedConnectionFactory"
            jndi-name="java:/eis/AcmeConnectionFactory"
            pool-name="java:/eis/AcmeConnectionFactory">
         <config-property name="name">
            Acme Inc
         </config-property>
      </connection-definition>
   </connection-definitions>
   <admin-objects>
      <admin-object class-name="com.acme.eis.ra.EISAdminObjectImpl"
            jndi-name="java:/eis/AcmeAdminObject"
            pool-name="java:/eis/AcmeAdminObject">
         <config-property name="threshold">
            10
         </config-property>
      </admin-object>
   </admin-objects>
</resource-adapter>
```

6. *サーバーの起動*

   *新しい設定で実行されるよう **JBoss EAP 6** サーバーを再起動します。*

バグの報告

22.5. リソースアダプター記述子リファレンス

以下の表では、リソースアダプター記述子要素について説明しています。

表22.7 主要な要素

要素	説明
bean-validation-groups	使用する必要がある Bean 検証グループを指定します。
bootstrap-context	使用する必要があるブートストラップコンテキストの一意な名前を指定します。
config-property	config-property は、リソースアダプター設定プロパティーを指定します。
transaction-support	このリソースアダプターがサポートするトランザクションのタイプを定義します。有効な値は **NoTransaction**、**LocalTransaction**、**XATransaction**です。
connection-definitions	接続定義を指定します。
admin-objects	管理オブジェクトを指定します。

表22.8 Bean 有効グループ要素

要素	説明
bean-validation-group	検証に使用する Bean 検証グループの完全修飾クラス名を指定します。

表22.9 接続定義/管理オブジェクト属性

属性	説明
class-name	管理対象接続ファクトリーまたは管理オブジェクトの完全修飾クラス名を指定します。
jndi-name	JNDI 名を指定します。
enabled	オブジェクトをアクティベートするかどうか。
use-java-context	**java:/** JNDI コンテキストを使用するかどうかを指定します。
pool-name	オブジェクトのプール名を指定します。
use-ccm	キャッシュ済み接続マネージャーを有効にします。

表22.10 接続定義要素

要素	説明
config-property	config-property は、管理対象接続ファクトリー設定プロパティーを指定します。
pool	プール設定を指定します。
xa-pool	XA プール設定を指定します。
security	セキュリティー設定を指定します。
timeout	タイムアウト設定を指定します。
validation	検証設定を指定します。
recovery	XA リカバリー設定を指定します。

表22.11 プール要素

要素	説明
min-pool-size	min-pool-size 要素は、プールが保持する最小接続数を指定します。これらは、サブジェクトが接続の要求から認識されるまで作成されません。デフォルトは 0 です。
max-pool-size	max-pool-size 要素は、プールの最大接続数を指定します。各サブプールには、max-pool-size を超える接続は作成されません。デフォルトは **20** です。

要素	説明
prefill	接続プールのプレフィルを試行するかどうか。デフォルトは **false**です。
use-strict-min	min-pool-size を厳格に考慮するかどうかを指定します。デフォルトは **false**です。
flush-strategy	エラーの場合にプールがどのようにフラッシュされるかを指定します。有効な値は **FailingConnectionOnly**（デフォルト）、**IdleConnections**、**EntirePool**です。

表22.12 XA プール要素

要素	説明
min-pool-size	min-pool-size 要素は、プールが保持する最小接続数を指定します。これらは、サブジェクトが接続の要求から認識されるまで作成されません。デフォルトは **0**です。
max-pool-size	max-pool-size 要素は、プールの最大接続数を指定します。各サブプールには、max-pool-size を超える接続は作成されません。デフォルトは **20** です。
prefill	接続プールのプレフィルを試行するかどうか。デフォルトは **false**です。
use-strict-min	min-pool-size を厳格に考慮するかどうかを指定します。デフォルトは **false**です。
flush-strategy	エラーの場合にプールがどのようにフラッシュされるかを指定します。有効な値は **FailingConnectionOnly**（デフォルト）、**IdleConnections**、**EntirePool**です。
is-same-rm-override	is-same-rm-override 要素を使用すると、javax.transaction.xa.XAResource.isSameRM(XAResource)が **true** または **false**を返すかどうかを無条件に設定できます。
interleaving	XA 接続ファクトリのインターリービングを有効にする要素
no-tx-separate-pools	Oracle では、XA 接続を JTA トランザクションの内部と外部の両方で使用することが推奨されません。この問題を回避するには、さまざまなコンテキストに個別のサブプールを作成します。
pad-xid	Xid をパディングするかどうか
wrap-xa-resource	XAResource インスタンスを org.jboss.tm.XAResourceWrapper インスタンスにラップするかどうか

表22.13 セキュリティー要素

要素	説明
application	プール内の接続を区別するために、アプリケーションにより提供されたパラメーター（**getConnection（user, pw からなど）**が使用されることを示します。
security-domain	プール内の接続を区別するために（セキュリティードメインからの）サブジェクトを使用することを指定します。security-domain の内容は認証を処理する JAAS セキュリティーマネージャーの名前です。この名前は JAAS **login-config.xml** 記述子の application-policy/name 属性に相関します。
security-domain-and-application	プール内の接続を区別するために、アプリケーションにより提供されたパラメーター（**getConnection(user, pw)**からなど）または（セキュリティードメインからの）Subject を使用するかどうかを示します。security-domain の内容は認証を処理する JAAS セキュリティーマネージャーの名前です。この名前は JAAS **login-config.xml** 記述子の application-policy/name 属性に相関します。

表22.14 タイムアウト要素

要素	説明
blocking-timeout-millis	blocking-timeout-millis 要素は、例外をスローする前に接続を待機する間にブロックする最大時間（ミリ秒単位）を指定します。このブロックは、接続許可の待機中にのみブロックし、新しい接続の作成に長時間要している場合は例外は発生しません。デフォルトは **30000**（30 秒）です。
idle-timeout-minutes	idle-timeout-minutes 要素は、接続が閉じられるまでのアイドル最大時間（分単位）を指定します。実際の最大時間は、IdleRemover スキャン時間によって異なります。これは、プールの最小 idle-timeout-minutes の 1/2 です。
allocation-retry	allocation retry 要素は、接続の割り当てを再試行する回数を示します。この回数を超えると、例外が発生します。デフォルトは **0** です。
allocation-retry-wait-millis	allocation retry wait millis 要素は、接続割り当てを再試行する間隔をミリ秒単位で指定します。デフォルトは **5000**（5 秒）です。
xa-resource-timeout	**XAResource.setTransactionTimeout()** に渡されます。デフォルトはゼロで、セッターを呼び出さません。秒単位で指定

表22.15 検証要素

要素	説明
background-validation	接続をバックグラウンドで検証するか、使用する前に検証するかを指定する要素
background-validation-minutes	background-validation-minutes 要素は、バックグラウンド検証が実行される時間を分単位で指定します。
use-fast-fail	無効な場合に最初の接続で接続割り当てを失敗させるか(true)、またはプールが潜在的な接続をすべて使い切るまで試行を続けます(false)。デフォルトは **false**です。

表22.16 管理オブジェクト要素

要素	説明
config-property	管理オブジェクト設定プロパティーを指定します。

表22.17 復元要素

要素	説明
recover-credential	復元に使用する名前とパスワードのペアまたはセキュリティードメインを指定します。
recover-plugin	org.jboss.jca.core.spi.recovery.RecoveryPlugin クラスの実装を指定します。

*デプロイメントスキーマは **jboss-as-resource-adapters_1_0.xsd** に、自動アクティベーション用に定義し **http://www.ironjacamar.org/doc/schema/ironjacamar_1_0.xsd** ます。*

バグの報告

22.6. 定義された接続統計の表示

***deployment=name.rar** サブツリーから定義された接続の統計を読み取ることができます。*

*統計はこのレベルで定義され、**standalone.xml** ファイルまたは **domain.xml** ファイルの設定に定義されていない **rar** からアクセスできるためです。*

以下に例を示します。

例22.1 定義された接続統計の表示

```
/deployment=example.rar/subsystem=resource-adapters/statistics=statistics/connection-definitions=java\:\/testMe:read-resource(include-runtime=true)
```

注記

*統計はすべてランタイムのみの情報で、デフォルトは **false** であるため、必ず **include-runtime=true** 引数を指定してください。*

バグの報告

22.7. リソースアダプターの統計

主要統計

サポートされるリソースアダプターの主要統計は下表のとおりです。

表22.18 主要統計

名前	説明
ActiveCount	アクティブな接続の数。各接続はアプリケーションによって使用されているか、プールで使用可能な状態であるかのいずれかになります。
AvailableCount	プールの使用可能な接続の数。
AverageBlockingTime	プールの排他ロックの取得をブロックするために費やされた平均時間。値はミリ秒単位です。
AverageCreationTime	接続の作成に費やされた平均時間。値はミリ秒単位です。
CreatedCount	作成された接続の数。
DestroyedCount	破棄された接続の数。
InUseCount	現在使用中の接続の数。
MaxCreationTime	接続の作成にかかった最大時間。値はミリ秒単位です。

名前	説明
MaxUsedCount	使用される接続の最大数。
MaxWaitCount	同時に接続を待機する要求の最大数。
MaxWaitTime	プールの排他ロックの待機に費やされた最大時間。
TimedOut	タイムアウトした接続の数。
TotalBlockingTime	プールの排他ロックの待機に費やされた合計時間。値はミリ秒単位です。
TotalCreationTime	接続の作成に費やされた合計時間。値はミリ秒単位です。
WaitCount	接続を待機する必要がある要求の数。

バグの報告

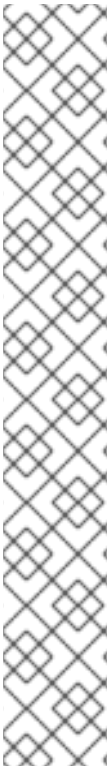
22.8. WEBSPHERE MQ リソースアダプターのデプロイ

WebSphere MQ

***WebSphere MQ** は、分散システム上のアプリケーションがお互いに通信できるようにする **IBM** の **Messaging Oriented Middleware(MOM)** ソフトウェアです。これは、メッセージとメッセージキューを使用して実行できます。**WebSphere MQ** は、メッセージキューにメッセージを配信し、メッセージチャネルを使用してデータを他のキューマネージャーに転送します。**WebSphere MQ** の詳細は、「**WebSphere MQ**」を参照してください。*

概要

*このトピックでは、**Red Hat JBoss Enterprise Application Platform 6** に **WebSphere MQ** リソースアダプターをデプロイおよび設定する手順を説明します。これは、設定ファイルを手動で編集したり、管理 **CLI** ツールを使用するか、または **Web** ベースの管理コンソールを使用して実行できます。*



注記

WebSphere MQ Resource Adapter バージョン **7.5.0.3** とそれ以前のバージョンには、周期的なリカバリが失敗し、**XA** 例外が発生する既知の問題があります。**JBoss EAP** サーバーログに以下のようなメッセージが記録されます。

> **WARN [com.arjuna.ats.jta] (Periodic Recovery) ARJUNA016027: Local XARecoveryModule.xaRecovery got XA exception XAException.XAER_INVAL: javax.transaction.xa.XAException: The method 'xa_recover' has failed with errorCode '-5'.**

バージョン **7.5.0.4** では修正を利用できます。この問題の詳細な説明は、を参照し **http://www-01.ibm.com/support/docview.wss?uid=swg1IC97579** てください。

EAP 6.x では、**WebSphere MQ 8.0** 以降はサポートされないことに注意してください。

前提条件

作業を開始する前に、**WebSphere MQ** リソースアダプターのバージョンを確認して、一部の **WebSphere MQ** 設定プロパティーについて理解しておく必要があります。

- **WebSphere MQ** リソースアダプターは、**wmq.jmsra-VERSION.rar** と呼ばれる **Resource Archive(RAR)** ファイルとして提供されます。バージョン **7.5.0.x** を使用する必要があります。必要なバージョンの詳細は、上記の注記を参照してください。
- 以下の **WebSphere MQ** 設定プロパティーの値を知っている必要があります。これらのプロパティーの詳細は、**WebSphere MQ** 製品のドキュメントを参照してください。
  - **MQ.QUEUE.MANAGER: WebSphere MQ** キューマネージャーの名前
  - **MQ.HOST.NAME: WebSphere MQ** キューマネージャーへの接続に使用するホストの名前
  - **MQ.CHANNEL.NAME: WebSphere MQ** キューマネージャーへの接続に使用するサーバーチャネル

- ***MQ.QUEUE.NAME:*** *宛先キューの名前*
- ***MQ.TOPIC.NAME:*** *宛先トピックの名前*
- ***MQ.PORT: WebSphere MQ*** *キューマネージャーへの接続に使用するポート*
- ***MQ.CLIENT:*** *トランスポートタイプ*

- *送信接続には、以下の設定プロパティーに精通している必要があります。*
  - ***:MQ.CONNECTIONFACTORY.NAME:*** *リモートシステムへの接続を提供する接続ファクトリーインスタンスの名前*

注記

*以下は、**IBM** のデフォルト設定であり、変更される可能性があります。詳細は、**WebSphere MQ** のドキュメントを参照してください。*

*手順**22.8** リソースアダプターの手動でのデプロイ*

1. ***wmq.jmsra-VERSION.rar*** *ファイルを* ***EAP_HOME/standalone/deployments/*** *ディレクトリーにコピーします。*

2. *サーバー設定ファイルにリソースアダプターを追加します。*

a. *エディターで **EAP_HOME/standalone/configuration/standalone-full.xml** ファイルを開きます。*

b. *設定ファイルで **urn:jboss:domain:resource-adapters** サブシステムを見つけます。*

c. *このサブシステムに対して定義されているリソースアダプターがない場合、最初に以下を置き換えます。*

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1"/>
```

上記を以下のように置き換えます。

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
   <resource-adapters>
      <!-- <resource-adapter> configuration listed below -->
   </resource-adapters>
</subsystem>
```

d. *リソースアダプターの設定は、トランザクションサポートとリカバリーが必要であるかどうかによって異なります。トランザクションサポートが必要ない場合は、以下の最初の設定手順を選択します。トランザクションサポートが必要な場合は、**2** 番目の設定手順を選択します。どちらの例でも、**config-property name** 要素は大文字と小文字を区別し、例にあるように入力する必要があります。*

- *非トランザクションデプロイメントの場合、**<!-- <resource-adapter> configuration listed below -->** を以下に置き換えます。*

```
<resource-adapter>
   <archive>
      wmq.jmsra-VERSION.rar
   </archive>
   <transaction-support>NoTransaction</transaction-support>
   <connection-definitions>
      <connection-definition
            class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
```

```
            jndi-name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
            pool-name="MQ.CONNECTIONFACTORY.NAME">
          <config-property name="hostName">
            MQ.HOST.NAME
          </config-property>
          <config-property name="port">
            MQ.PORT
          </config-property>
          <config-property name="channel">
            MQ.CHANNEL.NAME
          </config-property>
          <config-property name="transportType">
            MQ.CLIENT
          </config-property>
          <config-property name="queueManager">
            MQ.QUEUE.MANAGER
          </config-property>
          <security>
            <security-domain>MySecurityDomain</security-domain>
          </security>
      </connection-definition>
    </connection-definitions>
    <admin-objects>
      <admin-object
            class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
            jndi-name="java:jboss/MQ.QUEUE.NAME"
            pool-name="MQ.QUEUE.NAME">
          <config-property name="baseQueueName">
            MQ.QUEUE.NAME
          </config-property>
          <config-property name="baseQueueManagerName">
            MQ.QUEUE.MANAGER
          </config-property>
      </admin-object>
      <admin-object class-name="com.ibm.mq.connector.outbound.MQTopicProxy"
            jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
          <config-property name="baseTopicName">
            MQ.TOPIC.NAME
          </config-property>
          <config-property name="brokerPubQueueManager">
            MQ.QUEUE.MANAGER
          </config-property>
      </admin-object>
    </admin-objects>
</resource-adapter>
```

VERSION を RAR 名の実際のバージョンに置き換えてください。

- トランザクションデプロイメントの場合は、**<!-- <resource-adapter> configuration listed below -->** を以下に置き換えます。

```
<resource-adapter>
   <archive>
      wmq.jmsra-VERSION.rar
   </archive>
   <transaction-support>XATransaction</transaction-support>
   <connection-definitions>
      <connection-definition
            class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
            jndi-name="java:jboss/MQ.CONNECTIONFACTORY.NAME"
            pool-name="MQ.CONNECTIONFACTORY.NAME">
         <config-property name="hostName">
            MQ.HOST.NAME
         </config-property>
         <config-property name="port">
            MQ.PORT
         </config-property>
         <config-property name="channel">
            MQ.CHANNEL.NAME
         </config-property>
         <config-property name="transportType">
            MQ.CLIENT
         </config-property>
         <config-property name="queueManager">
            MQ.QUEUE.MANAGER
         </config-property>
        <security>
            <security-domain>MySecurityDomain</security-domain>
         </security>
         <recovery>
          <recover-credential>
           <security-domain>RECOVERY_SECURITY_DOMAIN</security-domain>
          </recover-credential>
         </recovery>
      </connection-definition>
   </connection-definitions>
   <admin-objects>
      <admin-object
            class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
            jndi-name="java:jboss/MQ.QUEUE.NAME"
            pool-name="MQ.QUEUE.NAME">
         <config-property name="baseQueueName">
            MQ.QUEUE.NAME
         </config-property>
         <config-property name="baseQueueManagerName">
            MQ.QUEUE.MANAGER
         </config-property>
      </admin-object>
      <admin-object class-name="com.ibm.mq.connector.outbound.MQTopicProxy"
            jndi-name="java:jboss/MQ.TOPIC.NAME" pool-
name="MQ.TOPIC.NAME">
         <config-property name="baseTopicName">
```

```
            MQ.TOPIC.NAME
        </config-property>
        <config-property name="brokerPubQueueManager">
            MQ.QUEUE.MANAGER
        </config-property>
    </admin-object>
  </admin-objects>
</resource-adapter>
```

VERSION を *RAR* 名の実際のバージョンに置き換えてください。*USER_NAME* と *PASSWORD* は有効なユーザー名とパスワードに置き換える必要もあります。

注記

トランザクションをサポートするために、**<transaction-support>** 要素が **XATransaction** に設定されていました。**XA** リカバリーをサポートするために、**<recovery>** 要素が接続定義に追加されました。

e.

JBoss EAP 6 の *EJB3* メッセージングシステムのデフォルトプロバイダーを *HornetQ* から *WebSphere MQ* に変更する場合は、以下のように *urn:jboss:domain:ejb3:1.2* サブシステムを変更します。

以下を置き換えます。

```
<mdb>
   <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
   <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

上のコマンドを、下のコマンドに置き換えます。

```
<mdb>
   <resource-adapter-ref resource-adapter-name="wmq.jmsra-VERSION.rar"/>
   <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

VERSION を RAR 名の実際のバージョンに置き換えてください。

手順22.9 リソースアダプターを使用するように MDB コードを変更します。

- *以下のように、MDB コードで ActivationConfigProperty および ResourceAdapter を設定します。すべての propertyName 要素は大文字と小文字を区別するため、以下のように入力する必要があります。*

```
@MessageDriven(name="WebSphereMQMDB", activationConfig = {
   @ActivationConfigProperty(propertyName = "destinationType",propertyValue =
"javax.jms.Queue"),
   @ActivationConfigProperty(propertyName = "useJNDI", propertyValue = "false"),
   @ActivationConfigProperty(propertyName = "hostName", propertyValue =
"MQ.HOST.NAME"),
   @ActivationConfigProperty(propertyName = "port", propertyValue = "MQ.PORT"),
   @ActivationConfigProperty(propertyName = "channel", propertyValue =
"MQ.CHANNEL.NAME"),
   @ActivationConfigProperty(propertyName = "queueManager", propertyValue =
"MQ.QUEUE.MANAGER"),
   @ActivationConfigProperty(propertyName = "destination", propertyValue =
"MQ.QUEUE.NAME"),
   @ActivationConfigProperty(propertyName = "transportType", propertyValue =
"MQ.CLIENT")
})

@ResourceAdapter(value = "wmq.jmsra-VERSION.rar")
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public class WebSphereMQMDB implements MessageListener {
}
```

  VERSION を RAR 名の実際のバージョンに置き換えてください。

バグの報告

22.9. WEBSPHERE MQ リソースアダプターのインストール

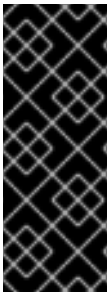
JBoss Active MQ(A-MQ)リソースアダプターを JBoss EAP 6 にインストールし、JBoss A-MQ 6.1.0 と動作させるには、の手順に従い https://access.redhat.com/documentation/ja-JP/Red_Hat_JBoss_A-MQ/6.1/html/Integrating_with_JBoss_Enterprise_Application_Platform/DeployRar-InstallRar.html ます。

バグの報告

22.10. サードパーティー JMS プロバイダーで使用する汎用 JMS リソースアダプターの設定

概要

JBoss EAP 6 は、サードパーティーの JMS プロバイダーと連携するように設定できますが、すべての JMS プロバイダーが Java アプリケーションプラットフォームとの統合のために JMS JCA リソースアダプターを生成する訳ではありません。この手順では、JBoss EAP 6 に含まれる汎用 JMS リソースアダプターを設定して JMS プロバイダーに接続する手順を説明します。この手順では、Tibco EMS 6.3 が JMS プロバイダーの例として使用されますが、他の JMS プロバイダーでは異なる設定が必要になる場合があります。



重要

汎用 JMS リソースアダプターを使用する前に、JMS プロバイダーをチェックして、JBoss EAP 6 で使用できる独自のリソースアダプターがあるかどうかを確認します。汎用 JMS JCA リソースアダプターは、JMS プロバイダーが固有のリソースアダプターを提供していない場合にのみ使用してください。

前提条件

- JMS プロバイダーサーバーが設定され、使用できる状態である。プロバイダーの JMS 実装に必要なバイナリーが必要になります。
- また、JMS リソース（接続ファクトリー、キュー、トピック）を検索することができるように、以下の JMS プロバイダープロパティーの値を知る必要があります。
  - java.naming.factory.initial
  - java.naming.provider.url
  - java.naming.factory.url.pkgs

  この手順で使用する XML の例では、これらのパラメーターはそれぞれ、PROVIDER_FACTORY_INITIAL、PROVIDER_URL、PROVIDER_CONNECTION_FACTORY として記述されています。これらのプレースホルダーを、お使いの環境の JMS プロバイダーの値に置き換えてください。

手順22.10 サードパーティー JMS プロバイダーで使用する汎用 JMS リソースアダプターの設定

1. ***JMS** プロバイダーの **JBoss** モジュールの作成*

   ***JMS** プロバイダーへの接続および通信に必要なすべてのライブラリーが含まれる **JBoss** モジュールを作成します。このモジュールの名前は **org.jboss.genericjms.provider** です。*

   a.
   ***EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main**というディレクトリー構造を作成します。*

   b.
   *プロバイダーの **JMS** 実装に必要なバイナリーを **EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** にコピーします。*

   注記

   ***Tibco EMS** の場合、必要なバイナリーは、**Tibco** インストールの **lib** ディレクトリーにある **tibjms .jar** および **t ibcrypt.jar** です。*

   c.
   *以下のように、**EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** に **module.xml** ファイルを作成し、以前の手順の **JAR** ファイルをリソースとして一覧表示します。*

```
<module xmlns="urn:jboss:module:1.1" name="org.jboss.genericjms.provider">
  <resources>
    <!-- all jars required by the JMS provider, in this case Tibco -->
    <resource-root path="tibjms.jar"/>
    <resource-root path="tibcrypt.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
    <module name="javax.jms.api"/>
  </dependencies>
</module>
```

2. ***JMS** プロバイダーへの **JNDI** 外部コンテキストの設定*

   ***JMS** リソース（接続ファクトリーおよび宛先）は **JMS** プロバイダーで検索されます。**JBoss EAP 6** インスタンスに外部コンテキストを追加し、このリソースの **ローカル** ルックアップがリモート **JMS** プロバイダー上のリソースを自動的に検索できるようにします。*

注記

この手順では、**EAP_HOME/standalone/configuration/standalone-full.xml** を **JBoss EAP 6** 設定ファイルとして使用します。

EAP_HOME/standalone/configuration/standalone-full.xml の **< subsystem xmlns="urn:jboss:domain:naming:1.4">** の下に以下を 追加します。

```
<bindings>
  <external-context name="java:global/remoteJMS/"
    module="org.jboss.genericjms.provider"
    class="javax.naming.InitialContext">
    <environment>
      <property name="java.naming.factory.initial"
value="${PROVIDER_FACTORY_INITIAL}"/>
      <property name="java.naming.provider.url" value="${PROVIDER_URL}"/>
      <property name="java.naming.factory.url.pkgs" value="${PROVIDER_URL_PKGS}"/>
    </environment>
  </external-context>
</bindings>
```

この **3** つのプロパティーの値は、リモート **JMS** プロバイダーに接続するために正しい値に置き換える必要があります。プレースホルダーテキストを置き換えて **${}** をそのまま維持する場合は注意が必要です。

3. 文字列によるルックアップの有効化

JNDI lookup(Name) メソッドをサポートしない **JMS** プロバイダー（**Tibco EMS** など）がいくつかあります。このような場合、この問題を回避するには、値が **true** の **org.jboss.as.naming.lookup.by.string** プロパティーを追加します。

例**22.2** **Tibco EMS** を使用した文字列によるルックアップの有効化

Tibco EMS への **external-context** の完全な定義は以下のようになります。

```
<bindings>
  <external-context name="java:global/remoteJMS/"
      module="org.jboss.genericjms.provider"
      class="javax.naming.InitialContext">
    <environment>
      <property name="java.naming.factory.initial"
value="com.tibco.tibjms.naming.TibjmsInitialContextFactory"/>
      <property name="java.naming.provider.url"
value="TIBCO_EMS_SERVER_HOST_NAME:PORT"/>
      <property name="java.naming.factory.url.pkgs" value="com.tibco.tibjms.naming"/>
      <property name="org.jboss.as.naming.lookup.by.string" value="true"/>
```

```
    </environment>
  </external-context>
</bindings>
```

この外部コンテキストでは、**java:global/remoteJMS/** で始まるリソースへの **JNDI** ルックアップがリモート **JMS** プロバイダーで行われます（この接頭辞の削除後）。たとえば、メッセージ駆動 **Bean** が **java:global/remoteJMS/Queue1** の **JNDI** ルックアップを実行する場合、外部コンテキストはリモート **JMS** プロバイダーに接続し、**Queue1** リソースのルックアップを実行します。

4. 汎用 **JMS** リソースアダプターの設定

EAP_HOME/standalone/configuration/standalone-full.xml で、汎用リソースアダプター設定を **< subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">** に追加します。

例**22.3** **Tibco EMS** リソースアダプターの設定

Tibco EMS の完全なリソースアダプター定義は以下のようになります。

```
<resource-adapter id="org.jboss.genericjms">
  <module slot="main" id="org.jboss.genericjms"/>
  <transaction-support>NoTransaction</transaction-support>
  <connection-definitions>
    <connection-definition class-
name="org.jboss.resource.adapter.jms.JmsManagedConnectionFactory"
      jndi-name="java:/jms/XAQCF"
      pool-name="XAQCF">
      <config-property name="ConnectionFactory">
        XAQCF
      </config-property>
      <config-property name="JndiParameters">

java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFactory;jav
a.naming.provider.url=TIBCO_EMS_SERVER_HOST_NAME:PORT
      </config-property>
      <security>
        <application/>
      </security>
    </connection-definition>
  </connection-definitions>
</resource-adapter>
```

5. デフォルトのメッセージ駆動 **Bean** プールを汎用リソースアダプターで設定します。

EAP_HOME/standalone/configuration/standalone-full.xml の **< subsystem xmlns="urn:jboss:domain:ejb3:1.5">** で、以下のように **< mdb>** 設定を 更新します。

```
<mdb>
  <resource-adapter-ref resource-adapter-name="org.jboss.genericjms"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

結果

これで、汎用 **JMS** リソースアダプターが設定され、使用できるようになりました。新しいメッセージ駆動 **Bean** を作成するときは、以下のようなコードを使用してリソースアダプターを使用します。

例**22.4** 汎用リソースアダプターの使用

```
@MessageDriven(name = "HelloWorldQueueMDB", activationConfig = {
  // The generic JMS resource adapter requires the JNDI bindings
  // for the actual remote connection factory and destination
  @ActivationConfigProperty(propertyName = "connectionFactory", propertyValue =
"java:global/remoteJMS/XAQCF"),
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
"java:global/remoteJMS/Queue1"),
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-
acknowledge") })
  public class HelloWorldQueueMDB implements MessageListener {
  public void onMessage(Message message) {
  // called every time a message is received from the `Queue1` queue on the JMS provider.
  }
}
```

警告

汎用 **JMS** リソースアダプターを使用する場合は、潜在的な **NullPointerException (NPE)**エラーを回避するために、必ずセッションが機能するように設定してください。**NPE** エラーが発生するのは、**Java EE** 仕様でパラメーターが処理され **ない** ことが示されているとき、汎用 **JMS** リソースアダプターがパラメーターの処理を試みるためです。

```
connection.createSession(true, Session.SESSION_TRANSACTED);
```

例22.5 プール接続の使用

プールされた接続ファクトリーをリソースアダプターから使用することもできます。

```
@Resource(lookup = "java:/jms/XAQCF")
private ConnectionFactory cf;
```

例22.6 ルックアップの実行

外部コンテキストから直接リソースを挿入することはできませんが、外部コンテキストを挿入してからルックアップを実行することはできます。たとえば、Tibco EMS ブローカーにデプロイされたキューのルックアップは以下のようになります。

```
@Resource(lookup = "java:global/remoteJMS")
private Context context;

...

Queue queue = (Queue) context.lookup("Queue1")
```

バグの報告

22.11. リモート接続の HORNETQ JCA アダプターの設定

手順22.11 2 つのリモート JBoss EAP インスタンスに接続するように RA アダプターを設定

HornetQ RA がリモート EAP インスタンスに接続するように設定するには、リモート JBoss EAP インスタンスを参照するコネクターを 2 つ作成する必要があります。

1. アウトバウンドソケットバインディングを作成します。

```
<outbound-socket-binding name="remote-jms-server-a">
 <remote-destination host="${remote.jms.server.one.bind.address:127.0.0.1}"
port="${remote.jms.server.one.bind.port:5445}"/>
</outbound-socket-binding>
<outbound-socket-binding name="remote-jms-server-b">
 <remote-destination host="${remote.jms.server.two.bind.address:127.0.0.1}"
port="${remote.jms.server.two.bind.port:5545}"/>
</outbound-socket-binding>
```

2.

*2 つの**netty** コネクターを作成します。*

```
<netty-connector name="netty-remote-node-a" socket-binding="remote-jms-server-a"/>
<netty-connector name="netty-remote-node-b" socket-binding="remote-jms-server-a"/>
```

3.

*2 つの **netty** コネクターを使用して **RA** 設定を作成します。*

```
<pooled-connection-factory name="hornetq-remote-ra">
  <inbound-config>
    <use-jndi>true</use-jndi>

<jndiparams>java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextF
actory;java.naming.provider.url=remote://${remote.jms.server.one.bind.address}:4447,
${remote.jms.server.two.bind.address}:4547;java.naming.security.principal=${user.na
me};java.naming.security.credentials=${user.password}</jndi-params>
  </inbound-config>
  <transaction mode="xa"/>
  <user>${user.name}</user>
  <password>${user.password}</password>
  <connectors>
    <connector-ref connector-name="netty-remote-node-a"/>
    <connector-ref connector-name="netty-remote-node-b"/>
  </connectors>
  <entries>
  <entry name="java:/RemoteJmsXA"/>
  </entries>
</pooled-connection-factory>
```

4.

*@ResourceAdapter アノテーションを使用してリソースアダプターを使用するように **MDB** にアノテーションを付けます。*

```
@MessageDriven(name = "InQueueMDB", activationConfig = {
@ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Queue"),
@ActivationConfigProperty(propertyName = "destination", propertyValue =
"${hornetq.in.queue.short}"),
@ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =
"Auto-acknowledge"),
@ActivationConfigProperty(propertyName = "useJNDI",propertyValue = "true")
},mappedName = "${hornetq.inf.queue.long}")
@ResourceAdapter("hornetq-remote-ra")
```

5.
以下の **CLI** コマンドを実行して、デプロイメント記述子でプロパティーを使用し、プロパティーの置換を有効にします。

```
/subsystem=ee:write-attribute(name=jboss-descriptor-property-replacement,value=true)
/subsystem=ee:write-attribute(name=spec-descriptor-property-replacement,value=true)
/subsystem=ee:write-attribute(name=annotation-property-replacement,value=true)
```

6.
MDB からメッセージを送信するために、外部コンテキストを作成してリモート宛先を見つけます。

```
<bindings>
 <external-context name="java:global/remote" module="org.jboss.remote-naming" class="javax.naming.InitialContext">
  <environment>
   <property name="java.naming.factory.initial" value="org.jboss.naming.remote.client.InitialContextFactory"/>
   <property name="java.naming.provider.url" value="remote://${remote.jms.server.one.bind.address:ragga}:4447,${remote.jms.server.two.bind.address:ragga}:4547"/>
   <property name="java.naming.security.principal" value="${user.name}"/>
   <property name="java.naming.security.credentials" value="${user.password}"/>
  </environment>
 </external-context>
</bindings>
```

7.
MDB コードは次のようになります。

```
@MessageDriven(name = "InQueueMDB", activationConfig = {
 @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
 @ActivationConfigProperty(propertyName = "destination", propertyValue = "${hornetq.in.queue.short}"),
 @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-acknowledge"),
 @ActivationConfigProperty(propertyName = "useJNDI",propertyValue = "true"),
 @ActivationConfigProperty(propertyName = "hA", propertyValue = "true")
},mappedName = "${hornetq.inf.queue.full}")
@ResourceAdapter("hornetq-remote-ra")
 public class InQueueMDB implements MessageListener {
```

```
 private static final Logger log = Logger.getLogger(InQueueMDB.class);
 @Resource(lookup = "java:global/remote")
 private InitialContext context;
 @Resource( name = "${hornetq.jms.connection}")
 private ConnectionFactory  qcf;
 public void onMessage(Message message){
 try {
  if ( message instanceof TextMessage){
   Object obj =  (Queue) context.lookup("/jms/queue/outQueue");
   qConnection = (QueueConnection) qcf.createConnection("quickuser","quick123+");
   qSession = qConnection.createQueueSession(true,
Session.SESSION_TRANSACTED);
   qSender = qSession.createSender(outQueue);
   qSender.send(message);
```

8.

***hornetq RA** モジュールには、上記の **MDB** コードの **remoting-naming** 依存関係が含まれることに注意してください。*

```
<module xmlns="urn:jboss:module:1.1" name="org.hornetq.ra">
 <properties>
  <property name="jboss.api" value="private"/>
 </properties>
 <resources>
  <resource-root path="hornetq-ra-2.3.25.Final-redhat-1.jar"/>
  <!-- Insert resources here -->
 </resources>
 <dependencies>
  <module name="org.hornetq"/>
  <module name="org.jboss.as.transactions"/>
  <module name="org.jboss.as.messaging"/>
  <module name="org.jboss.jboss-transaction-spi"/>
  <module name="org.jboss.logging"/>
  <module name="org.jboss.remote-naming"/>
  <module name="javax.api"/>
  <module name="javax.jms.api" />
  <module name="org.jboss.jts"/>
  <module name="org.jboss.netty"/>
  <module name="org.jgroups"/>
  <module name="javax.resource.api"/>
 </dependencies>
</module>
```

9.

***JMS API** がアプリケーションに表示されるようにするには、**org.hornetq** をグローバルモジュールに追加する必要があります。*

```
<global-modules>
```

```
    <module name="org.jboss.common-core" slot="main"/>
    <module name="org.hornetq" slot="main"/>
  </global-modules>
```

バグの報告

第23章 HIBERNATE SEARCH

23.1. HIBERNATE SEARCH の使用

23.1.1. Hibernate Search について

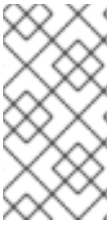
Hibernate Search は、**Hibernate** アプリケーションに完全なテキスト検索機能を提供します。これは、全文、あいまい、位置情報検索などの **SQL** ベースのソリューションには適していないアプリケーションの検索に適しています。**Hibernate Search** は **Apache Lucene** を全文検索エンジンとして使用しますが、メンテナンスのオーバーヘッドを最小限に抑えるように設計されています。設定が完了したら、インデックス、クラスタリング、およびデータ同期は透過的に維持されるため、ビジネス要件を満たすことができます。

バグの報告

23.1.2. 概要

Hibernate Search はインデックスコンポーネントとインデックス検索コンポーネントで構成されており、どちらも **Apache Lucene** によってサポートされます。エンティティーがデータベースで挿入、更新、または削除されるたびに、**Hibernate Search** は（**Hibernate** イベントシステムから）このイベントを追跡し、インデックスの更新をスケジュールします。これらの更新はすべて、**Apache Lucene API** と直接対話せずに処理されます。代わりに、基礎となる **Lucene** インデックスとの対話は **IndexManager** で処理されます。

インデックスが作成されると、基盤の **Lucene** インフラストラクチャーを処理するのではなく、エンティティーを検索し、管理エンティティーのリストを返すことができます。同じ永続コンテキストが **Hibernate** と **Hibernate Search** 間で共有されます。**FullTextSession** クラスは **Hibernate Session** クラスの上に構築され、アプリケーションコードで統一された **org.hibernate.Query** または **javax.persistence.Query API** が **HQL**、**JPA-QL** またはネイティブクエリーの実行方法と全く同じ方法を使用できます。



注記

データベースと **Hibernate Search** の両方に対しては、**JDBC** または **JTA** で操作を実行するために、トランザクションで実行することが推奨されます。

注記

Hibernate Search はアトミックな会話として知られる、**Hibernate / EntityManager** の長い対話パターンで完全に問題なく機能します。

バグの報告

23.1.3. インデックスマネージャー

エンティティーがデータベースから挿入、更新、または削除されるたびに、**Hibernate Search** は **Hibernate** イベントシステムからこのイベントを追跡し、インデックスの更新をスケジュール設定します。基礎となる **Lucene** インデックスとの対話は **IndexManager** によって処理されます。各インデックスは名前で一意に識別されます。デフォルトでは、**IndexManager** と **Lucene** インデックスの間に **1** 対 **1** の関係があります。**IndexManager** は、選択した バックエンド、リーダーストラテジー、および選択した **DirectoryProvider** を含む特定のインデックス設定を抽象化します。

バグの報告

23.1.4. ディレクトリープロバイダーについて

Hibernate Search インフラストラクチャーの一部である **Apache Lucene** には、インデックスの保管にディレクトリーという概念があります。**Hibernate Search** は、ディレクトリープロバイダーを介して **Lucene Directory** インスタンスの初期化および設定を処理し ます。

Directory_provider プロパティーは、インデックスを保存するために使用するディレクトリープロバイダーを指定します。デフォルトのファイルシステムディレクトリープロバイダーは、ローカルファイルシステム を使用してインデックスを保存するファイルシステムです。

バグの報告

23.1.5. Worker について

Lucene インデックスへの更新は、すべてのエンティティーの変更を受け取る **Hibernate Search** ワーカー によって処理されます。最も一般的なコンテキストはトランザクションですが、エンティティーの変更数や他のアプリケーション（ライフサイクル）イベントの数によって異なる場合があります。

効率性を向上するために、対話はバッチ化され、通常はコンテキストの終了時に適用されます。トランザクション以外では、インデックスの更新操作は実際のデータベース操作の直後に実行されます。継続中のトランザクションの場合、インデックス更新操作はトランザクションのコミットフェーズに対してスケジュール設定され、トランザクションのロールバック時に破棄されます。**worker** は特定のバッチサイズ制限で設定でき、その後、インデックスはコンテキストに関係なく実行されます。

ワーカー設定オプションの詳細は、「ワーカー設定」 を参照してください。

この手法では、インデックス更新を処理するのに以下のようなメリットがあります。

- パフォーマンス: 操作がバッチで実行されると、**Lucene** インデックスのパフォーマンスが向上しました。
- **ACIDity:** 実行されるワークはデータベーストランザクションによって実行されたものと同じスコーピングを持ち、トランザクションがコミットされた場合にのみ実行されます。これは、厳密な意味では **ACID** ではありませんが、**ACID** の動作では、ソースからいつでも再構築できるため、完全なテキスト検索インデックスにはあまり役に立ちません。

2 つのバッチモード（スコープとトランザクション）は、自動コミットの動作とトランザクション動作に相当します。パフォーマンスの観点からは、**transactional** モードが推奨されます。スコープの選択は透過的に行われます。**Hibernate Search** はトランザクションの存在を検出し、スコーピングを調整します（「ワーカー設定」を参照してください）。

バグの報告

23.1.6. バックエンド設定と操作

23.1.6.1. バックエンド

Hibernate Search はさまざまなバックエンドを使用して、作業のバッチを処理します。バックエンドは、設定オプション **default.worker.backend** に制限されません。このプロパティーは、バックエンド設定の一部である **BackendQueueProcessor** インターフェースの実装を指定します。バックエンド（**JMS** バックエンドなど）を設定するには、追加の設定が必要です。

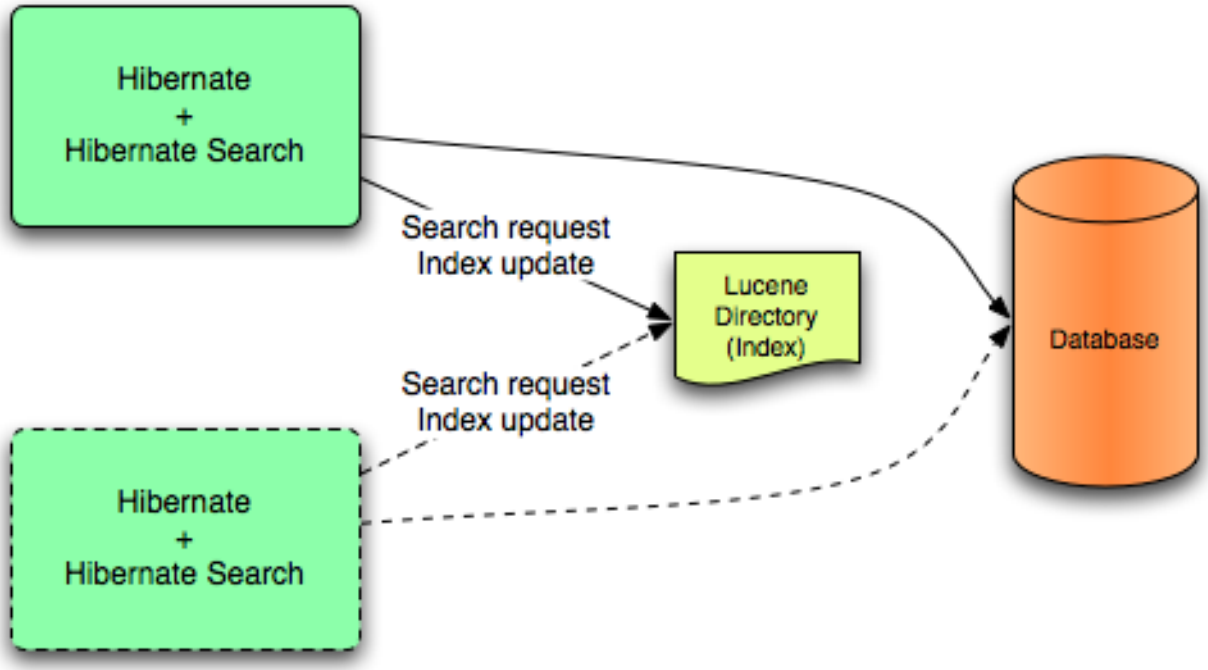
バグの報告

23.1.6.2. Lucene

Lucene モードでは、ノード**(JVM)**のすべてのインデックス更新は、ディレクトリープロバイダーを使用して同じノードで **Lucene** ディレクトリーに対して実行されます。このモードは、クラスター化さ

れていない環境や、共有ディレクトリーストアを持つクラスター環境で使用します。

図23.1 Lucene バックエンドの設定



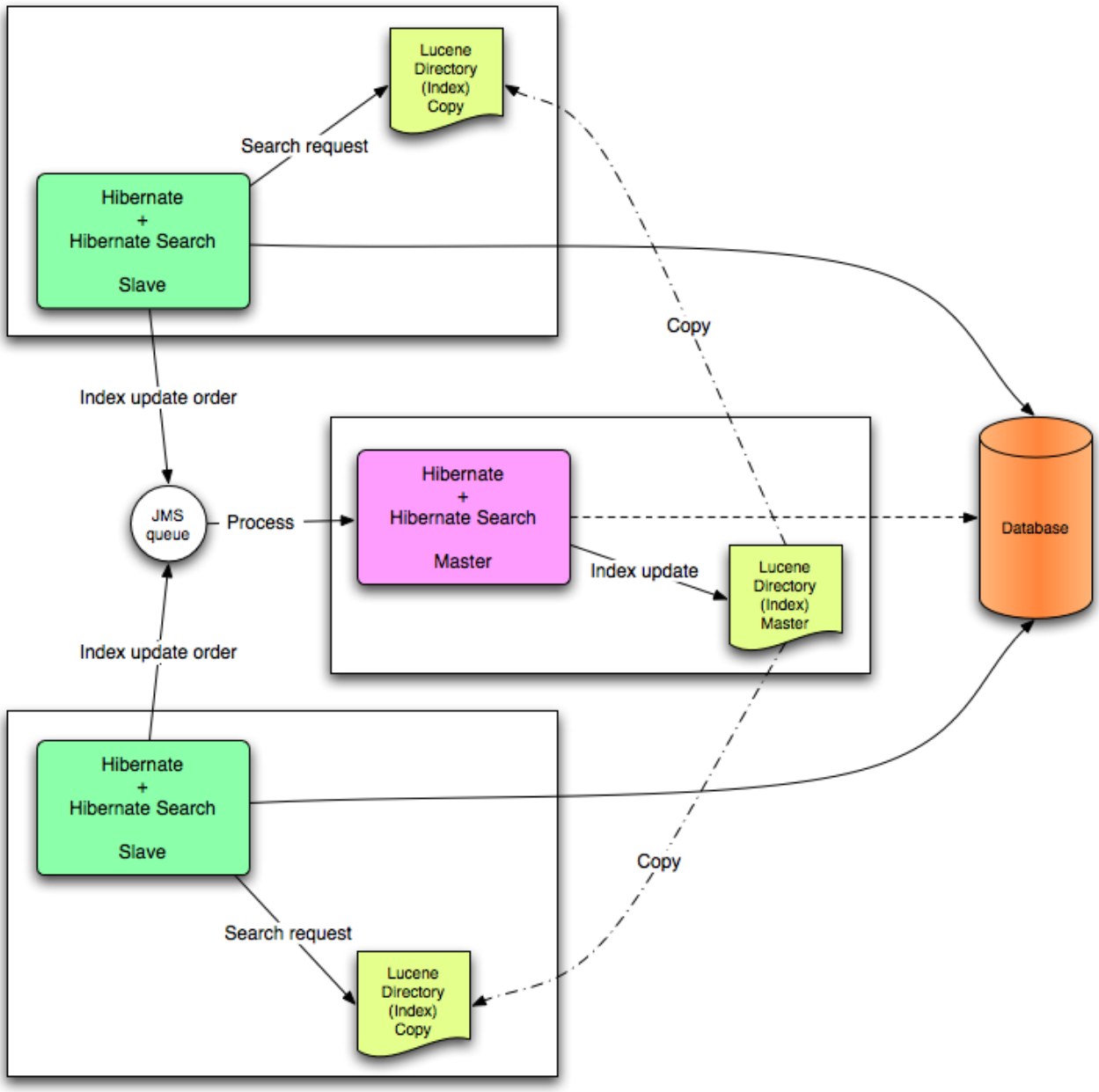
Lucene モードは、**Directory** がロックストラテジーを管理する非クラスター化アプリケーションまたはクラスター化されたアプリケーションをターゲットにします。**Lucene** モードの主な利点は、**Lucene** クエリーの変更の単純化と即時表示です。**Near Real Time(NRT)**バックエンドは、クラスター化されず、共有されていないインデックス設定の代替バックエンドです。

バグの報告

23.1.6.3. JMS

ノードのインデックス更新は **JMS** キューに送信されます。一意のリーダーはキューを処理し、マスターインデックスを更新します。マスターインデックスは、マスター/スレーブパターンを確立するために定期的にスレーブコピーに複製されます。マスターは **Lucene** インデックスの更新を行います。スレーブは読み取りおよび書き込み操作を受け入れますが、ローカルインデックスコピーの読み取り操作を処理します。マスターは **Lucene** インデックスの更新のみを行います。更新操作にローカルの変更を適用するのは、マスターのみです。

図23.2 JMS バックエンドの設定



*このモードは、スループットが重要であり、インデックス更新の遅延が発生するクラスター環境をターゲットにします。**JMS** プロバイダーは信頼性を確保し、スレーブを使用してローカルインデックスコピーを変更します。*

バグの報告

23.1.7. リーダーストラテジー

*クエリーの実行時に、**Hibernate Search** はリーダーストラテジーを使用して **Apache Lucene** インデックスと対話します。アプリケーションのプロファイル（ほとんどの場合、非同期インデックス更新など）に基づいてリーダーストラテジーを選択します。*

バグの報告

23.1.7.1. Shared ストラテジー

共有 ストラテジーを使用して、**IndexReader** が更新された場合に、**Hibernate Search** は指定の **Lucene** インデックスに同じ **IndexReader** を共有します。**IndexReader** が更新されないと、新しいものが開き、提供されます。各 **IndexReader** は複数の **SegmentReader** で構成されています。共有ストラテジーは、前回開いた後に修正または作成されたセグメントを分離し、以前のインスタンスのすでにロードされているセグメントを共有します。これはデフォルトのストラテジーです。

バグの報告

23.1.7.2. Not-shared ストラテジー

not-shared ストラテジーを使用すると、クエリーが実行されるたびに **Lucene IndexReader** が開きます。**IndexReader** を開いて起動するのは、コストのかかる操作です。そのため、各クエリー実行の **IndexReader** を開くことは、効率的なストラテジーではありません。

バグの報告

23.1.7.3. カスタムリーダーストラテジー

org.hibernate.search.reader.ReaderProvider の実装を使用して、カスタムリーダーストラテジーを作成できます。実装はスレッドセーフである必要があります。

バグの報告

23.1.7.4. リーダーストラテジーの設定

以下のように、ストラテジーをデフォルト（共有）から **not-shared** に変更します。

```
hibernate.search.[default|<indexname>].reader.strategy = not-shared
```

または、**my.corp.myapp.CustomReaderProvider** をカスタムストラテジー実装に置き換えてリーダーストラテジーをカスタマイズします。

```
hibernate.search.[default|<indexname>].reader.strategy =
my.corp.myapp.CustomReaderProvider
```

バグの報告

23.2. 設定

23.2.1. 最小設定

Hibernate Search は、設定および操作の柔軟性を提供するように設計されており、ほとんどのユースケースに合わせてデフォルト値を慎重に選択しています。少なくとも、**Directory Provider** をそのプロパティーとともに設定する必要があります。デフォルトの **Directory Provider** は、インデックス ストレージにローカルファイルシステムを使用するファイルシステムです。利用可能なディレクトリープロバイダーおよびその設定の詳細は、「**DirectoryProvider** の設定」 を参照してください。

Hibernate を直接使用する場合は、**DirectoryProvider** などの設定を、**hibernate.properties** または **hibernate.cfg.xml** のいずれかの設定ファイルに設定する必要があります。**JPA** 経由で **Hibernate** を使用している場合、設定ファイルは **persistence.xml** です。

バグの報告

23.2.2. IndexManager の設定

Hibernate Search は、このインターフェースにいくつかの実装を提供します。

- ディレクトリーベース の **: Lucene Directory** 抽象化を使用してインデックスファイルを管理するデフォルトの実装。
- **near-real-time:** コミット時にディスクへの書き込みをフラッシュしないようにします。このインデックスマネージャーも **Directory** に基づいていますが、**Lucene** のほぼリアルタイム **(NRT)**機能を使用します。

デフォルト以外の **IndexManager** を指定するには、以下のプロパティーを指定します。

```
hibernate.search.[default|<indexname>].indexmanager = near-real-time
```

バグの報告

23.2.2.1. Directory-based

Directory ベース の実装は、デフォルトの **IndexManager** 実装です。これは高度な設定が可能で、リーダーストラテジー、バックエンド、およびディレクトリープロバイダーに個別の設定を可能にします。

バグの報告

23.2.2.2. Near Real Time

NRTIndexManager はデフォルトの **IndexManager** の拡張機能であり、低レイテンシーのインデックス書き込みの **Lucene NRT(Near Real Time)**機能を利用します。ただし、**lucene** 以外の代替バックエンドの構成設定は、**Directory** で排他的書き込みロックを取得します。

IndexWriter は、低レイテンシーを提供するためにディスクへの変更をすべてフラッシュしません。クエリーは、フラッシュされていないインデックスライターバッファーから更新された状態を読み取ることができます。ただし、これは **IndexWriter** が強制終了されたり、アプリケーションがクラッシュした場合に更新が失われる可能性があるため、インデックスを再構築する必要があります。

Near Real Time 設定は、上記のデメリットとマスターノードを個別に設定できるため、データが限定されたクラスター化されていない **Web** サイトに対して推奨されます。

バグの報告

23.2.2.3. カスタム

カスタム **IndexManager** を設定するには、カスタム実装の完全修飾クラス名を指定します。以下のように、実装に **no-argument** コンストラクターを設定します。

```
[default|<indexname>].indexmanager = my.corp.myapp.CustomIndexManager
```

カスタムインデックスマネージャーの実装には、デフォルトの実装と同じコンポーネントは必要ありません。たとえば、**Directory** インターフェースを公開しないリモートインデックスサービスに委譲

します。

バグの報告

23.2.3. DirectoryProvider の設定

DirectoryProvider は、**Lucene Directory** に関する **Hibernate Search** 抽象化で、基礎となる **Lucene** リソースの設定および初期化を処理します。 ディレクトリープロバイダーおよびそのプロパティー は、**Hibernate Search** で利用可能なディレクトリープロバイダーと、対応するオプションの一覧を表示します。

インデックス化された各エンティティーは **Lucene** インデックスに関連付けられます (複数のエンティティーが同じインデックスを共有している場合を除きます)。インデックスの名前は、**@Indexed** アノテーションの **index** プロパティーによって指定されます。**index** プロパティーが指定されていない場合、インデックス化されたクラスの完全修飾名が名前として使用されます（推奨）。

DirectoryProvider および追加のオプションは、接頭辞 **hibernate.search.<indexname>** を使用して設定できます。名前 **default (hibernate.search.default)**は予約されており、すべてのインデックスに適用されるプロパティーを定義するために使用できます。**例23.2「ディレクトリープロバイダーの設定」** は、**hibernate.search.default.directory_provider** を使用してデフォルトのディレクトリープロバイダーをファイルシステムに設定する方法を示しています。**hibernate.search.default.indexBase** 次に、インデックスのデフォルトベースディレクトリーを設定します。その結果、エンティティー **Status** のインデックスが **/usr/lucene/indexes/org.hibernate.example.Status** に作成されます。

ただし、**Rule** エンティティーのインデックスは、このエンティティーのデフォルトディレクトリープロバイダーはプロパティー **hibernate.search.Rules.directory_provider** によって上書きされるため、インメモリーディレクトリーを使用しています。

最後に、アクション エンティティーは、**hibernate.search.Actions.directory_provider** で指定されたカスタムディレクトリープロバイダー **CustomDirectoryProvider** を使用します。

> 例**23.1** インデックス名の指定
>
> ```
> package org.hibernate.example;
>
> @Indexed
> public class Status { ... }
>
> @Indexed(index="Rules")
> public class Rule { ... }
>
> @Indexed(index="Actions")
> public class Action { ... }
> ```

例**23.2** ディレクトリープロバイダーの設定

```
hibernate.search.default.directory_provider = filesystem
hibernate.search.default.indexBase=/usr/lucene/indexes
hibernate.search.Rules.directory_provider = ram
hibernate.search.Actions.directory_provider =
com.acme.hibernate.CustomDirectoryProvider
```

注記

上記の設定スキームを使用すると、ディレクトリープロバイダーやベースディレクトリーなどの一般的なルールを簡単に定義でき、これらのデフォルトをインデックスごとに後で上書きできます。

ディレクトリープロバイダーおよびそのプロパティー

ram

None

Filesystem

ファイルシステムベースのディレクトリー使用されるディレクトリーは **<indexBase> /< indexName >** です。

- **indexBase :** ベースディレクトリー
- **indexName: @Indexed.index** を上書きします（シャードインデックスに役立ちます）。
- **locking_strategy : optional**、を参照してください。 「**LockFactory** 設定」
- **filesystem_access_type:** は、この **DirectoryProvider** で使用される **FSDirectory** 実装の正確なタイプを判断できるようにします。許可される値は **auto** （デフォルト値、非 **Windows** システムの **NIOFSDirectory**、**Windows** の **SimpleFSDirectory** ）、**Simple(SimpleFSDirectory)**、**nio (NIOFSDirectory)**、**mmap (MMapDirectory)**です。この設定を変更する前に、これらの ディレクトリー 実装の **Javadocs** を参照してくださ

い。**NIOFSDirectory** または **MMapDirectory** でも、パフォーマンスが大幅に向上しますが、問題も向上します。

filesystem-master

ファイルシステムベースのディレクトリー**filesystem** と類似しています。また、定期的にインデックスをソースディレクトリー**(**コピーディレクトリー**)** にコピーします。

更新期間に推奨される値は **(**最低 **50%)**、情報をコピーする時間 **(**デフォルトは **3600** 秒 **- 60** 分**)** です。

コピーは、増分コピーメカニズムをベースにしているため、コピーの平均時間が短縮されることに注意してください。

DirectoryProvider は通常、**JMS** バックエンドクラスターのマスターノードで使用されます。

buffer_size_on_copy optimum は、オペレーティングシステムと利用可能な **RAM** によって異なります。ほとんどのユーザーは、**16 MB** から **64 MB** までの値を使用して適切な結果を報告しました。

- **indexBase:** ベースディレクトリー
- **indexName: @Indexed.index** を上書きします（シャードインデックスに役立ちます）。
- **sourceBase:** ソース（コピー）ベースディレクトリー。
- **source:** ソースディレクトリーのサフィックス（デフォルトは **@Indexed.index**）。実際のソースディレクトリー名は **< sourceBase>/<source>** です。
- 更新**:** 更新期間を秒単位で行います（コピーは **refresh** 秒ごとに行われます）。以下の **refresh** 期間が経過してもコピーが進行中であれば、**2** 番目のコピー操作が省略されます。
- **buffer_size_on_copy:** 単一の低レベルのコピー命令で移動する **MegaBytes** の量。デ

フォルトは **16MB** です。

- **locking_strategy : optional**、を参照してください。 「**LockFactory** 設定」

- **filesystem_access_type:** は、この **DirectoryProvider** で使用される **FSDirectory** 実装の正確なタイプを判断できるようにします。許可される値は **auto** （デフォルト値、非 **Windows** システムの **NIOFSDirectory**、**Windows** の **SimpleFSDirectory** ）、**Simple(SimpleFSDirectory)**、**nio (NIOFSDirectory)**、**mmap (MMapDirectory)**です。この設定を変更する前に、これらの ディレクトリー 実装の **Javadocs** を参照してください。**NIOFSDirectory** または **MMapDirectory** はパフォーマンスを大幅に向上させる可能性がありますが、問題もあるため、認識する必要があります。

filesystem-slave

ファイルシステムベースのディレクトリー**filesystem** と似ていますが、マスターバージョン **(**ソース**)** を定期的に取得します。ロックおよび一貫性のない検索結果を避けるため、**2** つのローカルコピーが維持されます。

更新期間に推奨される値は **(**最低 **50%)**、情報をコピーする時間 **(**デフォルトは **3600** 秒 **- 60** 分**)** です。

コピーは、増分コピーメカニズムをベースにしているため、コピーの平均時間が短縮されることに注意してください。**refresh** の期間が経過してもコピーが進行中であれば、**2** 番目のコピー操作が省略されます。

DirectoryProvider は通常、**JMS** バックエンドを使用するスレーブノードで使用されます。

buffer_size_on_copy optimum は、オペレーティングシステムと利用可能な **RAM** によって異なります。ほとんどのユーザーは、**16 MB** から **64 MB** までの値を使用して適切な結果を報告しました。

- **indexBase:** ベースディレクトリー

- **indexName: @Indexed.index** を上書きします（シャードインデックスに役立ちます）。

- ***sourceBase: Source(copy)**ベースディレクトリー。*

- ***Source : ソース ディレクトリーのサフィックス（デフォルトは **@Indexed.index** です）。実際のソースディレクトリー名は **< sourceBase>/<source>** です。*

- *更新: 更新期間を秒単位で行います（コピーは更新の秒数ごとに行われます）。*

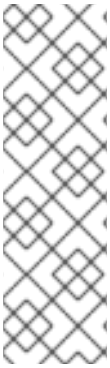
- ***buffer_size_on_copy:** 単一の低レベルのコピー命令で移動する **MegaBytes** の量。デフォルトは **16MB** です。*

- ***locking_strategy : optional**、を参照してください。 「**LockFactory** 設定」*

- ***retry_marker_lookup : optional**。デフォルトは **0** です。**Hibernate Search** がソースディレクトリーのマーカーファイルをチェックする回数を定義します。試行ごとに **5** 秒待機します。*

- ***retry_initialize_period :** オプション。再試行初期化機能を有効にするには、整数値を秒単位で設定します。スレーブがマスターインデックスを見つけられない場合、アプリケーションが起動するのを妨げずにバックグラウンドで見つかったまで再試行します。インデックスが初期化される前に実行された完全な**Text** クエリーはブロックされませんが、空の結果が返されます。オプションを有効にしない、または明示的にゼロに設定すると、再試行タイマーのスケジュール設定ではなく、例外により失敗します。無効なインデックスなしでアプリケーションが起動しないようにし、初期化のタイムアウトを制御するには、代わりに **retry_marker_lookup** を参照してください。*

- ***filesystem_access_type:** は、この **DirectoryProvider** で使用される **FSDirectory** 実装の正確なタイプを判断できるようにします。許可される値は **auto** （デフォルト値、非 **Windows** システムの **NIOFSDirectory**、**Windows** の **SimpleFSDirectory** ）、**Simple(SimpleFSDirectory)**、**nio (NIOFSDirectory)**、**mmap (MMapDirectory)**です。この設定を変更する前に、これらの ディレクトリー 実装の **Javadocs** を参照してください。**NIOFSDirectory** または **MMapDirectory** により、パフォーマンスが大幅に向上しますが、問題を認識する必要もあります。*



注記

*組み込みディレクトリープロバイダーがニーズに適さない場合は、**org.hibernate.store.DirectoryProvider** インターフェースを実装することで独自のディレクトリープロバイダーを作成できます。この場合、プロバイダーの完全修飾クラス名を **directory_provider** プロパティーに渡します。プレフィックス **hibernate.search.<indexname>** を使用して追加のプロパティーを渡すことができます。*

バグの報告

23.2.4. シャード化インデックス

*場合によっては、特定のエンティティーのインデックス付きデータを複数の **Lucene** インデックスに分割 (シャード) すると役に立つことがあります。*

警告

シャード化は、欠点が短所を上回った場合にのみ実装する必要があります。単一の検索用にシャードをすべて開く必要があるため、シャード化されたインデックスの検索は一般的に遅くなります。

シャード化のユースケースを以下に示します。

- *単一のインデックスが大きいと、インデックスの更新時間は遅くなります。*
- *一般的な検索は、データが顧客、地域、またはアプリケーションによってセグメント化された場合など、インデックスのサブセットのみに一致します。*

*デフォルトでは、シャードの数が設定されていないとシャード化は有効になりません。これには、**hibernate.search.<indexName>.sharding_strategy.nbr_of_shards** プロパティーを使用します。*

*例**23.3** インデックスシャード化の有効化*

この例では、シャードが 5 個有効になります。

```
hibernate.search.<indexName>.sharding_strategy.nbr_of_shards = 5
```

データをサブインデックスに分割することが、**IndexShardingStrategy** です。デフォルトのシャーディングストラテジーは、**ID** 文字列表現のハッシュ値（**FieldBridge**によって生成される）に従ってデータを分割します。これにより、シャードが大幅に分散されます。カスタム **IndexShardingStrategy** を実装して、デフォルトのストラテジーを置き換えることができます。カスタムストラテジーを使用するには、**hibernate.search.<indexName>.sharding_strategy** プロパティーを設定する必要があります。

例**23.4** カスタムシャード化ストラテジーの指定

```
hibernate.search.<indexName>.sharding_strategy = my.shardingstrategy.Implementation
```

IndexShardingStrategy プロパティーでは、クエリーを実行するシャードを選択して検索を最適化することもできます。フィルターシャードストラテジーをアクティベートすることで、クエリーに応答するために使用されるシャードのサブセット**(IndexShardingStrategy.getIndexManagersForQuery)**を選択し、クエリーの実行時間を短縮できます。

各シャードには独立した **IndexManager** があるため、異なるディレクトリープロバイダーおよびバックエンド設定を使用するように設定できます。例**23.5**「エンティティーアニメーション設定のシャーディング」 の **Animal** エンティティーの **IndexManager** インデックス名は **Animal.0** から **Animal.4** までです。つまり、各シャードには独自のインデックスの名前の後に **.**（ドット）およびそのインデックス番号が続きます。

例**23.5** エンティティーアニメーション設定のシャーディング

```
hibernate.search.default.indexBase = /usr/lucene/indexes
hibernate.search.Animal.sharding_strategy.nbr_of_shards = 5
hibernate.search.Animal.directory_provider = filesystem
hibernate.search.Animal.0.indexName = Animal00
hibernate.search.Animal.3.indexBase = /usr/lucene/sharded
hibernate.search.Animal.3.indexName = Animal03
```

例**23.5**「エンティティーアニメーション設定のシャーディング」 では、設定では、デフォルトの **id** 文字列ハッシュストラテジーを使用し、**Anonim al** インデックスを **5** つのサブインデックスにシャード化します。すべてのサブインデックスはファイルシステムのインスタンスで、各サブインデックスが保存されるディレクトリーは、以下のようになります。

- サブインデックス **0** の場合： **/usr/lucene/indexes/Animal00** （共有 **indexBase** が上書きされた **indexName**）の場合

- **for sub-index 1: /usr/lucene/indexes/Animal.1 (shared indexBase, default indexName)**

- **for sub-index 2: /usr/lucene/indexes/Animal.2 (shared indexBase, default indexName)**

- サブインデックス **3** の場合： **/usr/lucene/shared/Animal03 (overridden indexBase, overridden indexName)**

- サブインデックス **4** の場合： **/usr/lucene/indexes/Animal.4** （共有 **indexBase**、デフォルトの **indexName**）

IndexShardingStrategy を実装する場合、このフィールドを使用してシャーディングの選択を判別できます。**deletion**、**purge**、**purgeAll** などの操作を処理するには、すべてのフィールド値またはプライマリー識別子を読み取れずにインデックスを返す必要があることもあります。このような場合、すべてのインデックスが返されるため、削除操作は、削除されるドキュメントが含まれる可能性のあるすべてのインデックスに伝播されます。

バグの報告

23.2.5. ワーカー設定

workder 設定では、**Hibernate Search** が **Lucene** と対話する方法を詳細化することができます。複数のアーキテクチャーコンポーネントと可能な拡張ポイントが存在します。詳しく見てみましょう。

最初に ワーカーがあります。ワーカー インターフェースの実装は、すべてのエンティティーの変更を受け取り、コンテキストでキューイングし、コンテキストが終了するとそれらを適用します。特に **ORM** との接続で最も直感的なコンテキストはトランザクションです。このため、**Hibernate Search** はデフォルトで **TransactionalWorker** を使用してトランザクションごとのすべての変更のスコープを設定します。ただし、コンテキストがエンティティーの変更数や他のアプリケーション**(lifecycle)**イベントの数に依存するシナリオを考えてみます。このため、表**23.1**「スコープ設定」 に示されるように、**Worker** 実装は設定可能です。

表**23.1** スコープ設定

プロパティー	説明

hibernate.search.worker.scope	使用する **ワーカー** 実装の完全修飾クラス名。このプロパティーが設定されていない場合、空または **トランザクション** の場合はデフォルトの **TransactionalWorker** が使用されます。
hibernate.search.worker.*	接頭辞 **hibernate.search.worker** が付いたすべての設定プロパティーは初期化中にワーカーに渡されます。これにより、カスタムのワーカー固有のパラメーターを追加できます。
hibernate.search.worker.batch_size	コンテキストごとにバッチ処理されるインデックス操作の最大数を定義します。制限に達すると、コンテキストが終了していなくてもインデックスがトリガーされます。このプロパティーは、**Worker** 実装によってキューに追加された作業を BatchedQueueingProcessor に委譲する場合にのみ機能します（**TransactionalWorker** が実行する動作です）。

コンテキストが終了すると、インデックスの変更を準備して適用します。これは、新規スレッド内で同期または非同期に実行できます。同期更新には、常にインデックスがデータベースと同期しているという利点があります。一方、非同期のアップデートは、ユーザーの応答時間を最小限に抑えるのに役立ちます。欠点は、データベースとインデックスの状態間で不一致が生じる可能性があることです。表 23.2「実行設定」 に記載されている設定オプションを確認しましょう。

注記

*以下のオプションはインデックスごとに異なる場合があります。実際には、**indexName** 接頭辞が必要になるか、**default** を使用してすべてのインデックスのデフォルト値を設定する必要があります。*

表23.2 実行設定

プロパティー	説明
hibernate.search.<indexName>.worker.execution	**sync**: 同期実行 (デフォルト) **async**: 非同期実行
hibernate.search.<indexName>.worker.thread_pool.size	バックエンドは、スレッドプールを使用して、同じトランザクションコンテキスト（またはバッチ）からの更新を並行して適用することができます。デフォルト値は 1 です。トランザクションごとに多数の操作がある場合は、大きな値を試すことができます。

hibernate.search.<indexName>.worker.buffer_queue.max	スレッドポーリングが不足している場合、ワークキューの最大数を定義します。非同期実行のみに便利です。デフォルトは infinite です。制限に達すると、ワークはメインスレッドによって行われます。

*これまでは、実行モードに関係なく、すべての作業が同じ仮想マシン(VM)内で実行されます。単一仮想マシンの作業合計量は変更されていません。常に、より適切なアプローチ (つまり委任) があります。**hibernate.search.default.worker.backend** を設定して、インデックスを別のサーバーに送信することができます。表23.3「バックエンドの設定」 を参照してください。このオプションも、インデックスごとに異なる方法で設定できます。*

表23.3 バックエンドの設定

プロパティー	説明
hibernate.search.<indexName>.worker.backend	**lucene**: 同じ仮想マシンでインデックスの更新を実行するデフォルトのバックエンド。プロパティーが定義されていないか、または空の場合に使用されます。 **JMS**: JMS バックエンド。インデックスの更新は JMS キューに送信され、インデックスマスターによって処理されます。追加の設定オプションは 表23.4「JMS バックエンドの設定」 で、この設定の詳細な説明は 「JMS マスター/ スレーブバックエンド」 を参照してください。 **BlackHole**: 主にテスト/開発者の設定で、すべてのインデックス作業を無視します。 **BackendQueueProcessor** を実装するクラスの完全修飾名を指定することもできます。これにより、独自の通信層を実装することができます。この実装は実行時に **Runnable** インスタンスを返し、インデックスが機能するようにします。

表23.4 JMS バックエンドの設定

プロパティー	説明
hibernate.search.<indexName>.worker.jndi.*	JNDI プロパティーを定義して InitialContext を開始します（必要な場合）。JNDI は JMS バックエンドによってのみ使用されます。
hibernate.search.<indexName>.worker.jms.connection_factory	JMS バックエンドには必須です。JMS 接続ファクトリーを検索する JNDI 名を定義します (Red Hat JBoss Enterprise Application Platform では、/**ConnectionFactory** がデフォルト)。
hibernate.search.<indexName>.worker.jms.queue	JMS バックエンドには必須です。JMS キューを検索する JNDI 名を定義します。キューはワークメッセージをポストするために使用されます。

警告

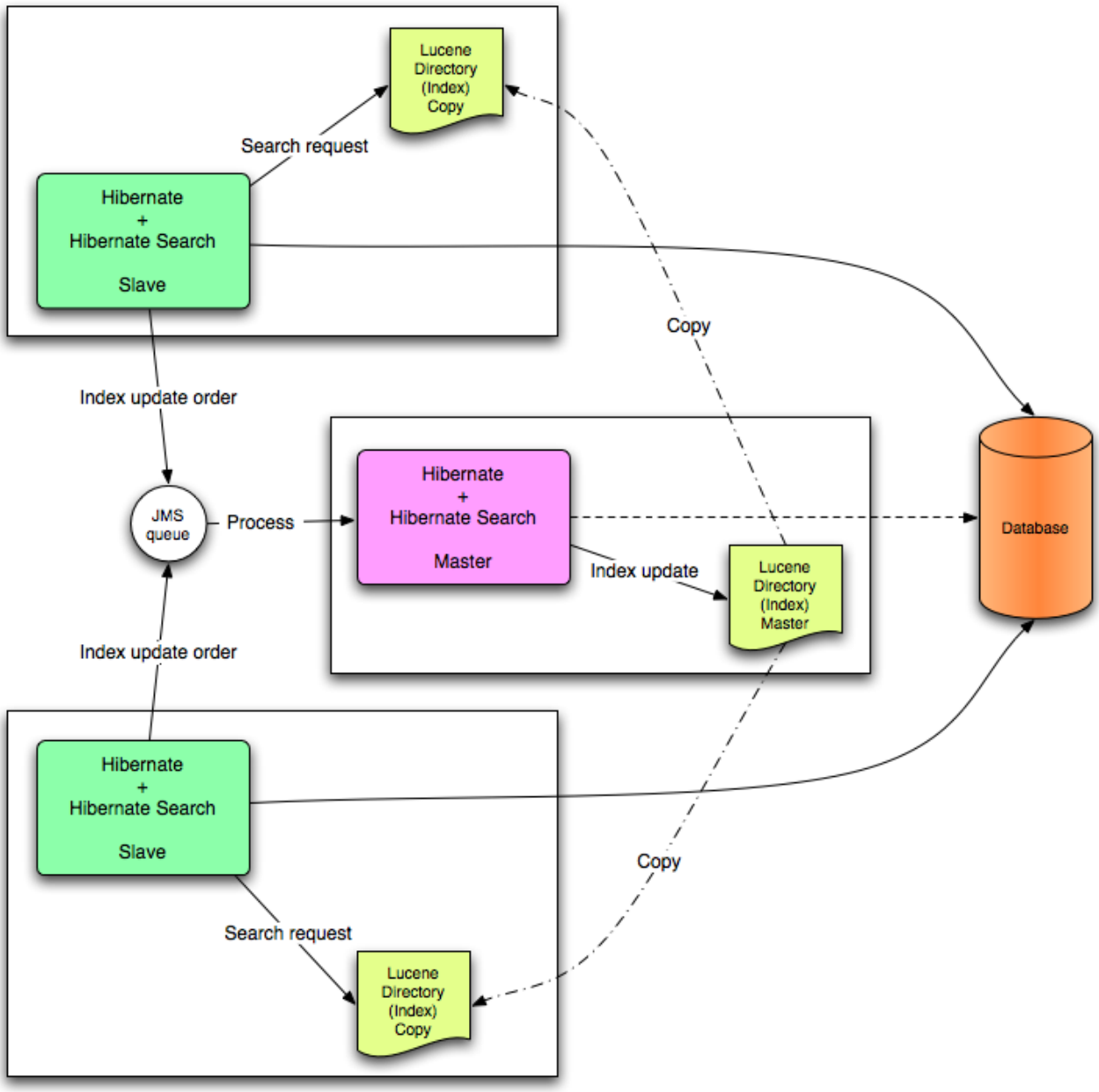
おそらく、表示されるプロパティーの一部は関連付けられるため、プロパティー値のすべての組み合わせが適切であるとは限りません。実際には、機能以外の設定を行うことができます。これは、特に、ここに示されるインターフェースの独自の実装を提供する場合が該当します。独自の ***Worker*** *または* ***BackendQueueProcessor*** *実装を作成する前に、既存のコードを調査してください。*

バグの報告

23.2.5.1. JMS マスター/スレーブバックエンド

本セクションでは、 ***Master/Slave Hibernate Search*** *アーキテクチャーを設定する方法を説明します。*

図23.3 JMS バックエンドの設定



バグの報告

23.2.5.2. スレーブノード

*すべてのインデックス更新操作は **JMS** キューに送信されます。インデックスクエリー操作は、ローカルインデックスコピーで実行されます。*

例23.6 JMS スレーブの設定

```
### slave configuration

## DirectoryProvider
# (remote) master location
```

```
hibernate.search.default.sourceBase = /mnt/mastervolume/lucenedirs/mastercopy

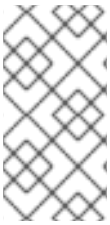
# local copy location
hibernate.search.default.indexBase = /Users/prod/lucenedirs

# refresh every half hour
hibernate.search.default.refresh = 1800

# appropriate directory provider
hibernate.search.default.directory_provider = filesystem-slave

## Backend configuration
hibernate.search.default.worker.backend = jms
hibernate.search.default.worker.jms.connection_factory = /ConnectionFactory
hibernate.search.default.worker.jms.queue = queue/hibernatesearch
#optional jndi configuration (check your JMS provider for more information)

## Optional asynchronous execution strategy
# hibernate.search.default.worker.execution = async
# hibernate.search.default.worker.thread_pool.size = 2
# hibernate.search.default.worker.buffer_queue.max = 50
```



注記

ファイルシステムローカルコピーは、より高速な検索結果を得るために推奨されます。

バグの報告

23.2.5.3. マスターノード

*すべてのインデックス更新操作は **JMS** キューから取得され、実行されます。マスターインデックスは定期的にコピーされます。*

例23.7 JMS マスターの設定

```
### master configuration

## DirectoryProvider
# (remote) master location where information is copied to
hibernate.search.default.sourceBase = /mnt/mastervolume/lucenedirs/mastercopy

# local master location
hibernate.search.default.indexBase = /Users/prod/lucenedirs

# refresh every half hour
hibernate.search.default.refresh = 1800
```

```
# appropriate directory provider
hibernate.search.default.directory_provider = filesystem-master

## Backend configuration
#Backend is the default lucene one
```

Hibernate Search フレームワーク設定の他に、**JMS** を介してインデックスが動作するよう、メッセージ駆動 **Bean** を作成および設定する必要があります。

例**23.8** インデクシングキューの処理中のメッセージ駆動型 **Bean**

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName="destinationType",
                              propertyValue="javax.jms.Queue"),
    @ActivationConfigProperty(propertyName="destination",
                              propertyValue="queue/hibernatesearch"),
    @ActivationConfigProperty(propertyName="DLQMaxResent", propertyValue="1")
  } )
public class MDBSearchController extends AbstractJMSHibernateSearchController
                         implements MessageListener {
   @PersistenceContext EntityManager em;

   //method retrieving the appropriate session
   protected Session getSession() {
      return (Session) em.getDelegate();
   }

   //potentially close the session opened in #getSession(), not needed here
   protected void cleanSessionIfNeeded(Session session)
   }
}
```

この例では、**Hibernate Search** ソースコードで利用可能な抽象 **JMS** コントローラークラスを継承し、**MDB** を実装します。この実装は例として提供され、**Java EE** 以外のメッセージ駆動型 **Bean** を使用するよう調整できます。**getSession()** および **cleanSessionIfNeeded()** の詳細は、「**AbstractJMSHibernateSearchController 's javadoc**」を参照してください。

バグの報告

23.2.6. Lucene インデックスのチューニング

23.2.6.1. Lucene インデックスのパフォーマンスチューニング

Hibernate Search は、**mergeFactor**、**maxMergeDocs**、**maxBufferedDocs** などの基礎となる

Lucene IndexWriter に渡されるパラメーターセットを指定して、Lucene インデックスのパフォーマンスを調整するために使用されます。これらのパラメーターは、インデックスベースまたはシャードごとに、すべてのインデックスに適用されるデフォルト値として指定します。

各種ユースケースに合わせて調整できる低レベルの IndexWriter 設定がいくつかあります。これらのパラメーターは、indexwriter キーワードでグループ化されます。

```
hibernate.search.[default|<indexname>].indexwriter.<parameter_name>
```

特定のシャード設定の indexwriter 値に値が設定されていないと、Hibernate Search は index セクションをチェックしてから default セクションをチェックします。

以下の表の設定により、これらの設定は Animal インデックスの 2 つ目のシャードに適用されます。

- *max_merge_docs = 10*
- *merge_factor = 20*
- *ram_buffer_size = 64MB*
- *term_index_interval = Lucene default*

他のすべての値は、Lucene で定義されたデフォルトを使用します。

デフォルトでは、すべての値は Lucene の独自のデフォルトのままにします。表23.5「インデックス化のパフォーマンスおよび動作プロパティーの一覧」 に記載されている値は、使用している Lucene のバージョンにより異なります。上記の値はバージョン 2.4 に相対的です。Lucene インデックスのパフォーマンスに関する詳細は、Lucene のドキュメントを参照してください。

注記

***Hibernate Search** の以前のバージョンには **batch** および **transaction** プロパティーの概念がありました。バックエンドは常に同じ設定を使用して機能するため、これは当てはまりません。*

*表**23.5** インデックス化のパフォーマンスおよび動作プロパティーの一覧*

プロパティー	説明	デフォルト値
hibernate.search.[default|<indexname>].exclusive_index_use	他のプロセスが同じインデックスに書き込む必要がない場合は **true** に設定します。これにより、Hibernate Search はインデックスの排他モードで動作し、インデックスへの変更を書き込む際にパフォーマンスが向上します。	**true** (パフォーマンスの向上、シャットダウン時のみロックされる)
hibernate.search.[default|<indexname>].max_queue_length	各インデックスには、インデックスに適用される更新が含まれる個別の「pipeline」があります。このキューが満杯になると、ブロック操作になります。**worker.execution** が **async** として設定されていない限り、この設定を設定するとかなり意味がありません。	**1000**
hibernate.search.[default|<indexname>].indexwriter.max_buffered_delete_terms	バッファーインメモリー削除の条件が適用され、フラッシュされるまでに最低限必要な削除用語を決定します。時点でバッファーされたドキュメントがメモリーにある場合、ドキュメントはマージされ、新しいセグメントが作成されます。	Disabled (RAM 使用率によるフラッシュ)
hibernate.search.[default|<indexname>].indexwriter.max_buffered_docs	インデックス作成中にメモリーでバッファーされたドキュメントの量を制御します。RAM が大きいほど消費されます。	Disabled (RAM 使用率によるフラッシュ)
hibernate.search.[default|<indexname>].indexwriter.max_merge_docs	セグメント内で許容されるドキュメントの最大数を定義します。値を小さくすると、頻繁に変更されるインデックスでのパフォーマンスが向上します。値を大きくすると、インデックスが頻繁に変更されない場合に検索パフォーマンスが向上します。	Unlimited (Integer.MAX_VALUE)

プロパティー	説明	デフォルト値
hibernate.search.[default|<indexname>].indexwriter.merge_factor	セグメントマージの頻度とサイズを制御します。 挿入時にセグメントインデックスをマージする頻度を決定します。値を小さくすると、インデックス処理時に使用されるRAM が少なくなり、最適化されていないインデックスの検索速度が速くなりますが、インデックスの速度は遅くなります。値が大きいと、インデックス時により多くの RAM が使用され、最適化されていないインデックスの検索速度が遅くなるため、インデックスがより速くなります。そのため、大きな値 (> 10) はバッチインデックスの作成に最も適しており、対話的に維持されるインデックスに小さい値 (< 10) が適しています。この値は 2 未満にしないでください。	10
hibernate.search.[default|<indexname>].indexwriter.merge_min_size	セグメントマージの頻度とサイズを制御します。 このサイズより小さいセグメント (MB 単位) は常に次のセグメントマージ操作の対象となります。 この値を大きく設定すると、マージ操作が高価になり、頻度が低くなる可能性があります。 See also **org.apache.lucene.index.LogDocMergePolicy.minMergeSize**.	0 MB (実際 ~1K)
hibernate.search.[default|<indexname>].indexwriter.merge_max_size	セグメントマージの頻度とサイズを制御します。 このサイズより大きいセグメント (MB 単位) は、大きなセグメントにマージされることはありません。 これにより、メモリー要件が減り、一部のマージ操作が最適な検索速度で回避されます。インデックスの最適化時にこの値は無視されます。 See also **org.apache.lucene.index.LogDocMergePolicy.maxMergeSize**.	無制限

プロパティー	説明	デフォルト値
hibernate.search.[default|<indexname>].indexwriter.merge_max_optimize_size	セグメントマージの頻度とサイズを制御します。 このサイズよりも大きい (MB 単位) セグメントは、インデックスの最適化中にも大きなセグメントではマージされません (**merge_max_size** 設定も参照)。 **org.apache.lucene.index.LogDocMergePolicy** に適用されます。**maxMergeSizeForOptimize**.	無制限
hibernate.search.[default|<indexname>].indexwriter.merge_calibrate_by_deletes	セグメントマージの頻度とサイズを制御します。 **false** に設定すると、マージポリシーを見積もるときに削除されたドキュメントを考慮しません。 **org.apache.lucene.index.LogMergePolicy** に適用されます。**calibrateSizeByDeletes**.	**true**
hibernate.search.[default|<indexname>].indexwriter.ram_buffer_size	ドキュメントバッファー専用の RAM の容量 (MB 単位) を制御します。Max_buffered_docs を一緒に使用すると、いずれのイベントに対してもフラッシュが発生します。 通常、インデックスの高速化には、ドキュメント数ではなく、RAM の使用状況によってフラッシュされ、可能な限り大きな RAM バッファーとして使用することが推奨されます。	16 MB
hibernate.search.[default|<indexname>].indexwriter.term_index_interval	エキスパート： インデックス用語の間隔を設定します。 大きな値を設定すると、IndexReader が使用するメモリーは少なくなりますが、ランダムアクセスの速度が遅くなります。値を小さくすると、IndexReader によりより多くのメモリーが使用され、用語へのランダムアクセスが速くなります。詳細は、Lucene ドキュメントを参照してください。	128

プロパティー	説明	デフォルト値
hibernate.search.[default|<indexname>].indexwriter.use_compound_file	複合ファイル形式を使用すると、使用するファイル記述子が少なくなります。欠点は、インデックス作成にかかる時間と一時ディスク容量が多いことです。インデックス処理時間を短縮するために、このパラメーターを **false** に設定できますが、**mergeFactor** が大きくなるとファイル記述子が不足する可能性があります。 boolean パラメーター。「**true**」または「**false**」を使用します。このオプションのデフォルト値は **true** です。	true
hibernate.search.enable_dirty_check	すべてのエンティティーの変更に Lucene インデックスの更新が必要であるわけではありません。更新されたエンティティープロパティー（ダーティープロパティー）がすべてインデックス化されていない場合、Hibernate Search はインデックス変更プロセスを省略します。 各更新イベントで呼び出す必要があるカスタムの **FieldBridge**s を使用する場合は、このオプションを無効にします（フィールドブリッジが設定されているプロパティーは変更されません）。 この最適化は、**@ClassBridge** または **@DynamicBoost**を使用するクラスには適用されません。 boolean パラメーター。「**true**」または「**false**」を使用します。このオプションのデフォルト値は **true** です。	true

警告

ブラックリストバックエンドは、 インデックスのボトルネックを特定するためのツールとしてのみ、実稼働環境で使用することは意図されていません。

バグの報告

23.2.6.2. Lucene IndexWriter

各種ユースケースに合わせて調整できる低レベルの **IndexWriter** 設定がいくつかあります。これらのパラメーターは、**indexwriter** キーワードでグループ化されます。

```
default.<indexname>.indexwriter.<parameter_name>
```

シャード設定で **indexwriter** の値が設定されていない場合、**Hibernate Search** は **index** セクションを確認してから **default** セクションを確認します。

バグの報告

23.2.6.3. パフォーマンスオプションの設定

以下の設定では、これらの設定は **Animal** インデックスの **2** つ目のシャードに適用されます。

例**23.9** パフォーマンスオプションの設定例

```
default.Animals.2.indexwriter.max_merge_docs = 10
default.Animals.2.indexwriter.merge_factor = 20
default.Animals.2.indexwriter.term_index_interval = default
default.indexwriter.max_merge_docs = 100
default.indexwriter.ram_buffer_size = 64
```

- **max_merge_docs = 10**
- **merge_factor = 20**
- **ram_buffer_size = 64MB**
- **term_index_interval = Lucene default**

他のすべての値は、**Lucene** で定義されたデフォルトを使用します。

Lucene のデフォルト値は、Hibernate Search のデフォルト設定です。したがって、以下の表に記載する値は、使用される Lucene のバージョンによって異なります。上記の値はバージョン 2.4 に相対的です。Lucene インデックスのパフォーマンスに関する詳細は、Lucene のドキュメントを参照してください。

注記

バックエンドは常に同じ設定を使用して動作します。

表23.6 インデックス化のパフォーマンスおよび動作プロパティーの一覧

プロパティー	説明	デフォルト値
default.<indexname>.exclusive_index_use	他のプロセスが同じインデックスに書き込む必要がない場合は **true** に設定します。これにより、Hibernate Search はインデックスの排他モードで動作し、インデックスへの変更を書き込む際にパフォーマンスが向上します。	**true** (パフォーマンスの向上、シャットダウン時のみロックされる)
default.<indexname>.max_queue_length	各インデックスには、インデックスに適用される更新が含まれる個別の「pipeline」があります。このキューが満杯になると、ブロック操作になります。**worker.execution** が **async** として設定されていない限り、この設定を設定するとかなり意味がありません。	**1000**
default.<indexname>.indexwriter.max_buffered_delete_terms	バッファーインメモリー削除の条件が適用され、フラッシュされるまでに最低限必要な削除用語を決定します。時点でバッファーされたドキュメントがメモリーにある場合、ドキュメントはマージされ、新しいセグメントが作成されます。	Disabled (RAM 使用率によるフラッシュ)
default.<indexname>.indexwriter.max_buffered_docs	インデックス作成中にメモリーでバッファーされたドキュメントの量を制御します。RAM が大きいほど消費されます。	Disabled (RAM 使用率によるフラッシュ)
default.<indexname>.indexwriter.max_merge_docs	セグメント内で許容されるドキュメントの最大数を定義します。値を小さくすると、頻繁に変更されるインデックスでのパフォーマンスが向上します。値を大きくすると、インデックスが頻繁に変更されない場合に検索パフォーマンスが向上します。	Unlimited (Integer.MAX_VALUE)

プロパティー	説明	デフォルト値
default.<indexname>.indexwriter.merge_factor	セグメントマージの頻度とサイズを制御します。 挿入時にセグメントインデックスをマージする頻度を決定します。値を小さくすると、インデックス処理時に使用されるRAM が少なくなり、最適化されていないインデックスの検索速度が速くなりますが、インデックスの速度は遅くなります。値が大きいと、インデックス時により多くの RAM が使用され、最適化されていないインデックスの検索速度が遅くなるため、インデックスがより速くなります。そのため、大きな値 (> 10) はバッチインデックスの作成に最も適しており、対話的に維持されるインデックスに小さい値 (< 10) が適しています。この値は 2 未満にしないでください。	10
default.<indexname>.indexwriter.merge_min_size	セグメントマージの頻度とサイズを制御します。 このサイズより小さいセグメント (MB 単位) は常に次のセグメントマージ操作の対象となります。 この値を大きく設定すると、マージ操作が高価になり、頻度が低くなる可能性があります。 See also **org.apache.lucene.index.LogDocMergePolicy.minMergeSize**.	0 MB (実際 ~1K)
default.<indexname>.indexwriter.merge_max_size	セグメントマージの頻度とサイズを制御します。 このサイズより大きいセグメント (MB 単位) は、大きなセグメントにマージされることはありません。 これにより、メモリー要件が減り、一部のマージ操作が最適な検索速度で回避されます。インデックスの最適化時にこの値は無視されます。 See also **org.apache.lucene.index.LogDocMergePolicy.maxMergeSize**.	無制限

プロパティー	説明	デフォルト値
default.<indexname>.indexwriter.merge_max_optimize_size	セグメントマージの頻度とサイズを制御します。 このサイズよりも大きい (MB 単位) セグメントは、インデックスの最適化中にも大きなセグメントではマージされません (**merge_max_size** 設定も参照)。 **org.apache.lucene.index.LogDocMergePolicy** に適用されます。**maxMergeSizeForOptimize**.	無制限
default.<indexname>.indexwriter.merge_calibrate_by_deletes	セグメントマージの頻度とサイズを制御します。 **false** に設定すると、マージポリシーを見積もるときに削除されたドキュメントを考慮しません。 **org.apache.lucene.index.LogMergePolicy** に適用されます。**calibrateSizeByDeletes**.	**true**
default.<indexname>.indexwriter.ram_buffer_size	ドキュメントバッファー専用の RAM の容量 (MB 単位) を制御します。Max_buffered_docs を一緒に使用すると、いずれのイベントに対してもフラッシュが発生します。 通常、インデックスの高速化には、ドキュメント数ではなく、RAM の使用状況によってフラッシュされ、可能な限り大きな RAM バッファーとして使用することが推奨されます。	16 MB
default.<indexname>.indexwriter.term_index_interval	エキスパート： インデックス用語の間隔を設定します。 大きな値を設定すると、IndexReader が使用するメモリーは少なくなりますが、ランダムアクセスの速度が遅くなります。値を小さくすると、IndexReader によりより多くのメモリーが使用され、用語へのランダムアクセスが速くなります。詳細は、Lucene ドキュメントを参照してください。	128

プロパティー	説明	デフォルト値
default.<indexname>.indexwriter.use_compound_file	複合ファイル形式を使用すると、使用するファイル記述子が少なくなります。欠点は、インデックス作成にかかる時間と一時ディスク容量が多いことです。インデックス処理時間を短縮するために、このパラメーターを **false** に設定できますが、**mergeFactor** が大きくなるとファイル記述子が不足する可能性があります。 boolean パラメーター。「**true**」または「**false**」を使用します。このオプションのデフォルト値は **true** です。	true
default.enable_dirty_check	すべてのエンティティーの変更に Lucene インデックスの更新が必要であるわけではありません。更新されたエンティティープロパティー（ダーティープロパティー）がすべてインデックス化されていない場合、Hibernate Search はインデックス変更プロセスを省略します。 各更新イベントで呼び出す必要があるカスタムの **FieldBridge**s を使用する場合は、このオプションを無効にします（フィールドブリッジが設定されているプロパティーは変更されません）。 この最適化は、**@ClassBridge** または **@DynamicBoost**を使用するクラスには適用されません。 boolean パラメーター。「**true**」または「**false**」を使用します。このオプションのデフォルト値は **true** です。	true

バグの報告

23.2.6.4. インデックス速度の調整

*アーキテクチャーが許可する場合、インデックスの書き込み効率を向上させるために **default.exclusive_index_use=true** のままにします。*

*インデックスの速度を調整する場合は、最初にオブジェクトの読み込みの最適化に焦点を合わせ、次にインデックスプロセスのチューニングの基準として達成するタイミングを使用することが推奨されます。**blackhole** をワーカーバックエンドとして設定し、インデックスルーチンを開始します。このバックエンドは **Hibernate Search** を無効にしません。必要な変更セットがインデックスに生成されますが、インデックスにフラッシュせずに破棄します。**hibernate.search.indexing_strategy** を **manual** に設定するのとは対照的に、**blackhole** を使用すると、関連付けられたエンティティーも再度インデックス化されるため、データベースからより多くのデータを読み込む可能性があります。*

hibernate.search.[default|<indexname>].worker.backend blackhole

警告

blackhole バックエンドは 、インデックスのボトルネックを特定する診断ツールとしてのみ、実稼働環境で使用することは意図されていません。

バグの報告

23.2.6.5. コントロールセグメントサイズ

以下のオプションは、作成されるセグメントの最大サイズを設定します。

- **merge_max_size**
- **merge_max_optimize_size**
- **merge_calibrate_by_deletes**

例23.10 コントロールセグメントサイズ

```
//to be fairly confident no files grow above 15MB, use:
hibernate.search.default.indexwriter.ram_buffer_size = 10
hibernate.search.default.indexwriter.merge_max_optimize_size = 7
hibernate.search.default.indexwriter.merge_max_size = 7
```

セグメントを **2** つのセグメントを **1** つの大きなセグメントに統合するため、マージ操作をハード制限セグメントサイズの半分未満にするには **max_size** を設定します。

新しいセグメントのサイズは、最初に想定よりも大きくなる可能性がありますが、セグメントは **ram_buffer_size** よりもはるかに大きく作成されることはありません。このしきい値は予測としてチェックされます。

バグの報告

23.2.7. LockFactory 設定

Lucene ディレクトリーは、**Hibernate Search** によって管理される各インデックスの **LockingFactory** を使用してカスタムロックストラテジーで設定できます。

一部のロックストラテジーには、ファイルシステムレベルのロックが必要で、**RAM** ベースのインデックスで使用できます。このストラテジーを使用する場合、**IndexBase** 設定オプションは、ロックマーカーファイルを保存するファイルシステムの場所を参照するように指定する必要があります。

ロックファクトリーを選択するには、**hibernate.search.<index>.locking_strategy** オプションを **1** つのオプションを設定します。

- **simple**
- **native**
- **single**
- **none**

表**23.7** 利用可能な **LockFactory** 実装の一覧

name	クラス	説明

name	クラス	説明
simple	org.apache.lucene.store.SimpleFSLockFactory	Java の File API に基づく安全な実装では、マーカーファイルを作成してインデックスの使用をマークします。 何らかの理由でアプリケーションを強制終了する必要がある場合には、このファイルを削除してから再起動する必要があります。
native	org.apache.lucene.store.NativeFSLockFactory	**simple** と同様に、マーカーファイルを作成してインデックスの使用をマークします。しかし、これはネイティブの OS ファイルロックを使用しているため、JVM が終了してもロックはクリーンアップされます。 この実装では、NFS で既知の問題があるため、ネットワーク共有で使用しないでください。 **native** は、**filesystem**、**filesystem-master**、**filesystem-slave** ディレクトリープロバイダーのデフォルトの実装です。
single	org.apache.lucene.store.SingleInstanceLockFactory	この LockFactory はファイルマーカーを使用しませんが、メモリーに保持される Java オブジェクトロックであるため、インデックスが他のプロセスで共有されないことが確実な場合にのみ使用できます。 これは、**ram** ディレクトリープロバイダーのデフォルト実装です。
none	org.apache.lucene.store.NoLockFactory	このインデックスへの変更は、ロックによって調整されません。

以下は、ロックストラテジーの設定例です。

```
hibernate.search.default.locking_strategy = simple
hibernate.search.Animals.locking_strategy = native
hibernate.search.Books.locking_strategy = org.custom.components.MyLockingFactory
```

バグの報告

23.2.8. 例外処理の設定

***Hibernate Search** では、インデックス化プロセス中に例外の処理方法を設定できます。設定が指定されていない場合、デフォルトでは例外がログ出力に記録されます。以下のように例外ロギングメカニズムを明示的に宣言できます。*

```
hibernate.search.error_handler = log
```

デフォルトの例外処理は、同期インデックスと非同期インデックスの両方で行われます。**Hibernate Search** では、デフォルトのエラー処理の実装を上書きする簡単なメカニズムを利用できます。

独自の実装を提供するには、**handle(ErrorContext context)** メソッドを提供する **ErrorHandler** インターフェースを実装する必要があります。**ErrorContext** プライマリー **LuceneWork** インスタンス、基礎となる例外、およびプライマリー例外により処理できなかった後続の **LuceneWork** インスタンスへの参照を提供します。

```
public interface ErrorContext {
  List<LuceneWork> getFailingOperations();
  LuceneWork getOperationAtFault();
  Throwable getThrowable();
  boolean hasErrors();
}
```

このエラーハンドラーを **Hibernate Search** に登録するには、設定プロパティーで **ErrorHandler** 実装の完全修飾クラス名を宣言する必要があります。

```
hibernate.search.error_handler = CustomerErrorHandler
```

バグの報告

23.2.9. インデックス形式の互換性

現在、**Hibernate Search** には後方互換性の **API** またはツールが含まれておらず、アプリケーションを新しいバージョンに移植することができます。**API** は、インデックスの書き込みおよび検索に **Apache Lucene** を使用します。インデックス形式の更新が必要になる場合があります。この場合、**Lucene** が古い形式を読み取れない場合は、データのインデックスを再作成する必要があります。

警告

インデックス形式を更新する前にインデックスをバックアップします。

***Hibernate Search** は **hibernate.search.lucene_version** 設定プロパティーを公開します。このプロパティーは、旧バージョンの **Lucene** で定義されたように、**Analyzly** クラスおよびその他の **Lucene** クラスが動作に準拠するように指示します。**lucene-core.jar** に含まれる **org.apache.lucene.util.Version** も参照してください。このオプションが指定されていない場合、**Hibernate Search** は **Lucene** にバージョンのデフォルトを使用するように指示します。使用されるバージョンは、アップグレード時に自動的に変更されないように設定に明示的に定義することが推奨されます。アップグレード後、必要に応じて設定値を明示的に更新できます。*

> *例**23.11 Lucene 3.0** で作成されたインデックスと **force Analyzers** の互換性。*
>
> ```
> hibernate.search.lucene_version = LUCENE_30
> ```

*設定された **SearchFactory** はグローバルで、関連するパラメーターが含まれるすべての **Lucene API** に影響します。**Lucene** を使用し、**Hibernate Search** がバイパスされている場合は、一貫した結果を得るために同じ値をこれに適用します。*

バグの報告

23.2.10. Hibernate Search の無効化

***Hibernate Search** は必要に応じて部分的にまたは完全に無効にできます。**Hibernate Search** のインデックスは、たとえばインデックスが読み取り専用の場合や、自動ではなく手動でインデックスを実行したい場合など、無効にすることができます。**Hibernate Search** を完全に無効にして、インデックス設定や検索を阻止することもできます。*

インデックスの無効化

***Hibernate Search** インデックスを無効にするには、**indexing_strategy** 設定オプションを **manual** に変更して **JBoss EAP** を再起動します。*

```
hibernate.search.indexing_strategy = manual
```

Hibernate Search を完全に無効にする

Hibernate Search を完全に無効にするには、**autoregister_listeners** 設定オプションを**false** に変更してすべてのリスナーを無効にし、**JBoss EAP** を再起動します。

```
hibernate.search.autoregister_listeners = false
```

バグの報告

23.3. モニタリング

23.3.1. モニタリング

Hibernate Search は、**SearchFactory.getStatistics()** 経由で 統計 オブジェクトへのアクセスを提供します。たとえば、インデックス付けされるクラスや、インデックスに含まれるエンティティーの数を判別できます。この情報は、常に利用できます。ただし、設定で**hibernate.search.generate_statistics** プロパティーを指定すると、合計および平均の **Lucene** クエリーおよびオブジェクトの読み込みタイミングを収集することもできます。

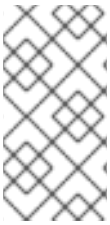
Hibernate Search には、操作を監視する複数のメソッドがあります。インデックスされたクラスやインデックスごとのエンティティー数の一覧は、**SearchFactory.getStatistics()** メソッドを介して常に **Statistics** オブジェクトから利用できます。**Lucene** クエリーおよびオブジェクトロードのタイミングの合計および平均を取得するには、設定に **hibernate.search.generate_statistics** プロパティーを指定します。

JMX 経由での統計へのアクセス

JMX による統計へのアクセスを有効にするには、**hibernate.search.jmx_enabled** プロパティーを **true** に設定します。これにより、**Statistic InfoMBean Bean** が自動的に登録され、統計オブジェクトを介して統計にアクセスできます。設定によっては、**IndexingProgressMonitorMBean Bean** も登録される可能性があります。

インデックスのモニタリング

マスクサー **API** を使用している場合は、**IndexingProgressMonitorMBean Bean** を使用してインデックスの進捗をモニターできます。**Bean** は、インデックスが進行中に **JMX** にのみバインドされます。



注記

*システムプロパティー **com.sun.management.jmxremote** を **true** に設定すると、**JConsole** 経由で **JMX Bean** にリモートでアクセスできます。*

バグの報告

第24章 AMAZON EC2 での JBOSS EAP 6 のデプロイ

24.1. はじめに

24.1.1. Amazon EC2 について

Amazon Elastic Compute Cloud(Amazon EC2)は、**amazon.com** が運用するサービスで、カスタマイズ可能な仮想コンピューティング環境を提供します。サービスを使用して **Amazon Machine Image(AMI)**を起動し、仮想マシンまたはインスタンスを作成できます。ユーザーは、インスタンスに必要なソフトウェアをインストールし、使用する容量に応じて課金することができます。**Amazon EC2** は、柔軟であり、ユーザーがデプロイされたアプリケーションを迅速にスケーリングできるように設計されています。

詳細は、**Amazon EC2** の **Web** サイト を参照して **http://aws.amazon.com/ec2/** ください。

バグの報告

24.1.2. Amazon Machine Instance (AMI)

Amazon Machine Image(AMI)は、**EC2** 仮想マシンインスタンスのテンプレートです。ユーザーは、インスタンスを作成するための適切な **AMI** を選択して **EC2** インスタンスを作成します。**AMI** の主要コンポーネントは、インストール済みのオペレーティングシステムやその他のソフトウェアを含む読み取り専用ファイルシステムです。**AMI** ごとに、さまざまなユースケース用にインストールされるソフトウェアが異なります。**Amazon EC2** には、**amazon.com** とサードパーティーが提供する多くの **AMI** が含まれています。また、ユーザーは独自のカスタム **AMI** を作成することもできます。

バグの報告

24.1.3. JBoss Cloud Access

JBoss Cloud Access は、**Amazon EC2** などの **Red Hat** 認定クラウドインフラストラクチャープロバイダーで **JBoss EAP 6** のサポートを提供する **Red Hat** サブスクリプション機能です。**JBoss Cloud Access** では、従来のサーバーとパブリッククラウドベースのリソースとの間のサブスクリプションをシンプルかつコスト効果の高い方法で移動できます。

詳細は、 を参照して **http://www.redhat.com/en/technologies/cloud-computing/cloud-access** ください。

バグの報告

24.1.4. JBoss Cloud Access 機能

***JBoss Cloud Access** プログラムのメンバーシップにより、**Red Hat** が作成したサポート対象プライベート **Amazon Machine Image (AMI)** へのアクセスが提供されます。*

***Red Hat AMI** では、次のソフトウェアが事前にインストールされ、**Red Hat** により完全にサポートされます。*

- ***Red Hat Enterprise Linux 6***
- ***JBoss EAP 6***
- ***JBoss Operations Network (JON) 3** エージェント*
- ***Red Hat Update Infrastructure** を使用した **RPM** による製品アップデート*

*各 **Red Hat AMI** は開始点にすぎず、アプリケーションの要件を満たすためにさらに設定が必要です。*

重要

*現在、**JBoss Cloud Access** はスタンドアロンインスタンスと管理対象ドメインの両方で **full-ha** プロファイルのサポートを提供しません。*

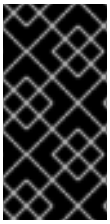
バグの報告

24.1.5. サポートされる Amazon EC2 インスタンスタイプ

***JBoss Cloud Access** は、以下の **Amazon EC2** インスタンスタイプをサポートします。各インスタンスタイプの詳細は、**http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/instance-types.html** 『『**Amazon EC2 ユーザーガイド**』』 を参照してください。*

*表24.1 サポートされる **Amazon EC2** インスタンスタイプ*

インスタンスタイプ	説明
標準のインスタンス	標準インスタンスは、メモリーと CPU の比率が調整された汎用的な環境です。
High Memory Instance	High Memory Instance には、標準インスタンスよりも多くのメモリーが割り当てられています。High Memory Instance は、データベースやメモリーキャッシングアプリケーションなどの高スループットアプリケーションに適しています。
High CPU Instance	High CPU Instance にはメモリーよりも多くの CPU リソースが割り当てられています。このインスタンスは比較的低いスループットですが、CPU 集約型アプリケーションに適しています。



重要

*インスタンスタイプ **Micro(t1.micro)** は **JBoss EAP 6** のデプロイメントには適しません。*

バグの報告

*24.1.6. サポート対象の **Red Hat AMI***

*サポート対象 **Red Hat AMI** は、**AMI** 名により識別できます。*

***JBoss EAP 6** の **AMI** は、次の構文を使用して指定されます。*

> ***RHEL-osversion-JBEAP-version-arch-creationdate***

version** は、**AMI** にインストールされた **JBoss EAP** のバージョン番号です。例 **6.3

osversion** は、**AMI** にインストールされた **Red Hat Enterprise Linux** のバージョン番号です。例: **6.2

***arch** は **AMI** のアーキテクチャーです。これは **x86_64** または **i386** です。*

***CreationDate** は、**AMI** が **YYYYMMDD** の形式で 作成 された日付です。**20120501** の例*

AMI 名の例： RHEL-6.2-JBEAP-6.0.0-x86_64-20120501

バグの報告

24.2. AMAZON EC2 での JBOSS EAP 6 のデプロイ

24.2.1. Amazon EC2 での JBoss EAP 6 のデプロイ (概要)

Amazon EC2 AMI を使用して JBoss EAP 6 をデプロイできます。AMI には、クラスター化されたインスタンスおよびクラスター化されていないインスタンスのデプロイメントに必要なすべてのものが含まれます。

非クラスターインスタンスをデプロイすることは、最も簡単なシナリオです。インスタンスの作成時にアプリケーションのデプロイメントを指定するには、いくつかの設定変更を行う必要があります。

クラスター化されたインスタンスをデプロイするには、さらに設定を行う必要があります。クラスターを含める Virtual Private Cloud を作成することが推奨されます。JBoss EAP インスタンスを使用して mod_cluster プロキシーとして動作するのは任意ですが、このオプションを使用する場合は、S3_PING JGroups 検出プロトコルの S3 バケットも必要になります。

これらの各手順の詳細は以下に示されていますが、JBoss EAP 6、Red Hat Enterprise Linux 6、および Amazon EC2 についてある程度の経験があることを前提としています。

追加リファレンスとして、以下のドキュメンテーションが推奨されます。

- *JBoss EAP 6*

  https://access.redhat.com/documentation/ja-jp/red_hat_jboss_enterprise_application_platform/?version=6.4

- *Red Hat Enterprise Linux 6*

  https://access.redhat.com/documentation/ja-JP/Red_Hat_Enterprise_Linux/

- ***Amazon Web Services***

  http://aws.amazon.com/documentation/

バグの報告

24.3. 非クラスター化の JBOSS EAP 6

24.3.1. 非クラスターインスタンス

*非クラスターインスタンスは、**JBoss EAP 6** を実行する単一の **Amazon EC2** インスタンスです。これはクラスターの一部ではありません。*

バグの報告

24.4. 非クラスターインスタンス

24.4.1. 非クラスター化の JBoss EAP 6 インスタンスの起動

概要

*このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上の **JBoss EAP 6** の非クラスターインスタンスを起動するために必要な手順について説明します。*

前提条件

- *適切な **Red Hat AMI**。「サポート対象の **Red Hat AMI**」 を参照してください。*
- *少なくともポート **22**、**8080**、および **9990** で受信要求を許可する事前設定済みセキュリティーグループ。*

*手順24.1 **Red Hat AMI (Amazon Machine Image)** 上の **JBoss EAP 6** の非クラスターインスタンスを起動する*

1. ***User Data** フィールドを設定します。設定可能なパラメーターは 「永続的な設定パラメーター」、「カスタムスクリプトパラメーター」 から入手できます。*

例24.1 User Data フィールドの例

この例は、非クラスター **JBoss EAP 6** インスタンスの **User Data** フィールドを示しています。ユーザー **admin** のパスワードが **adminpwd** に設定されている。

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
# deploy /usr/share/java/jboss-ec2-eap-applications/<app name>.war
EOC

EOF
```

2. *本番稼働インスタンスの場合*

実稼働インスタンスの場合は、**USER_SCRIPT** フィールドの **User Data** 行の下に次の行を追加し、セキュリティー更新が起動時に適用されるようにします。

```
yum -y update
```

注記

セキュリティー修正と機能強化を適用するには、**yum -y update** が定期的に実行する必要があります。

3. **Red Hat AMI** インスタンスを起動します。

結果

JBoss EAP 6 の非クラスターインスタンスが設定され、**Red Hat AMI** で起動されます。

バグの報告

24.4.2. 非クラスター化 JBoss EAP 6 インスタンスでのアプリケーションのデプロイ

概要

*このトピックでは、**Red Hat AMI** 上の非クラスター **JBoss EAP 6** インスタンスへのアプリケーションのデプロイについて説明します。*

1.

- *サンプルアプリケーションのデプロイ*

  *以下の行を **User Data** フィールドに追加します。*

```
# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
```

*例24.2 サンプルアプリケーションの **User Data** フィールドの例*

*この例では、**Red Hat AMI** で提供されるサンプルアプリケーションを使用します。また、非クラスター化 **JBoss EAP 6** インスタンスの基本設定も含まれます。ユーザー **admin** のパスワードが **adminpwd** に設定されている。*

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security
reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"

# Deploy the sample application from the local filesystem
deploy --force /usr/share/java/jboss-ec2-eap-samples/hello.war
EOC

EOF
```

- *カスタムアプリケーションのデプロイ*

User Data フィールドに以下の行を追加し、アプリケーション名と **URL** を設定します。

```
# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war
```

例**24.3** カスタムアプリケーションの **User Data** フィールドの例

この例では、**MyApp** という名前のアプリケーションが使用され、非クラスター **JBoss EAP 6** インスタンスの基本設定が含まれます。ユーザー **admin** のパスワードが **adminpwd** に設定されている。

```
JBOSSAS_ADMIN_PASSWORD=adminpwd
JBOSS_IP=0.0.0.0 #listen on all IPs and interfaces

# In production, access to these ports needs to be restricted for security
reasons
PORTS_ALLOWED="9990 9443"

cat> $USER_SCRIPT << "EOF"

# Get the application to be deployed from an Internet URL
mkdir -p /usr/share/java/jboss-ec2-eap-applications
wget https://PATH_TO_MYAPP/MyApp.war -O /usr/share/java/jboss-ec2-eap-
applications/MyApp.war

# Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"
deploy /usr/share/java/jboss-ec2-eap-applications/MyApp.war
EOC

EOF
```

2. **Red Hat AMI** インスタンスを起動します。

結果

アプリケーションが **JBoss EAP 6** に正常にデプロイされます。

バグの報告

24.4.3. 非クラスター化 JBoss EAP 6 インスタンスのテスト

概要

このトピックでは、非クラスター **JBoss EAP 6** が正常に実行されていることをテストするために必要な手順について説明します。

手順**24.2** 非クラスター **JBoss EAP 6** インスタンスが正常に実行されていることをテストする

1. インスタンスの詳細ペインにあるインスタンスの パブリック **DNS** を確認します。
2. **http://<public-DNS>:8080** に移動します。
3. 管理コンソールへのリンクを含む、**JBoss EAP** のホームページが表示されることを確認します。ホームページが利用できない場合は、「**Amazon EC2** のトラブルシューティングについて」 を参照してください。
4. 管理コンソール のハイパーリンクをクリックします。
5. ログイン**:**
   - ユーザー名**: admin**
   - **Password: User Data** の 「非クラスター化の **JBoss EAP 6** インスタンスの起動」 フィールドで指定します。
6. サンプルアプリケーションのテスト

   http://<public-DNS>:8080/hello に移動して、サンプルアプリケーションが正常に実行されていることをテストします。**Hello World!** というテキストがブラウザーに表示されます。テキストが表示されない場合は、「**Amazon EC2** のトラブルシューティングについて」 を参照してください。

7. **JBoss EAP 6** の 管理コンソールからログアウトします。

結果

JBoss EAP 6 インスタンスが正常に実行されます。

バグの報告

24.5. 非クラスター化管理対象ドメイン

24.5.1. インスタンスをドメインコントローラーとして提供するための起動

概要

このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上の非クラスター化された **JBoss EAP 6** 管理対象ドメインを起動するために必要な手順について説明します。

前提条件

- 適切な **Red Hat AMI**。「サポート対象の **Red Hat AMI**」 を参照してください。
- 「**Virtual Private Cloud (VPC)** の作成」
- 「**mod_cluster** プロキシとして使用する **Apache HTTP** サーバーインスタンスと **VPC** 用 **NAT** インスタンスの起動 」
- 「**VPC** プライベートサブネットデフォルトルートの設定」
- 「**IAM** セットアップの設定」
- 「**S3** バケットセットアップの設定」

手順24.3 Red Hat AMI 上での非クラスター化 JBoss EAP 6 の管理対象ドメインの起動

1.

セキュリティーグループ タブで、すべてのトラフィックが許可されていることを確認します。必要に応じて、**Red Hat Enterprise Linux** の組み込みファイアウォール機能を使用して、アクセスを制限することができます。

2. **実行** 予定の **VPC** のパブリックサブネットを設定します。

3. 静的 **IP** を選択します。

4. **User Data** フィールドを設定します。設定可能なパラメーターは 「永続的な設定パラメーター」、「カスタムスクリプトパラメーター」 から入手できます。**Amazon EC2** でのドメインコントローラー検出の詳細は、「ドメインコントローラー検索およびフェールオーバーの **Amazon EC2** での設定」 を参照してください。

例**24.4 User Data** フィールドの例

この例は、非クラスター化 **JBoss EAP 6** 管理対象ドメインの **User Data** フィールドを示しています。ユーザー **admin** のパスワードが **admin** に設定されている。

```
## password that will be used by slave host controllers to connect to the domain
controller
JBOSSAS_ADMIN_PASSWORD=admin

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

#### to run the example no modifications below should be needed ####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP
address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Create a file of CLI commands to be executed after starting the server
cat> $USER_CLI_COMMANDS << "EOC"

# Add the modcluster subsystem to the default profile to set up a proxy
/profile=default/subsystem=web/connector=ajp:add(name=ajp,protocol=AJP/1.3,sc
```

```
heme=http,socket-binding=ajp)
/:composite(steps=[ {"operation" => "add", "address" => [ ("profile" => "default"),
("subsystem" => "modcluster") ] },{ "operation" => "add", "address" => [ ("profile"
=> "default"), ("subsystem" => "modcluster"), ("mod-cluster-config" =>
"configuration") ], "advertise" => "false", "proxy-list" =>
"${jboss.modcluster.proxyList}", "connector" => "ajp"}, { "operation" => "add",
"address" => [ ("profile" => "default"), ("subsystem" => "modcluster"), ("mod-
cluster-config" => "configuration"), ("dynamic-load-provider" => "configuration") ]},
{ "operation" => "add", "address" => [ ("profile" => "default"), ("subsystem" =>
"modcluster"), ("mod-cluster-config" => "configuration"), ("dynamic-load-provider"
=> "configuration"), ("load-metric" => "busyness")], "type" => "busyness"} ])

# Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/hello.war --server-groups=main-
server-group
EOC

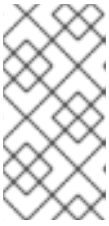
## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF
```

5. *本番稼働インスタンスの場合*

   *実稼働インスタンスの場合は、**USER_SCRIPT** フィールドの **User Data** 行の下に次の行を追加し、セキュリティー更新が起動時に適用されるようにします。*

```
yum -y update
```



注記

*セキュリティー修正と機能強化を適用するには、**yum -y update** が定期的に実行する必要があります。*

6. ***Red Hat AMI** インスタンスを起動します。*

結果

*非クラスター化された **JBoss EAP 6** 管理対象ドメインが設定され、**Red Hat AMI** で起動されます。*

バグの報告

24.5.2. 1 以上のインスタンスを起動し、ホストコントローラーとして提供

概要

このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上の非クラスターホストコントローラーとして機能する **JBoss EAP 6** の **1** つまたは複数のインスタンスを起動するために必要な手順について説明します。

前提条件

- クラスター化されていないドメインコントローラーを設定して起動します。「インスタンスをドメインコントローラーとして提供するための起動」 を参照してください。
- 「**IAM** セットアップの設定」
- 「**S3** バケットセットアップの設定」

手順**24.4** ホストコントローラーの起動

作成する各インスタンスに対して、以下の手順を繰り返します。

1. **AMI** を選択します。
2. インスタンスの必要な数 (スレーブホストコントローラーの数) を定義します。
3. **VPC** およびインスタンスタイプを選択します。
4. **Security Group** をクリックします。
5. **JBoss EAP 6** サブネットからのすべてのトラフィックが許可されることを確認します。

6. 必要に応じて他の制限を定義します。

7. 以下の内容を **User Data:** フィールドに追加します。

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed ####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/other-server-group/main-server-group/"
$JBOSS_CONFIG_DIR/$JBOSS_HOST_CONFIG

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
    echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF
```

Amazon EC2 でのドメインコントローラー検出の詳細は、「ドメインコントローラー検索およびフェールオーバーの **Amazon EC2** での設定」 を参照してください。

8. 本番稼働インスタンスの場合

実稼働インスタンスの場合は、**USER_SCRIPT** フィールドの **User Data** 行の下に次の行を追加し、セキュリティー更新が起動時に適用されるようにします。

```
yum -y update
```

注記

*セキュリティー修正と機能強化を適用するには、**yum -y update** が定期的に実行する必要があります。*

9. ***Red Hat AMI** インスタンスを起動します。*

結果

***JBoss EAP 6** の非クラスターホストコントローラーが設定され、**Red Hat AMI** で起動されます。*

バグの報告

24.5.3. 非クラスター化 JBoss EAP 6 の管理対象ドメインのテスト

概要

*このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上の非クラスター化 **JBoss EAP 6** 管理対象ドメインをテストするために必要な手順について説明します。*

*管理対象ドメインをテストするには、**Apache HTTP** サーバーと **JBoss EAP 6** ドメインコントローラーのエラスティック **IP** アドレスを知っておく必要があります。*

前提条件

- *ドメインコントローラーを設定および起動します。「インスタンスをドメインコントローラーとして提供するための起動」 を参照してください。*
- *ホストコントローラーを設定し、起動します。「**1** 以上のインスタンスを起動し、ホストコントローラーとして提供」 を参照してください。*

*手順**24.5 Web** サーバーのテスト*

- *ブラウザーで **http://ELASTIC_IP_OF_APACHE_HTTPD** に移動し、**Web** サーバーが正常に実行されていることを確認します。*

*手順**24.6** ドメインコントローラーのテスト*

1. *移動先* ***http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console***

2. *ドメインコントローラーの* ***User Data*** *フィールドに指定された* ***admin*** *のユーザー名とパスワードを使用してログインします。また、管理対象ドメインの管理コンソールランディングページが表示されます* ***(http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances)***。

3. *画面右上の* ***Server*** *ラベルをクリックし、画面左上にある* ***Host*** *ドロップダウンメニューでホストコントローラーを選択します。*

4. *各ホストコントローラーに* ***server-one*** *と* ***server-two*** *という* ***2*** *つのサーバー設定があり、それらの両方が* ***main-server-group*** *に属することを確認します。*

5. ***JBoss EAP 6*** *の 管理コンソールからログアウトします。*

手順 ***24.7*** *ホストコントローラーのテスト*

1. ***http://ELASTIC_IP_OF_APACHE_HTTPD/hello*** *に移動して、サンプルアプリケーションが正常に実行されていることをテストします。* ***Hello World!*** *というテキストがブラウザーに表示されます。*

   テキストが表示されない場合は、 ***18.5.1*** *の「* ***Amazon EC2*** *のトラブルシューティングについて」を参照してください。*

2. ***Apache HTTP*** *サーバーインスタンスに接続します:*

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTPD
```

3. ***http://localhost:7654/mod_cluster-manager*** *に移動して、すべてのインスタンスが正常に実行されていることを確認します。*

結果

JBoss EAP 6 Web *サーバー、ドメインコントローラー、およびホストコントローラーが* ***Red Hat AMI*** *で正常に実行されています。*

バグの報告

24.5.4. ドメインコントローラー検索およびフェールオーバーの Amazon EC2 での設定

Amazon EC2 で実行している管理対象ドメインの場合、ドメインコントローラーは Amazon S3 ストレージシステムを使用してドメインコントローラーを動的に検出できます。特に、ホストコントローラーとドメインコントローラーは、Amazon S3 バケットにアクセスするために必要な情報で設定できます。

この設定を使用すると、ドメインコントローラーが起動したときに、その通信情報がバケットの S3 ファイルに書き込まれます。ホストコントローラーがドメインコントローラーへ接続を試みるたびに、S3 ファイルからドメインコントローラーのコンタクト情報を取得します。

つまり、ドメインコントローラーのコンタクト情報が変更されても（たとえば、EC2 インスタンスの IP アドレスが停止および起動時に変更されるのが一般的です）、ホストコントローラーを再設定する必要はありません。ホストコントローラーは、S3 ファイルからドメインコントローラーの新しいコンタクト情報を取得できます。

ドメインコントローラーの検出を有効にするには、JBOSS_DOMAIN_S3_ACCESS_KEY、JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY、および JBOSS_DOMAIN_S3_BUCKET パラメーターを JBoss EAP 6 インスタンスに渡します。設定可能なパラメーターについては、「永続的な設定パラメーター」 を参照してください。または、以下の設定を使用して、ドメイン検出を手動で設定することもできます。

手動によるドメインコントローラーの検出設定は、以下のプロパティーを使用して指定します。

access-key

Amazon AWS ユーザーアカウントのアクセスキー。

secret-access-key

Amazon AWS ユーザーアカウントの秘密アクセスキー。

location

使用される Amazon S3 バケット。

以下は、ホストコントローラーおよびドメインコントローラーの設定例です。以下の例では検出オプションが **1** つ示されていますが、静的検出オプションまたは **S3** 検出オプションをいくつでも設定できます。ドメイン検出およびフェイルオーバープロセスの詳細は、「ドメインコントローラーの検索およびフェールオーバー」 を参照してください。

例**24.5** ホストコントローラーの設定

```
<domain-controller>
  <remote security-realm="ManagementRealm">
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery" module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key" value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </remote>
</domain-controller>
```

例**24.6** ドメインコントローラーの設定

```
<domain-controller>
  <local>
    <discovery-options>
      <discovery-option name="s3-discovery"
code="org.jboss.as.host.controller.discovery.S3Discovery" module="org.jboss.as.host-controller">
        <property name="access-key" value="S3_ACCESS_KEY"/>
        <property name="secret-access-key" value="S3_SECRET_ACCESS_KEY"/>
        <property name="location" value="S3_BUCKET_NAME"/>
      </discovery-option>
    </discovery-options>
  </local>
</domain-controller>
```

バグの報告

24.6. クラスター化された JBOSS EAP 6

24.6.1. クラスターインスタンスについて

クラスターインスタンスは、クラスタリングが有効な **JBoss EAP 6** を実行する **Amazon EC2** インスタンスです。**Apache HTTP** サーバーを実行している別のインスタンスは、クラスター内のインスタンスのプロキシーとして動作します。

JBoss EAP 6 AMI には、クラスターインスタンス（standalone-ec2-ha .xml および standalone-mod_cluster-ec2-ha.xml ）で使用される設定ファイルが 2 つ含まれます。これらの各設定ファイルは、Amazon EC2 がマルチキャストをサポートしないため、マルチキャストを使用せずにクラスタリングを提供します。これは、クラスター通信に TCP ユニキャストを使用し、S3_PING を検出プロトコルとして使用して行います。standalone-mod_cluster-ec2-ha.xml 設定は、mod_cluster プロキシーを使用した登録を容易にします。

同様に、domain-ec2.xml 設定ファイルはクラスター化された管理対象ドメインで使用される ec2-ha および mod_cluster-ec2-ha の 2 つのプロファイルを提供します。

バグの報告

24.6.2. Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) は、プライベートネットワークで AWS リソースのセットを分離できるようにする Amazon Web Services (AWS) の機能です。このプライベートネットワークのトポロジーおよび設定は、ニーズに合わせてカスタマイズできます。

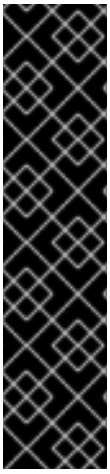
詳細 http://aws.amazon.com/vpc/ は、Amazon Virtual Private Cloud の Web サイトを参照してください。

バグの報告

24.6.3. Virtual Private Cloud (VPC) の作成

概要

このトピックでは、VPC に対して外部のデータベースを例として使用して Virtual Private Cloud を作成するのに必要な手順について説明します。セキュリティーポリシーでは、データベースへの接続を暗号化する必要がある場合があります。データベース接続の暗号化に関する詳細は、Amazon の『RDS FAQ』 を参照してください。



重要

***VPC** はクラスターノード、**JON** サーバー、および **mod_cluster** プロキシー間のセキュアな通信を大幅に簡略化するため、**JBoss EAP 6** クラスターセットアップに推奨されます。**VPC** がない場合、これらの通信チャネルは暗号化され、認証される必要があります。*

***SSL** の設定に関する詳細は、『『**Core Management Security Guide** 』』を参照してください。*

1. ***AWS** コンソールの **VPC** タブに移動します。*

2. *必要な場合は、サービスをサブスクライブします。*

3. *「**Create new VPC"**」をクリックします。*

4. ***1** つのパブリックサブネットと **1** つのプライベートサブネットがある **VPC** を選択します。*

   a. *パブリックサブネットを **10.0.0.0/24** に設定し ます。*

   b. *プライベートサブネットを **10.0.1.0/24** に 設定します。*

5. ***Elastic IPs** に移動します。*

6. ***mod_cluster** プロキシ/**NAT** インスタンスが使用するエラスティック **IP** を作成します。*

7. *セキュリティー グループに移動 し、すべての送受信トラフィックを許可するセキュリティーグループを作成します。*

8. *ネットワーク **ACL** に移動します。*

a.
すべての送受信トラフィックを許可する **ACL** を作成します。

b.
TCP ポート **22**、**8009**、**8080**、**8443**、**9990**、および **16163** でのみすべての送受信トラフィックを許可する **ACL** を作成します。

結果

Virtual Private Cloud が正常に作成されます。

バグの報告

24.6.4. mod_cluster プロキシとして使用する Apache HTTP サーバーインスタンスと VPC 用 NAT インスタンスの起動

概要

このトピックでは、**mod_cluster** プロキシとして使用する **Apache HTTP** サーバーインスタンスと **Virtual Private Cloud** 用 **NAT** インスタンスを起動するために必要な手順について説明します。

前提条件

- 「**Virtual Private Cloud (VPC)** の作成」

手順24.8 mod_cluster プロキシとして使用する Apache HTTP サーバーインスタンスと VPC 用 NAT インスタンスの起動

1. このインスタンスに対してエラスティック **IP** を作成します。
2. **AMI** を選択します。
3. **Security Group** に移動し、すべてのトラフィックを許可します（必要な場合は、アクセスを制限する **Red Hat Enterprise Linux** の組み込みファイアウォール機能を使用）。
4. **VPC** のパブリックサブネットで「**Running "running"**」を選択します。

-

5. *静的 **IP**（例： **10.0.0.4**）を選択します。*

6. ***User Data:** フィールドに以下を追加します。*

```
JBOSSCONF=disabled

cat > $USER_SCRIPT << "EOS"

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 0 > /proc/sys/net/ipv4/conf/eth0/rp_filter

iptables -I INPUT 4 -s 10.0.1.0/24 -p tcp --dport 7654 -j ACCEPT
iptables -I INPUT 4 -p tcp --dport 80 -j ACCEPT

iptables -I FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -I FORWARD -s 10.0.1.0/24 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth0 ! -s 10.0.0.4 -j MASQUERADE

# balancer module incompatible with mod_cluster
sed -i -e 's/LoadModule proxy_balancer_module/#\0/' /etc/httpd/conf/httpd.conf

cat > /etc/httpd/conf.d/mod_cluster.conf << "EOF"
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 7654

# workaround JBPAPP-4557
MemManagerFile /var/cache/mod_proxy/manager

<VirtualHost *:7654>
  <Location /mod_cluster-manager>
    SetHandler mod_cluster-manager
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
  </Location>

  <Location />
    Order deny,allow
    Deny from all
    Allow from 10.
    Allow from 127.0.0.1
  </Location>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0
```

```
    ManagerBalancerName mycluster
    ServerAdvertise Off
    EnableMCPMReceive
</VirtualHost>
EOF

echo "`hostname | sed -e 's/ip-//' -e 'y/-/./'`       `hostname`" >> /etc/hosts

semanage port -a -t http_port_t -p tcp 7654 #add port in the apache port list for the
below to work
setsebool -P httpd_can_network_relay 1 #for mod_proxy_cluster to work
chcon -t httpd_config_t -u system_u /etc/httpd/conf.d/mod_cluster.conf

#### Uncomment the following line when launching a managed domain ####
# setsebool -P httpd_can_network_connect 1

service httpd start

EOS
```

7. *このインスタンスに対する **Amazon EC2** クラウド送信元/送信先チェックを無効にして、ルーターとして動作できるようにします。*

   a. *実行中の **Apache HTTP** サーバーインスタンスを右クリックし、「**Change Source/Dest check"**」を選択します。*

   b. ***Yes, Disable** をクリックします。*

8. *このインスタンスにエラスティック **IP** を割り当てます。*

結果

***Apache HTTP** サーバーインスタンスが正常に起動されます。*

バグの報告

24.6.5. VPC プライベートサブネットデフォルトルートの設定

概要

*このトピックでは、**VPC** プライベートサブネットデフォルトルートを設定するために必要な手順について説明します。**JBoss EAP 6** クラスターノードは **VPC** のプライベートサブネットで実行されます*

が、クラスターノードには **S3** 接続のインターネットアクセスが必要です。**NAT** インスタンスを通過するには、デフォルトのルートを設定する必要があります。

手順**24.9 VPC** プライベートサブネットデフォルトルートの設定

1. **Amazon AWS** コンソールで **Apache HTTP** サーバーインスタンスに移動します。
2. **VPC** → ルートテーブル に移動します。
3. プライベートサブネットで使用されたルーティングテーブルをクリックします。
4. 新規ルートのフィールドに **0.0.0.0/0** を入力 し ます。
5. 「**Select a target"**」をクリックします。
6. **Enter Instance ID"** を選択します。
7. 稼働している **Apache HTTP** サーバーインスタンスの **ID** を選択します。

結果

デフォルトルートが、**VPC** サブネットに対して正常に設定されます。

バグの報告

24.6.6. Identity and Access Management (IAM)

Identity and Access Management(IAM)は、**AWS** リソースの設定可能なセキュリティーを提供します。**IAM** は、**IAM** で作成されたアカウントを使用するか、**IAM** と独自の **ID** サービス間のアイデンティティーフェデレーションを提供するように設定できます。

詳細 **http://aws.amazon.com/iam/** は、**AWS Identity and Access Management** の **Web** サイトを参照してください。

バグの報告

24.6.7. IAM セットアップの設定

概要

このトピックでは、クラスター JBoss EAP 6 インスタンスの IAM 設定に必要な設定手順について説明します。S3_PING プロトコルは、S3 バケットを使用して他のクラスターメンバーを検出します。JGroups バージョン 3.0.x では、S3 サービスに対して認証を行うために Amazon AWS アカウントアクセスおよびシークレットキーが必要です。

S3 ドメインコントローラー検出は S3 バケットを利用するため、S3 サービス（JGroups によって使用される S3 _PING プロトコルと同様に）に対して認証を行うために Amazon AWS アカウントアクセスと秘密鍵が必要です。S3 検出に使用される IAM ユーザーと S3 バケットは、クラスタリングに使用される IAM ユーザーと S3 バケットとは異なる必要があります。

user-data フィールドにメインのアカウント認証情報を入力し、オンラインまたは AMI に保存するセキュリティー上のリスクがあります。これを回避するには、単一の S3 バケットへのアクセスのみが付与される Amazon IAM 機能を使用して別のアカウントを作成できます。

手順24.10 IAM セットアップの設定

1. AWS コンソールの IAM タブに移動します。
2. ユーザー をクリックします。
3. 「Create New Users」を選択します。
4. 名前を選択し、各 User オプションに対して「Generate an access key for each User」オプションがチェックされていることを確認します。
5. Download credentials を選択し、安全な場所に保存します。
6. ウィンドウを閉じます。

–

7. 新しく作成されたユーザーをクリックします。

8. **User ARM** の値を書き留めておきます。この値は、**S3** バケットをセットアップするために必要です（「**S3** バケットセットアップの設定」 を参照）。

結果

IAM ユーザーアカウントが正常に作成されます。

バグの報告

24.6.8. S3 バケット

S3 バケットは、**Amazon Simple Storage System(Amazon S3)**の基本的な組織ストア単位です。バケットは任意の数の任意のオブジェクトを保存でき、**Amazon S3** で識別するには一意の名前が必要です。

詳細は、**Amazon S3** の **Web** サイトを参照して **http://aws.amazon.com/s3/** ください。

バグの報告

24.6.9. S3 バケットセットアップの設定

概要

このトピックでは、新しい **S3** バケットを設定するのに必要な手順について説明します。

前提条件

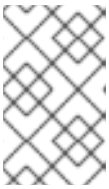
- 「**IAM** セットアップの設定」.

手順**24.11 S3** バケットセットアップの設定

1. **AWS** コンソールで **S3** タブを開きます。

2. ***Create Bucket*** *をクリックします。*

3. *バケットの名前を選択し、* ***Create*** *を クリックします。*



注記

バケット名は ***S3*** *全体で一意です。名前を再利用することはできません。*

4. *新しいバケットを右クリックし、* ***Properties*** *を選択します。*

5. ***permissions*** *タブで* ***Add bucket policy*** *をクリックします。*

6. ***New policy*** *をクリックして、ポリシー作成ウィザードを開きます。*

a. *以下の内容を新しいポリシーにコピーし、* ***arn:aws:iam::055555555555:user/jbosscluster**** *を* *「IAM セットアップの設定」* *に定義された値に置き換えます。* ***clusterbucket123*** *の両方のインスタンスを、この手順のステップ* ***3*** *で定義されたバケットの名前に変更します。*

```
{
  "Version": "2008-10-17",
  "Id": "Policy1312228794320",
  "Statement": [
    {
      "Sid": "Stmt1312228781799",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::055555555555:user/jbosscluster"
        ]
      },
      "Action": [
        "s3:ListBucketVersions",
        "s3:GetObjectVersion",
        "s3:ListBucket",
        "s3:PutBucketVersioning",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:GetObject",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts",
        "s3:PutObject",
```

```
            "s3:GetBucketVersioning"
        ],
        "Resource": [
            "arn:aws:s3:::clusterbucket123/*",
            "arn:aws:s3:::clusterbucket123"
        ]
    }
  ]
}
```

結果

新しい **S3** バケットが正常に作成および設定されます。

バグの報告

24.7. クラスターインスタンス

24.7.1. クラスター化された JBoss EAP 6 AMI の起動

概要

このトピックでは、クラスター **JBoss EAP 6 AMI** を起動するのに必要な手順について説明します。

前提条件

- 「**Virtual Private Cloud (VPC)** の作成」.
- 「**mod_cluster** プロキシとして使用する **Apache HTTP** サーバーインスタンスと **VPC** 用 **NAT** インスタンスの起動 」.
- 「**VPC** プライベートサブネットデフォルトルートの設定」.
- 「**IAM** セットアップの設定」.
- 「**S3** バケットセットアップの設定」.

警告

***JBoss EAP** クラスターを、**24** ビット未満のネットワークマスクがあるサブネットで実行したり、複数のサブネットにまたがるようにすると、各クラスターメンバーに対して一意のサーバーピア **ID** を取得することが複雑になります。*

*このような設定作業を安定的に行う方法については、**JBOSS_CLUSTER_ID** 変数を参照してください(* *「永続的な設定パラメーター」* *)。*

重要

*自動スケーリング **Amazon EC2** 機能は、**JBoss EAP 6** クラスターノードで使用できます。ただし、デプロイメント 前にテストする ようにしてください。特定のワークロードを必要なノード数にスケーリングし、使用するインスタンスタイプに対してパフォーマンスが要件を満たすようにする必要があります（異なるインスタンスタイプは、**EC2** クラウドリソースの別の共有を受け取ります）。*

*さらに、インスタンスの局所性と現在のネットワーク/ストレージ/ホストマシン/**RDS** の使用率は、クラスターのパフォーマンスに影響を与える可能性があります。予想される実際の負荷をテストし、予期しない状況を考慮するようにしてください。*

クラスターのダウンスケール

***Amazon EC2** の **スケールダウン** アクションは、正常にシャットダウンせずにノードを終了します。トランザクションが中断される可能性があるため、他のクラスターノード（およびロードバランサー）がフェールオーバーする時間が必要になります。これは、アプリケーションユーザーエクスペリエンスに影響を与える可能性があります。*

*処理されたセッションが完了するまで **mod_cluster** 管理インターフェースからサーバーを無効にして手動でアプリケーションクラスターをスケールダウンするか、**JBoss EAP 6** インスタンスを正常にシャットダウンすることが推奨されます (インスタンスへの **SSH** アクセスまたは **JON** を使用できます)。*

選択したスケールダウンの手順が原因で、ユーザー体験に悪影響が及ぼさないことをテストします。特定のワークロード、ロードバランサー、および設定には追加の措置が必要になる可能性があります。

*手順24.12 クラスター化された **JBoss EAP 6 AMI** の起動*

1. ***AMI** を選択します。*
2. *必要な数のインスタンス (クラスターサイズ) を定義します。*
3. ***VPC** およびインスタンスタイプを選択します。*
4. *セキュリティーグループ を クリックします。*
5. ***JBoss EAP 6** クラスターサブネットからのすべてのトラフィックが許可されることを確認します。*
6. *必要に応じて他の制限を定義します。*

7. 以下の内容を **User Data** フィールドに追加します。

例**24.7 User Data** フィールドの例

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## password to access admin console
JBOSSAS_ADMIN_PASSWORD=<your password for opening admin console>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -
Ddb.host=instancename.something.rds.amazonaws.com -
Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed ####
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP
address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM

cat > $USER_CLI_COMMANDS << "EOC"
## Deploy sample application from local filesystem
```

```
deploy --force /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war

## ExampleDS configuration for MySQL database
data-source remove --name=ExampleDS
/subsystem=datasources/jdbc-driver=mysql:add(driver-name="mysql",driver-
module-name="com.mysql")
data-source add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/subsystem=datasources/data-source=ExampleDS:enable
/subsystem=datasources/data-source=ExampleDS:test-connection-in-pool
EOC

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF
```

結果

クラスター **JBoss EAP 6 AMI** が正常に設定および起動されます。

バグの報告

24.7.2. クラスター化 JBoss EAP 6 インスタンスのテスト

概要

このトピックでは、クラスター化された **JBoss EAP 6** インスタンスが正常に実行されていることを確認する手順について説明します。

手順24.13 クラスター化されたインスタンスのテスト

1. ブラウザーで に **http://ELASTIC_IP_OF_APACHE_HTTPD** 移動し、**Web** サーバーが正常に実行されていることを確認します。

2. クラスター化されたノードのテスト

   a. ブラウザーで に移動し **http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp** ます。

b. いずれかのクラスターノードが次のメッセージをログに記録することを確認します。

```
Putting date now
```

c. 前の手順でメッセージをログに記録したクラスターノードを停止します。

d. ブラウザーで に移動し **http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp** ます。

e. 示された時間が、手順 **2-a** で **put.jsp** を使用して **PUT** であった時間と同じであることを確認します。

f. いずれかの稼働クラスターノードが次のメッセージをログに記録することを確認します。

```
Getting date now
```

g. 停止されたクラスターノードを再起動します。

h. **Apache HTTP** サーバーインスタンスに接続します**:**

```
ssh -L7654:localhost:7654 <ELASTIC_IP_OF_APACHE_HTTPD>
```

i. に移動し **http://localhost:7654/mod_cluster-manager**、すべてのインスタンスが正常に実行されていることを確認します。

結果

クラスター化された **JBoss EAP 6** インスタンスがテストされ、正常に稼働していることが確認されます。

バグの報告

24.8. クラスター化管理対象ドメイン

24.8.1. クラスタードメインコントローラーとして機能するインスタンスの起動

概要

このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上のクラスター化された **JBoss EAP 6** 管理対象ドメインを起動するために必要な手順について説明します。

前提条件

- 適切な **Red Hat AMI**。「サポート対象の **Red Hat AMI**」 を参照してください。
- 「**Virtual Private Cloud (VPC)** の作成」
- 「**mod_cluster** プロキシとして使用する **Apache HTTP** サーバーインスタンスと **VPC** 用 **NAT** インスタンスの起動 」
- 「**VPC** プライベートサブネットデフォルトルートの設定」
- 「**IAM** セットアップの設定」
- 「**S3** バケットセットアップの設定」

手順**24.14** クラスタードメインコントローラーの起動

1. このインスタンスに対してエラスティック **IP** を作成します。
2. **AMI** を選択します。
3. **Security Group** に移動し、すべてのトラフィックを許可します **(**必要な場合は、アクセスを制限する **Red Hat Enterprise Linux** の組み込みファイアウォール機能を使用**)**。

4. *VPC のパブリックサブネットで **running** を選択します。*

5. *静的 **IP**（例： **10.0.0.5**）を選択 します。*

6. *以下の内容を **User Data:** フィールドに指定します。*

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## password that will be used by slave host controllers to connect to the domain
controller
JBOSSAS_ADMIN_PASSWORD=<password for slave host controllers>

## subnet prefix this machine is connected to
SUBNET=10.0.0.

## S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

#### to run the example no modifications below should be needed ####
JBOSS_DOMAIN_CONTROLLER=true
PORTS_ALLOWED="9999 9990 9443"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Get the application to be deployed from an Internet URL
# mkdir -p /usr/share/java/jboss-ec2-eap-applications
# wget https://<your secure storage hostname>/<path>/<app name>.war -O
/usr/share/java/jboss-ec2-eap-applications/<app name>.war

## Install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM
```

```
cat > $USER_CLI_COMMANDS << "EOC"
## Deploy the sample application from the local filesystem
deploy /usr/share/java/jboss-ec2-eap-samples/cluster-demo.war --server-
groups=other-server-group

## ExampleDS configuration for MySQL database
data-source --profile=mod_cluster-ec2-ha remove --name=ExampleDS
/profile=mod_cluster-ec2-ha/subsystem=datasources/jdbc-driver=mysql:add(driver-
name="mysql",driver-module-name="com.mysql")
data-source --profile=mod_cluster-ec2-ha add --name=ExampleDS --connection-
url="jdbc:mysql://${db.host}:3306/${db.database}" --jndi-
name=java:jboss/datasources/ExampleDS --driver-name=mysql --user-
name="${db.user}" --password="${db.passwd}"
/profile=mod_cluster-ec2-ha/subsystem=datasources/data-source=ExampleDS:enable
EOC

## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF
```

7. *本番稼働インスタンスの場合*

   *実稼働インスタンスの場合は、**USER_SCRIPT** フィールドの **User Data** 行の下に次の行を追加し、セキュリティー更新が起動時に適用されるようにします。*

```
yum -y update
```

注記

*セキュリティー修正と機能強化を適用するには、**yum -y update** が定期的に実行する必要があります。*

8. ***Red Hat AMI** インスタンスを起動します。*

結果

*クラスター化された **JBoss EAP 6** 管理対象ドメインが設定され、**Red Hat AMI** で起動されます。*

バグの報告

24.8.2. クラスターホストコントローラーとして機能する 1 つまたは複数のインスタンスの起動

概要

このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上のクラスターホストコントローラーとして機能する **JBoss EAP 6** の **1** つまたは複数のインスタンスを起動するために必要な手順について説明します。

前提条件

- クラスタードメインコントローラーを設定および起動します。「クラスタードメインコントローラーとして機能するインスタンスの起動」 を参照してください。
- 「**IAM** セットアップの設定」
- 「**S3** バケットセットアップの設定」

手順**24.15** ホストコントローラーの起動

作成する各インスタンスに対して、以下の手順を繰り返します。

1. **AMI** を選択します。
2. インスタンスの必要な数 **(**スレーブホストコントローラーの数**)** を定義します。
3. **VPC** およびインスタンスタイプを選択します。
4. **Security Group** をクリックします。
5. **JBoss EAP 6** クラスターサブネットからのすべてのトラフィックが許可されることを確認します。
6. 必要に応じて他の制限を定義します。

7. 以下の内容を **User Data:** フィールドに追加します。

```
## mod cluster proxy addresses
MOD_CLUSTER_PROXY_LIST=10.0.0.4:7654

## clustering setup
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY=<your secret key>
JBOSS_JGROUPS_S3_PING_ACCESS_KEY=<your access key>
JBOSS_JGROUPS_S3_PING_BUCKET=<your bucket name>

## host controller setup
### static domain controller discovery setup
JBOSS_DOMAIN_MASTER_ADDRESS=10.0.0.5
### S3 domain controller discovery setup
# JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY=<your secret key>
# JBOSS_DOMAIN_S3_ACCESS_KEY=<your access key>
# JBOSS_DOMAIN_S3_BUCKET=<your bucket name>

JBOSS_HOST_PASSWORD=<password for slave host controllers>

## database credentials configuration
JAVA_OPTS="$JAVA_OPTS -Ddb.host=instancename.something.rds.amazonaws.com
-Ddb.database=mydatabase -Ddb.user=<user> -Ddb.passwd=<pass>"

## subnet prefix this machine is connected to
SUBNET=10.0.1.

#### to run the example no modifications below should be needed ####
JBOSS_HOST_USERNAME=admin
PORTS_ALLOWED="1024:65535"
JBOSS_IP=`hostname | sed -e 's/ip-//' -e 'y/-/./'` #listen on public/private EC2 IP address

cat > $USER_SCRIPT << "EOF"
## Server instance configuration
sed -i "s/main-server-group/other-server-group/"
$JBOSS_CONFIG_DIR/$JBOSS_HOST_CONFIG

## install the JDBC driver as a core module
yum -y install mysql-connector-java
mkdir -p /usr/share/jbossas/modules/com/mysql/main
cp -v /usr/share/java/mysql-connector-java-*.jar
/usr/share/jbossas/modules/com/mysql/main/mysql-connector-java.jar

cat > /usr/share/jbossas/modules/com/mysql/main/module.xml <<"EOM"
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
    <resource-root path="mysql-connector-java.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
EOM
```

```
## this will workaround the problem that in a VPC, instance hostnames are not
resolvable
echo -e "127.0.0.1\tlocalhost.localdomain localhost" > /etc/hosts
echo -e "::1\tlocalhost6.localdomain6 localhost6" >> /etc/hosts
for (( i=1 ; i<255 ; i++ )); do
  echo -e "$SUBNET$i\tip-${SUBNET//./-}$i" ;
done >> /etc/hosts

EOF
```

8. *本番稼働インスタンスの場合*

   *実稼働インスタンスの場合は、**USER_SCRIPT** フィールドの **User Data** 行の下に次の行を追加し、セキュリティー更新が起動時に適用されるようにします。*

```
yum -y update
```

注記

*セキュリティー修正と機能強化を適用するには、**yum -y update** が定期的に実行する必要があります。*

9. ***Red Hat AMI** インスタンスを起動します。*

結果

***JBoss EAP 6** のクラスターホストコントローラーが設定され、**Red Hat AMI** で起動されます。*

バグの報告

24.8.3. クラスター化された JBoss EAP 6 管理対象ドメインのテスト

概要

*このトピックでは、**Red Hat AMI (Amazon Machine Image)** 上のクラスター化された **JBoss EAP 6** 管理対象ドメインをテストするために必要な手順について説明します。*

*管理対象ドメインをテストするには、**Apache HTTP** サーバーと **JBoss EAP 6** ドメインコントローラーのエラスティック **IP** アドレスを知っておく必要があります。*

前提条件

- *クラスタードメインコントローラーを設定および起動します。「クラスタードメインコントローラーとして機能するインスタンスの起動」 を参照してください。*

- *クラスターホストコントローラーを設定し、起動します。「クラスターホストコントローラーとして機能する 1 つまたは複数のインスタンスの起動」 を参照してください。*

手順24.16 Apache HTTP サーバーインスタンスのテスト

- *ブラウザーで **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER** に移動し、**Web** サーバーが正常に実行されていることを確認します。*

手順24.17 ドメインコントローラーのテスト

1. *移動先 **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console***

2. *ドメインコントローラーの **User Data** フィールドで指定されるユーザー名 **admin** とパスワードを使用してログインします。ログインすると、管理対象ドメインの管理コンソールランディングページが表示されます **(http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances)**。*

3. *画面右上の **Server** ラベルをクリックします。画面左上にある **Host** ドロップダウンメニューでホストコントローラーを選択します。*

4. *このホストコントローラーに **server-one** と **server-two** という **2** つのサーバー設定があることを確認し、それら両方が **other-server-group** に属することを確認します。*

手順24.18 ホストコントローラーのテスト

1. *ブラウザーで **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/put.jsp** に移動します。*

2. *ホストコントローラーの **1** つが **Putting date now** というメッセージをログに記録することを確認します。*

3.

前の手順でメッセージをログに記録したサーバーインスタンスを停止します（「『管理コンソールを使用したサーバーの停止」を参照』）。

4. ブラウザーで **http://ELASTIC_IP_OF_APACHE_HTTP_SERVER/cluster-demo/get.jsp** に移動します。

5. 示された時間が、手順 **2** で **put.jsp** を使用して **PUT** の時間と同じであることを確認します。

6. 実行中のサーバーインスタンスの **1** つが **Getting date now** というメッセージをログに記録することを確認します。

7. 停止したサーバーインスタンスを再起動します（「管理コンソールを使用したサーバーの起動」を参照）。

8. **Apache HTTP** サーバーインスタンスに接続します。

```
$ ssh -L7654:localhost:7654 ELASTIC_IP_OF_APACHE_HTTP_SERVER
```

9. **http://localhost:7654/mod_cluster-manager** に移動して、すべてのインスタンスが正常に実行されていることを確認します。

結果

JBoss EAP 6 Web サーバー、ドメインコントローラー、およびホストコントローラーが **Red Hat AMI** で正常に実行されています。

バグの報告

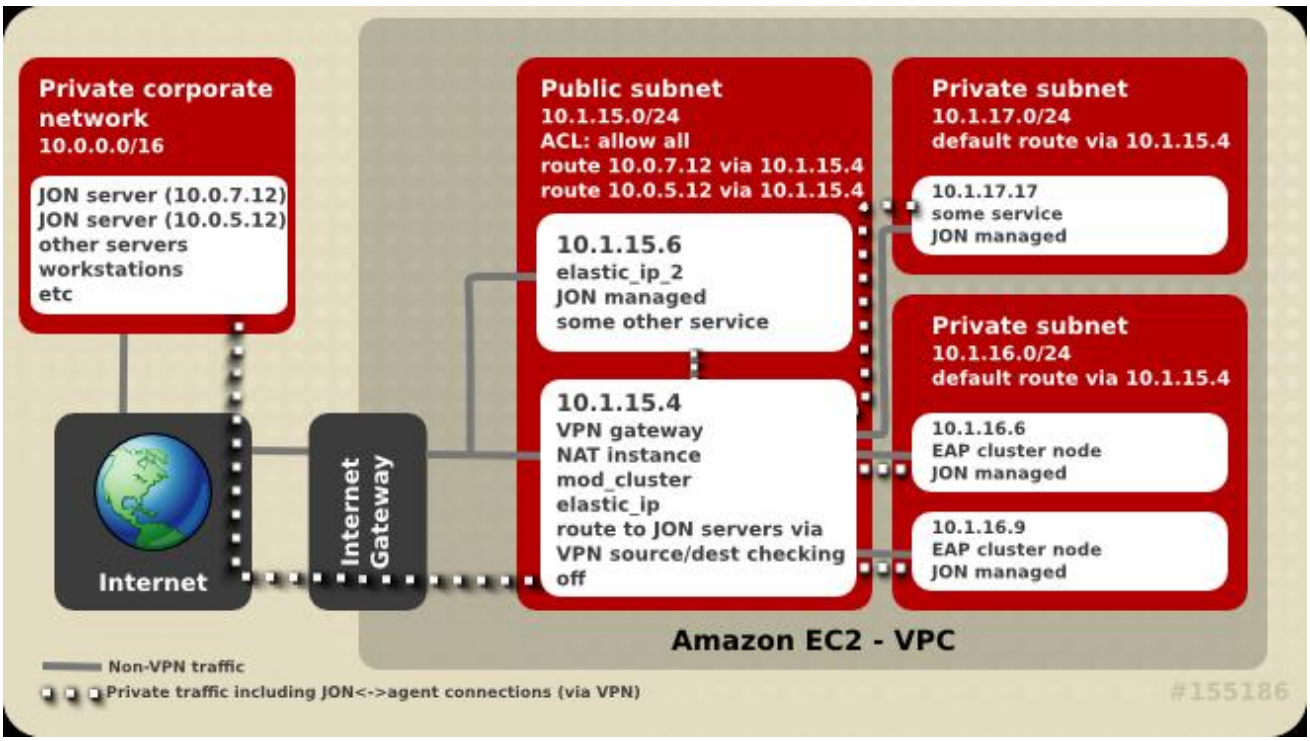
24.9. JBOSS OPERATIONS NETWORK (JON) での監視の確立

24.9.1. AMI 監視

適切に設定された **AMI** インスタンスにデプロイされたビジネスアプリケーションの場合、次の手順は **JBoss Operations Network (JON)** があるプラットフォームの監視を確立します。

JON サーバーは一般的に企業ネットワーク内にあるため、サーバーと各エージェントとの間のセキュアな接続を確立する必要があります。2 点間で VPN を確立するのは最も一般的なソリューションですが、これは必要なネットワーク設定を複雑にします。本章では、JON エージェントと JON サーバー間の通信を有効にするためのネットワーク設定ガイドラインを提供します。設定、管理、および使用の詳細については、JBoss Operations Network(JON)の公式 Red Hat ドキュメントを参照してください。

図24.1 JON サーバーの接続性



[D]

バグの報告

24.9.2. 接続要件

JON エージェントをサーバーに登録するには、エージェントとサーバー間の双方向通信が必要です。JON エージェントは、ポート 7443 が使用されている場合を除き、すべての JON サーバーのポート 7080 にアクセスする必要があります。各 JON サーバーは、一意のホストとポートのペアで接続された各エージェントにアクセスできる必要があります。エージェントポートは通常 16163 です。

複数のクラスター化された JON サーバーがある場合は、JON サーバー管理コンソールで設定されたように、各エージェントが IP とホスト名のペアを介して JON クラスターのすべてのサーバーと通信できることを確認します。エージェントを登録するために使用される JON サーバーは、初期化後に使用しようとするサーバーではない可能性があります。

バグの報告

24.9.3. Network Address Translation (NAT)

企業 VPN ゲートウェイがルーティングモードで動作すると、ネットワーク設定が大幅に簡素化されます。ただし、企業 VPN ゲートウェイが NAT モードで動作している場合、JON サーバーはエージェントの直接的な可視性を持ちません。この場合、各エージェントにポート転送を設定する必要があります。

NAT VPN 設定では、ゲートウェイのポートを管理対象マシンのポートの JON エージェントのアドレスに転送する必要があります。また、JON エージェントは、転送されたポート番号と IP アドレスをサーバーに伝えるように設定する必要があります。詳細は、agent-configuration .xml 設定ファイルの rhq.communications.connector.* の説明を参照してください。

バグの報告

24.9.4. Amazon EC2 および DNS

JON サーバーおよび JON エージェントは、相互のホスト名を解決できる必要があります。VPN 構成の場合は、DNS 解決がより複雑になります。接続されたサーバーには複数のオプションがあります。1 つのオプションとして、Amazon EC2 または企業ネットワークの DNS サーバーのいずれかを使用できます。別のオプションとして、企業 DNS サーバーが特定のドメインで名前を解決するために使用され、Amazon EC2 DNS サーバーが他のすべての名前を解決するために使用される、分割された DNS 設定を使用します。

バグの報告

24.9.5. EC2 でのルーティング

すべての Amazon EC2 サーバーでは、デフォルトで ソース/宛先の確認 機能がアクティベートされます。この機能は、マシンの IP アドレスとは異なる宛先を持つサーバーに送信されるパケットを破棄します。JON サーバーへのエージェントの接続に選択した VPN ソリューションにルーターが含まれている場合は、ルーターまたは VPN ゲートウェイとして動作するサーバーに対してこの機能をオフにする必要があります。この設定は、Amazon AWS コンソールからアクセスできます。Virtual Private Cloud(VPC) で無効にされた ソース/宛先チェック も必要になります。

一部の VPN 設定は、デフォルトで企業 VPN を介して一般的なインターネットトラフィックをルーティングします。特定のニーズに対応する速度が低く、効率が悪い場合があるため、これを回避することが推奨されます。

適切なアドレス指定スキーマの使用は **JON** に固有のものではありませんが、スキーマの低下は影響を受ける可能性があります。**Amazon EC2** は、**10.0.0.0/8** ネットワークから **IP** アドレスを割り当てます。インスタンスには通常パブリック **IP** アドレスがありますが、同じアベイラビリティーゾーン内の内部 **IP** アドレス上のネットワークトラフィックのみが解放されます。プライベートアドレス指定で **10.0.0.0/8** ネットワークを使用しないように、考慮すべき事項がいくつかあります。

- **VPC** の作成時に、プライベートネットワークですでに使用されているアドレスの割り当てを回避し、接続の問題を回避します。
- インスタンスが利用可能なゾーンのローカルリソースにアクセスする必要がある場合は、**Amazon EC2** プライベートアドレスが使用され、トラフィックが **VPN** を介してルーティングされないことを確認します。
- **Amazon EC2** インスタンスが企業プライベートネットワークアドレスの小さなサブセット（**JON** サーバーなど）にアクセスする場合、これらのアドレスのみが **VPN** 経由でルーティングされる必要があります。これにより、セキュリティーが強化され、**Amazon EC2** またはプライベートネットワークアドレス空間の競合の可能性が低くなります。

バグの報告

24.9.6. JON での終了と再起動

クラウド環境の利点の **1** つは、マシンインスタンスを終了して起動できることです。初期インスタンスと同じインスタンスを起動することもできます。新しいインスタンスが以前に実行したエージェントと同じエージェント名を使用して **JON** サーバーで登録しようとすると、問題が発生することがあります。これが発生すると、**JON** サーバーは、エージェントが不足している識別トークンまたは一致しない **ID** トークンで再接続することを許可しません。

これを回避するには、同じ名前のエージェントを接続する前に、終了したエージェントが **JON** インベントリーから削除されていることを確認するか、新しいエージェントの起動時に正しい識別トークンを指定します。

別の問題は、エージェントマシンに **JON** 設定で記録されたアドレスと一致しなくなった新しい **VPN IP** アドレスが割り当てられている場合です。たとえば、再起動するマシンや **VPN** 接続が中断されるマシンが含まれる場合があります。この場合、**JON** エージェントのライフサイクルを **VPN** 接続のライフサイクルにバインドすることが推奨されます。接続が破棄されると、エージェントを停止できます。接続が再び復元されたら、**/etc/sysconfig/jon-agent-ec2** の **JON_AGENT_ADDR** を更新し、新しい **IP** アドレスを反映してエージェントを再起動します。

エージェントの IP アドレスを変更する方法については、にある JON サーバーおよびエージェントの設定ガイドを参照 https://access.redhat.com/documentation/ja-JP/JBoss_Operations_Network してください。

多数のインスタンスが起動または終了している場合は、JON インベントリーから手動で追加および削除できます。Jon のスクリプト機能を使用して、これらの手順を自動化できます。詳細は、JON ドキュメントを参照してください。

バグの報告

24.9.7. JBoss Operations Network で登録するインスタンスの設定

以下の手順を使用して JBoss EAP 6 インスタンスを JBoss Operations Network で登録します。

- JBoss EAP 6 の場合は、以下を User Data フィールドに追加します。

```
JON_SERVER_ADDR=jon2.it.example.com
## if instance not already configured to resolve its hostname
JON_AGENT_ADDR=`ip addr show dev eth0 primary to 0/0 | sed -n 's#.*inet \([0-
9.]\+\)/.*#\1#p'`
PORTS_ALLOWED=16163
# insert other JON options when necessary.
```

JON オプションの形式は、JON_ で始まる 「永続的な設定パラメーター」 パラメーターを参照してください。

バグの報告

24.10. ユーザースクリプト設定

24.10.1. 永続的な設定パラメーター

概要

次のパラメーターを使用して、JBoss EAP 6 の設定と操作に影響を与えることができます。これらの内容は、/etc/sysconfig/jbossas および /etc/sysconfig/jon-agent-ec2 に書き込まれます。

表24.2 設定可能なパラメーター

名前	説明	デフォルト
JBOSS_JGROUPS_S3_PING_ACCESS_KEY	S3_PING 検出用 Amazon AWS ユーザーアカウントアクセスキー (クラスタリングが使用される場合)。	該当なし
JBOSS_JGROUPS_S3_PING_SECRET_ACCESS_KEY	Amazon AWS ユーザーアカウント秘密アクセスキー。	該当なし
JBOSS_JGROUPS_S3_PING_BUCKET	S3_PING 検出に使用する Amazon S3 バケット。	該当なし
JBOSS_CLUSTER_ID	クラスターメンバーノードの ID。クラスタリングにのみ使用されます。許可される値は(in order)です。 ● **0** - **1023** の範囲の有効なクラスター ID 番号。 ● ネットワークインターフェース名 (IP の最後のオクテットが値として使用されます)。 ● 値としての "S3" は、jgroups' S3_PING で使用される S3 バケットを介して ID の使用を調整します。 すべてのクラスターノードが同じ 24 ビット以上のサブネット (VPC サブネットなど) に存在する場合は、IP の最後のオクテット (デフォルト値) を使用することが推奨されます。	eth0 の IP アドレスの最後のオクテット
MOD_CLUSTER_PROXY_LIST	mod_cluster プロキシーの IP/ホスト名のコンマ区切りリスト (mod_cluster を使用する場合)。	該当なし
PORTS_ALLOWED	デフォルトのもの以外に、ファイアウォールで許可される受信ポートのリスト。	該当なし
JBOSSAS_ADMIN_PASSWORD	**admin** ユーザーのパスワード。	該当なし

名前	説明	デフォルト
JON_SERVER_ADDR	登録する JON サーバーのホスト名または IP。これは登録にのみ使用されます。その後、エージェントは JON クラスターの他のサーバーと通信できます。	該当なし
JON_SERVER_PORT	サーバーと通信するためにエージェントが使用するポート。	7080
JON_AGENT_NAME	JON エージェントの名前 (一意である必要があります)。	インスタンスの ID
JON_AGENT_PORT	エージェントがリッスンするポート。	16163
JON_AGENT_ADDR	JON エージェントがバインドされる IP アドレス。これは、サーバーに複数のパブリックアドレスがある場合に使用されます（VPN など）。	JON エージェントは、デフォルトでローカルホスト名の IP を選択します。
JON_AGENT_OPTS	SSL、NAT、および他の高度な設定を指定するために使用できる追加の JON エージェントシステムプロパティー。	該当なし

名前	説明	デフォルト
JBOSS_SERVER_CONFIG	使用する JBoss EAP 6 サーバー設定ファイルの名前。JBOSS_DOMAIN_CONTROLLER=true の場合は、**domain-ec2.xml** が使用されます。それ以外の場合: ● S3 設定が存在する場合は、**standalone-ec2-ha.xml** が使用されます。 ● MOD_CLUSTER_PROXY_LIST が指定されている場合は、**standalone-mod_cluster-ec2-ha.xml** が選択されます。 ● 最初の 2 つのオプションのいずれかが使用されていない場合は、**standalone.xml** ファイルが使用されます。 ● **standalone-full.xml** に設定することもできます。	**standalone.xml**、**standalone-full.xml**、**standalone-ec2-ha.xml**、**standalone-mod_cluser-ec2-ha.xml**、**domain-ec2.xml** は他のパラメーターに依存します。
JAVA_OPTS	JBoss EAP 6 が起動する前に変数に追加するカスタム値。	JAVA_OPTS は、他のパラメーターの値から構築されます。
JBOSS_IP	サーバーがバインドされる IP アドレス。	127.0.0.1
JBOSSCONF	起動する JBoss EAP 6 プロファイルの名前。JBoss EAP 6 が起動しないようにするには、JBOSSCONF を **disabled**に設定します。	**standalone**
JBOSS_DOMAIN_CONTROLLER	このインスタンスをドメインコントローラーとして実行するかどうかを設定します。	**false**
JBOSS_DOMAIN_MASTER_ADDRESS	リモートドメインコントローラーの IP アドレス。	該当なし
JBOSS_HOST_NAME	論理ホスト名（ドメイン内）。これは別でなければなりません。	HOSTNAME 環境変数の値。

名前	説明	デフォルト
JBOSS_HOST_USERNAME	ドメインコントローラーでの登録時にホストコントローラーが使用する必要があるユーザー名。指定しないと、JBOSS_HOST_NAME が代わりに使用されます。	JBOSS_HOST_NAME
JBOSS_HOST_PASSWORD	ドメインコントローラーでの登録時にホストコントローラーが使用する必要があるパスワード。	該当なし
JBOSS_HOST_CONFIG	JBOSS_DOMAIN_CONTROLLER=true の場合は、**host-master.xml** が使用されます。JBOSS_DOMAIN_MASTER_ADDRESS が存在する場合は、**host-slave.xml** が使用されます。	**host-master.xml** または **host-slave.xml**（他のパラメーターによって異なります）。
JBOSS_DOMAIN_S3_ACCESS_KEY	S3 ドメインコントローラー検索用の Amazon AWS ユーザーアカウントのアクセスキー。	該当なし
JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY	S3 ドメインコントローラー検索用の Amazon AWS ユーザーアカウントの秘密アクセスキー。	該当なし
JBOSS_DOMAIN_S3_BUCKET	S3 ドメインコントローラー検索に使用される Amazon S3 バケット。	該当なし

バグの報告

24.10.2. カスタムスクリプトパラメーター

概要

*以下のパラメーターは、**User Data:** フィールドのユーザーカスタマイズセクションで使用できます。*

表24.3 設定可能なパラメーター

名前	説明
JBOSS_DEPLOY_DIR	アクティブプロファイルのデプロイディレクトリー（例：**/var/lib/jbossas/standalone/deployments/**）。このディレクトリーに置かれたデプロイ可能なアーカイブがデプロイされます。Red Hat は、deploy ディレクトリーを使用する代わりに、管理コンソールまたは CLI ツールを使用してデプロイメントを管理することを推奨します。

名前	説明
JBOSS_CONFIG_DIR	アクティブプロファイルの設定ディレクトリー（例：**/var/lib/jbossas/standalone/configuration**）。
JBOSS_HOST_CONFIG	アクティブなホスト設定ファイルの名前（例：**host-master.xml**）。Red Hat は、設定ファイルを編集するのではなく、管理コンソールまたは CLI ツールを使用してサーバーを設定することを推奨します。
JBOSS_SERVER_CONFIG	アクティブなサーバー設定ファイルの名前（例：**standalone-ec2-ha.xml**）。Red Hat は、設定ファイルを編集するのではなく、管理コンソールまたは CLI ツールを使用してサーバーを設定することを推奨します。
USER_SCRIPT	カスタム設定スクリプトへのパス (ソースユーザーデータ設定の前に利用可能)。
USER_CLI_COMMANDS	CLI コマンドのカスタムファイルへのパス (ソースユーザーデータ設定の前に利用可能)。

バグの報告

24.11. トラブルシューティング

24.11.1. Amazon EC2 のトラブルシューティングについて

***EC2** は、インスタンスごとに **Alarm Status** を提供します。これは、致命的なインスタンスの誤動作を示します。ただし、このような警告がなくても、インスタンスが正しく起動し、サービスが正しく実行されていることが保証されるわけではありません。**Amazon CloudWatch** をカスタムメトリクス機能とともに使用すると、インスタンスサービスの健全性を監視することができますが、エンタープライズ管理ソリューションを使用することが推奨されます。*

*トラブルシューティングを単純化するため、**Red Hat** は **JBoss Operations Network(JON)**を使用して **EC2** インスタンスを管理することを推奨します。**JON** は、**JON** エージェントがインストールされた **EC2** インスタンスで多くのサービス（**JBoss EAP 6**、**Tomcat**、**Apache HTTP Server**、**PostgreSQL** など）を自動的に検出、監視、および管理できます。**JON** を使用した **JBoss EAP** の監視に関する詳細は、「**AMI** 監視 」 を参照してください。*

バグの報告

24.11.2. 診断情報

***JBoss Operations Network**、**Amazon CloudWatch**、または手動の検査によって問題が検出された場合、診断情報の一般的なソースは以下のようになります。*

- */var/log/jboss_user-data.out* は ***jboss-ec2-eap init*** スクリプトとユーザーカスタム設定スクリプトの出力です。

- */var/cache/jboss-ec2-eap/* には、インスタンスの起動時に使用される実際のユーザーデータ、カスタムスクリプト、およびカスタム ***CLI*** コマンドが含まれます。

- */var/log* には、マシンの起動時に収集されたすべてのログ、***JBoss EAP 6***、***httpd***、およびその他のサービスも含まれます。

これらのファイルへのアクセスは、***SSH*** セッションからのみ利用できます。***Amazon EC2*** インスタンスで ***SSH*** セッションを設定および確立する方法については、『***Amazon EC Getting Started Guide***』を参照してください。

バグの報告

第25章 セッションの外部化

25.1. JBOSS EAP から JBOSS DATA GRID への HTTP セッションの外部化

Red Hat JBoss Data Grid は、Red Hat JBoss Enterprise Application Platform(JBoss EAP)6.4 以降で、HTTP セッションなどのアプリケーション固有のデータの外部キャッシュコンテナーとして使用できます。これにより、アプリケーションとは独立したデータレイヤーのスケーリングが可能になり、さまざまなドメインに存在する可能性がある異なる EAP クラスターが同じ JBoss Data Grid クラスターからデータにアクセスできるようになります。また、他のアプリケーションは Red Hat JBoss Data Grid によって提供されたキャッシュと対話できます。

以下の手順は、EAP のスタンドアロンモードとドメインモードの両方に適用されますが、ドメインモードでは、各サーバーグループに一意のリモートキャッシュを設定する必要があります。複数のサーバーグループは同じ Red Hat JBoss Data Grid クラスターを使用できますが、各リモートキャッシュは EAP サーバーグループに固有のものです。

注記

分散可能なアプリケーションごとに完全に新しいキャッシュを作成する必要があります。これは既存のキャッシュコンテナー（web など）で作成できます。

手順25.1 HTTP セッションの外部化

1. リモートキャッシュコンテナーが EAP の infinispan サブシステムで定義されているようにしてください。以下の例では、remote-store 要素の cache 属性はリモート JBoss Data Grid サーバーのキャッシュ名を定義します。

```
<subsystem xmlns="urn:jboss:domain:infinispan:4.0">
 [...]
 <cache-container name="web" default-cache="dist"
module="org.jboss.as.clustering.web.infinispan" statistics-enabled="true">
   <transport lock-timeout="60000"/>
   <invalidation-cache name="jdg" mode="SYNC">
    <locking isolation="REPEATABLE_READ"/>
    <transaction mode="BATCH"/>
    <remote-store remote-servers="remote-jdg-server1 remote-jdg-server2"
                  cache="default" socket-timeout="60000"
                  preload="true" passivation="false" purge="false" shared="true"/>
   </replicated-cache>
 </cache-container>
</subsystem>
```

2.

*ネットワーク情報を **socket-binding-group** に追加することにより、リモート **Red Hat JBoss Data Grid** サーバーの場所を定義します。*

```
<socket-binding-group ...>
  <outbound-socket-binding name="remote-jdg-server1">
    <remote-destination host="JDGHostName1" port="11222"/>
  </outbound-socket-binding>
  <outbound-socket-binding name="remote-jdg-server2">
    <remote-destination host="JDGHostName2" port="11222"/>
  </outbound-socket-binding>
</socket-binding-group>
```

3.

*上記の手順を各キャッシュコンテナーと各 **Red Hat JBoss Data Grid** サーバーで繰り返します。定義された各サーバーには、個別の **< outbound-socket-binding>** 要素が定義されている必要があります。*

4.

*パッシベーションおよびキャッシュ情報をアプリケーションの **jboss-web.xml** に追加します。以下の例では、**web** はキャッシュコンテナーの名前で、**jdg** はこのコンテナーにあるデフォルトキャッシュの名前です。ファイルの例を以下に示します。*

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web xmlns="http://www.jboss.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee
http://www.jboss.org/j2ee/schema/jboss-web_10_0.xsd"
          version="10.0">

    <replication-config>
       <replication-granularity>SESSION</replication-granularity>
       <cache-name>web.jdg</cache-name>
    </replication-config>

</jboss-web>
```

注記

*上記のパッシベーションタイムアウトは、典型的なセッションが **15** 分以内に破棄され、**JBoss EAP** の **30** 分間でデフォルトの **HTTP** セッションタイムアウトを使用することを前提とします。これらの値は、各アプリケーションのワークロードに基づいて調整する必要があります。*

バグの報告

付録A 補足リファレンス

A.1. RED HAT カスタマーポータルからのファイルのダウンロード

前提条件

- このタスクを開始する前に、カスタマーポータルのアカウントが必要です。右上隅の https://access.redhat.com **Register** リンクをクリックして、アカウントを作成します。

手順A.1 Red Hat カスタマーポータルにログインしファイルをダウンロード

1. 右上隅の **Log in** のリンクをクリックします。https://access.redhat.com認証情報を入力し、ログイン をクリックします。

   結果

   RHN にログインし、のメイン **Web** ページに https://access.redhat.com 戻ります。

2. **Downloads** ページに移動します。

   以下のオプションのいずれかを使用して **Downloads** ページに移動します。

   - 上部のナビゲーションバーの **Downloads** リンクをクリックします。
   - に直接移動し https://access.redhat.com/downloads/ ます。

3. ダウンロードする製品とバージョンを選択します。

   以下の方法を使い、正しい製品とバージョンを選びダウンロードしてください。

- ナビゲーションを使って 1 つずつ進めていきます。

- 画面の右上端にある検索エリアを使い製品を検索します。

4. お使いのオペレーティングシステムやインストール方法にあったファイルをダウンロードします。

   選択した製品に応じて、特定のオペレーティングシステムおよびアーキテクチャーに **Zip** アーカイブ、**RPM**、またはネイティブインストーラーを選択できます。ファイル名またはダウンロードするファイルの右側にある **Download** リンクをクリックします。

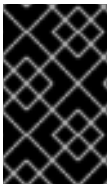
結果

お使いのコンピューターにファイルをダウンロードします。

バグの報告

A.2. RED HAT ENTERPRISE LINUX でのデフォルト JAVA DEVELOPMENT KIT の設定

複数の **Java** 開発キットを **Red Hat Enterprise Linux** システムにインストールすることができます。このタスクは、現在の環境が使用する環境を指定する方法を示しています。**alternatives** コマンドを使用します。



重要

このタスクは **Red Hat Enterprise Linux** のみが対象となります。

注記

この手順を行う必要がない場合があります。**Red Hat Enterprise Linux** は **OpenJDK 1.6** をデフォルトのオプションとして使用します。これが望ましく、システムが適切に動作している場合は、使用する **Java** 開発キットを手動で指定する必要はありません。

前提条件

- *このタスクを完了するには、直接ログインまたは **sudo** コマンドを使用してスーパーユーザー権限が必要です。*

*手順**A.2** デフォルトの **Java Development Kit** の設定*

1. *任意の **java** バイナリーおよび **javac** バイナリーのパスを決定します。*

   *__rpm -ql packagename |grep bin__ コマンドを使用して、**RPM** からインストールしたバイナリーの場所を検索できます。**Red Hat Enterprise Linux 32** ビットシステムの **java** バイナリーおよび **javac** バイナリーのデフォルトの場所は次のとおりです。*

   *表**A.1 java** バイナリーおよび **javac** バイナリーのデフォルトの場所*

Java Development Kit	パス
OpenJDK 1.6	**/usr/lib/jvm/jre-1.6.0-openjdk/bin/java** **/usr/lib/jvm/java-1.6.0-openjdk/bin/javac**
Oracle JDK 1.6	**/usr/lib/jvm/jre-1.6.0-sun/bin/java** **/usr/lib/jvm/java-1.6.0-sun/bin/javac**

2. *個別に代替の **JDK** を設定します。*

   *特定の **java** および **javac** を使用するようシステムを設定します（**/usr/sbin/alternatives --config java** または **/usr/sbin/alternatives --config javac** ）。画面上の指示に従ってください。*

3. *必要に応じて、**java_sdk_1.6.0** の代替オプションを設定します。*

   ***Oracle JDK** を使用する場合は、**java_sdk_1.6.0** の代替を設定する必要もあります。**/usr/sbin/alternatives --config java_sdk_1.6.0** コマンドを使用します。通常、正しいパスは **/usr/lib/jvm/java-1.6.0-sun** です。ファイル一覧を実行して検証できます。*

結果:

*別の **Java Development Kit** が選択され、アクティブになります。*

バグの報告

***A.3.** 管理インターフェース監査ロギングリファレンス*

ロガー属性のリファレンス

管理インターフェース監査ロギングを有効または無効にする他に、以下の ロガー 設定属性を使用できます。

log-boot

true に設定すると、サーバーの起動時に管理操作が監査ログに含まれます。そうでない場合は **false** になります。デフォルト： **true**

log-read-only

true に設定すると、すべての操作が監査ログに記録されます。**false** に設定すると、モデルを変更する操作のみがログに記録されます。デフォルト**: false**。

ログフォーマッター属性のリファレンス

フォーマッターはログエントリーの形式を指定します。**JSON** 形式でログエントリーを出力する **1** つのフォーマッターのみを使用できます。

例**A.1** ログレコードにタイムスタンプを含めます。

```
/core-service=management/access=audit/json-formatter=json-formatter:write-
attribute(name=include-date,value=true)
```

ログフォーマッター属性

include-date

フォーマットされたログレコードにタイムスタンプが含まれるかどうかを定義するブール値。デフォルト： **true**

date-separator

日付と残りのフォーマットされたログメッセージを分離するために使用する文字を含む文字列。**include-date=false** の場合は無視されます。デフォルト： **-** （これはスペース、その後にハイフン、その後にスペースが続く）

date-format

java.text.SimpleDateFormat が理解しているタイムスタンプに使用する日付形式。**include-date=false** の場合は無視されます。デフォルト： **yyyy-MM-dd HH:mm:ss**

compact

__true__ の場合、__JSON__ を __1__ 行にフォーマットします。新しい行が含まれる値が存在する可能性があるため、レコード全体を __1__ 行にするのが重要な場合は、__escape-new-line__ または __escape-control-characters__ を __true__ に設定します。デフォルト__: false__。

escape-control-characters

__true__ の場合、すべての制御文字（__10__ 進値が __32__ の __ASCII__ エントリー）を __8__ 進の __ASCII__ コードでエスケープします。たとえば、新しい行は __#012__ になります。__true__ の場合、__escape-new-line=false__ を上書きします。デフォルト__: false__。

escape-new-line

__true__ の場合は、新しい行を __8__ 進の __ASCII__ コードでエスケープします（例： __#012__ ）。デフォルト__: false__。

管理インターフェースの監査ログファイルハンドラー属性リファレンス

ファイルハンドラーは、監査ログレコードがファイルに出力されるパラメーターを指定します。具体的には、ファイルのフォーマッター、ファイル名、およびパスを定義します。

ファイルハンドラーの属性

formatter

ログレコードをフォーマットするために使用する __JSON__ フォーマッターの名前。デフォルト： __json-formatter__。

path

監査ログファイルのパス。デフォルト： __audit-log.log__

relative-to

以前指定された別のパスの名前、またはシステムによって提供される標準的なパスの __1__ つ。__relative-to__ を指定すると、__path__ 属性の値はこの属性によって指定されたパスに対する相対値とみなされます。デフォルト： __jboss.server.data.dir__

failure-count

ハンドラーが初期化された後に発生したロギング失敗数。デフォルト__: 0__

max-failure-count

*このハンドラーを無効化する前の最大ロギング失敗数。デフォルト： **10***

disabled-due-to-failure

*ロギングの失敗によりこのハンドラーが無効になっている場合は、値を **true** にします。デフォルト**: false**。*

*管理インターフェースの **Syslog** ハンドラー属性リファレンス*

***syslog** ハンドラーは、監査ログエントリーが **syslog** サーバーへ送信されるパラメーターを指定します。特に、**syslog** サーバーのホスト名と **syslog** サーバーがリッスンするポートを指定します。*

*監査ロギングを **syslog** サーバーへ送信すると、ローカルファイルまたはローカル **syslog** サーバーへロギングする場合よりも、セキュリティーオプションが多くなります。複数の **syslog** ハンドラーを同時に定義およびアクティブ化することができます。*

***syslog** サーバーごとに実装が異なるため、すべての設定をすべての **syslog** サーバーに適用できるとは限りません。テストは* **rsyslog** ***syslog** 実装を使用して実行されています。*

*『**Syslog ハンドラー属性**』は高レベルの属性のみを一覧表示します。各属性は設定パラメーターを持ち、一部の属性は個設定パラメーターを持ちます。**Syslog** ハンドラーの属性に関する詳細は、以下のコマンドを実行します。*

```
/core-service=management/access=audit/syslog-handler=mysyslog:read-resource-
description(recursive=true)
```

***syslog** ハンドラーの属性*

app-name

***RFC-5424** のセクション **6.2.5** で定義された **syslog** レコードに追加するアプリケーション名。指定されない場合、デフォルト値は製品の名前になります。*

disabled-due-to-failure

*ロギングの失敗によりこのハンドラーが無効になっている場合は、値を **true** にします。デフォルト**: false**。*

facility

RFC-5424 のセクション *6.2.1* と *RFC-3164* のセクション *4.1.1* で定義された *syslog* ロギングに使用する機能。

failure-count

ハンドラーが初期化された後に発生したロギング失敗数。デフォルト： ***0***

formatter

ログレコードのフォーマットに使用するフォーマッターの名前。デフォルト： ***json-formatter***。

max-failure-count

このハンドラーを無効化する前の最大ロギング失敗数。デフォルト： ***10***

max-length

ヘッダーを含むログメッセージの最大長（バイト単位）。未定義の場合、デフォルト値は ***1024*** バイト ***(syslog-format*** が ***RFC3164*** の場合) または ***2048*** バイト ***(syslog-format*** が ***RFC5424*** の場合) になります。

protocol

syslog ハンドラーに使用するプロトコル。***udp***、***tcp***、または ***tls*** のいずれか ***1*** つである必要があります。

reconnect-timeout

JBoss EAP 6.4 から利用できます。***syslog*** サーバーへの再接続を試行する前に待機する秒数。イベント接続が失われます。デフォルト： ***-1*** （無効）

syslog-format

syslog 形式： ***RFC-5424*** または ***RFC-3164***デフォルト： ***RFC-5424***

truncate

ヘッダーを含むメッセージの長さ（バイト単位）が ***max-length*** 属性の値よりも大きい場合は、そのメッセージを省略するかどうか。***false*** に設定すると分割され、同じヘッダー値で送信されます。デフォルト***: false***。

付録B 改訂履歴

改訂 6.4.0-47 **Thursday November 16 2017** **Red Hat Customer Content Services**
Red Hat JBoss Enterprise Application Platform 6.4.0.GA Continuous Release